A Practical Interference-Aware Power Management Protocol for Dense Wireless Networks

Paper 21, 14 pages

ABSTRACT

The availability of spectrum resources has not kept pace with wireless network popularity. As a result, data transfer performance is often limited by the number of devices interfering on the same frequency channel within an area. In this paper, we introduce a protocol that manages the transmission power and CCA threshold of 802.11 devices to maximize network performance. The protocol is based on the observation that for a pair of interfering wireless links, it is possible to calculate the *ratio* of the transmit power of the two senders that maximizes overall network capacity. We first present an algorithm that extends this result for dense clusters of nodes and then describe a distributed protocol that implements the transmission power setting algorithm and CCA tuning mechanism. Finally, we describe an implementation of the project in Linux. It uses commercial 802.11 cards and addressed several practical challenges such as protocol stability and calibration. Our experimental evaluation using an 8 node testbed shows that our protocol works well in practice and can improve the performance over default configurations by more than 200%. We also use OPNET simulations to evaluate larger topologies with different node densities.

1. INTRODUCTION

The popularity of wireless is fueling a rapid growth in the number of devices using the unlicensed frequency bands. Resulting dense wireless deployments in hotspots, residential neighborhoods, and campuses increasingly suffer from poor performance due to interference between the large number of devices sharing the same frequency band. This problem is exacerbated by the fact that the default transmit power and carrier sense threshold (aka Clear Channel Assesment or CCA threshold) in 802.11 devices results in inefficient use of the spectrum. In this paper, we describe the design and implementation of a distributed protocol for selecting the transmit power and CCA threshold so as to reduce interference and increase network capacity, while maintaining fairness.

Several projects [21][17][1][15][7] have explored adjusting transmit power and/or the CCA treshold to improve performance. For example, [1] proposed to use the minimum power level necessary to reach a receiver. However, this approach can increase unfairness since nodes closer to their destination may experience higher interference. In fact, several authors [19][17][1] have pointed out that systems that do not tune the CCA threshold along with the transmit power often reduce fairness since low power stations are less likely to be heard. More recent work [17] performs joint transmission power and CCA threshold tuning, but the proposed protocol uses the same transmit power level and CCA threshold for all nodes in a cell, thus limiting opportunities for spatial reuse. Also, practical challenges such as calibration are not addressed in earlier work.

In this paper, we first present an iterative greedy algorithm that uses perfect knowledge about the RF environment to determine the transmit power for all transmissions. The basic idea of the algorithm is to iteratively adjust the transmit power of individual links so that the number of edges in the transmission conflict graph [18] is reduced. We then describe a practical, distributed version of the power control algorithm for 802.11 nodes. In the protocol, each node builds a local topology graph by overhearing packets from its neighbors. It uses the RSSI of the incoming packets and transmission information stored in the packets. Each node then uses its local topology graph to optimize both the transmit power to each neighbor and to adjust its CCA threshold using an altruistic version of the Echos [21] algorithm.

Next, we describe an implementation of the protocol that addresses several practical challenges, including how to stablize the protocol, how to calibrate both the transmit power and RSSI readings of the cards, etc. We present an evaluation of the protocol using both testbed measurements and simulation, focusing on infrastructure 802.11 networks. Measurements in an eight node testbed show throughput gains of more than 200%. Using OPNET simulation, we study larger topologies and the impact of node density on performance. While the algorithm optimizes capacity for any type of wireless link and the implementation address many practical issues, we should note that we do not consider possible

interactions with transmit rate selection, AP selection in infrastructure networks, or multi-hop routing and opportunistic relaying in mesh networks in this paper.

The rest of the paper is organized as follows. We describe our interference model in Section 2. Section 3 discusses our greedy centralized algorithm for power control, extensions for CCA tuning, and power reallocation for further increasing spatial reuse. In Section 4, we present the distributed power management protocol. We discuss our implementation and an evaluation using a testbed and OPNET in Sections 5 through 7. We review related work in Section 8 and summarize our conclusions in Section 9.

2. MODELING INTERFERENCE

In this section, we first review two models that are widely used in the wireless community to identify when concurrent transmissions can occur. We then introduce our interference model and the conflict graph.

2.1 Concurrent Transmission Model

The **physical SINR model** [9][18] predicts that a packet can be successfully decoded if the signal to interference plus noise ratio exceeds some threshold. It takes into account received signal strength from the source and all sources of interference and noise. The SINR model predicts that uniformly increasing power levels increases system throughput by reducing the effects of thermal noise, though such gains are marginal in interference-dominated networks, e.g. dense 802.11 networks.

The **circle model** is a much simpler model that is used implicitly in many papers [14][1]. In this model, every source is associated with a transmission and an interference range. Nodes within the transmission range of a source can decode frames from the source, and nodes within interference range will be prevented from transmitting due to carrier sense [3]. The ranges depend on the power level each source uses. The circle model predicts that using the minimum possible power level minimizes interference – exactly the opposite conclusion of the SINR model!

The right choice of model depends on whether the interface hardware exhibits the capture effect [22], i.e. the ability to decode one transmission when multiple transmissions collide. In particular, it depends on how an incoming transmission is affected by a later frame with a higher signal strength. If the interface captures the stronger signal, then the SINR model is more accurate – otherwise the circle model is a better predictor. To determine the behavior of modern 802.11 hardware, we performed an experiment in a wireless emulator [12]. The experiment uses four laptops equipped with Atheros 5212 cards. We use *netperf* [11] to create two interfering non-rate controlled UDP flows, a

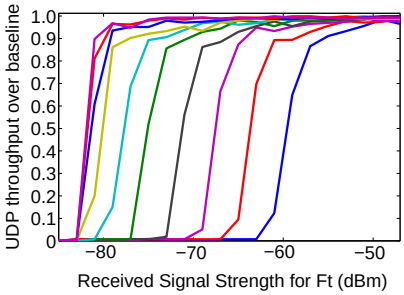


Figure 1: Packet Capture With Interfering Signals for Atheros 5212 cards

targer flow F_t and an interfering flow F_i, with transmit power levels P_t and P_i at the target receiver, respectively. Also, we use the emulator to prevent any interference at both sources, thus eliminating the effects of carrier sense and collisions with link-layer ACKs.

Figure 1 shows the throughput achieved by F_t as a function of P_t. Each curve is for a constant interference level P_i and we changed P_i from -100dBm (lower than the noise level) to -60dBm (high enough to be decoded) in increments of 4dB. If the hardware does not exhibit the capture effect, the throughput of F_t for high values of P_i should be half of maximum throughput. However, Figure 1 shows that except for signal strengths close to the noise floor, all curves have the same shape and are equally spaced. Also, the interfering signal (equally spaced curves) has a similar effect as noise (leftmost curve). These features are consistent with a strong capture effect, showing that the circle model is not representative for this hardware. Similar observations were made in [16], which presents a timing analysis of the capture effects. For Intel cards, capture effect is supported by setting the CCA threshold to a relatively high value [23], i.e. ignore the inteference signal that is below the CCA threshold.

Unfortunately, the SINR model is challenging to work with because each transmission interacts with all other transmissions. Thus, for deriving our protocol, we simplify the SINR model by making two assumptions: 1) interference is dominated by the strongest source (pairwise interference assumption), and 2) we can ignore thermal noise. Recent work [6] has shown that the pair-wise assumption usually holds and we can ensure the second assumption by keeping power levels high enough. Note that this model is similar to the protocol model [18][9], except that we do not assume that all nodes use the same power level.

2.2 Conflict Graph

To concisely represent the interference that is present in a network, researchers have used a conflict graph [18]. Each vertex in the conflict graph represents a link in the wireless network and there is an undirected edge

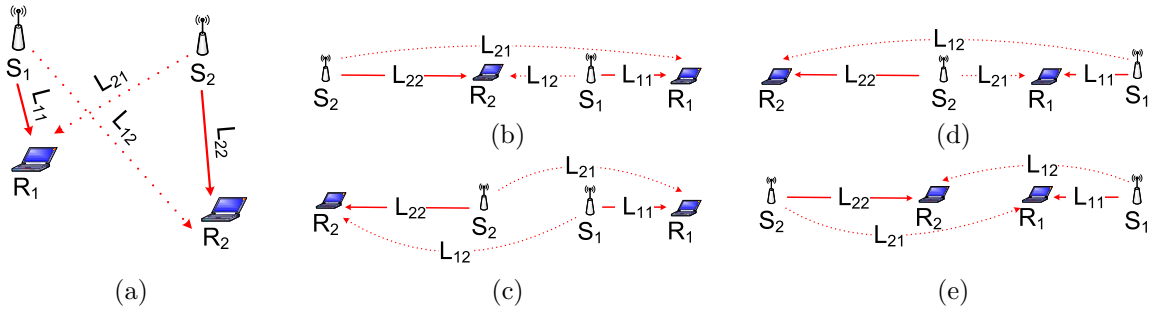


Figure 2: Topologies for two interfering flows: (a) in general, (b)(c)(d)(e) some common scenarios.

between two links if the two links interfere, i.e. they cannot be active at the same time. Clearly the conflict graph depends on the transmit power used by the nodes. We construct the conflict graph based only on the pairwise SINR model described above, independent of any MAC protocol. This is the reason why it is sufficient to use undirected edges: concurrent transmissions should be avoided both when two links interfere with each other and when interference happens in only one direction.

The goal of our work is to increase spatial reuse, which is beneficial for almost all practical network deployments. Since the lack of an edge in conflict graph is equivalent to enabling concurrent transmissions, our goal is equivalent to minimizing the number of edges in the conflict graph. Note that earlier work has generally used more specific optimization criteria, e.g. maximize multi-hop network capacity [18], achieve maxmin fairness [2], or minimize mean delay [17].

3. POWER CONTROL

In this section, we develop a centralized algorithm for per-link power control on a collection of nodes. We first analyze how tranmission power should be configured in a simple two flow scenario. We use the insights gained from this simple setting to design an iterative, greedy algorithm for a cluster of nodes, considering only physical layer effects. We then consider the impact of the MAC layer, specifically carrier sense, and we describe candidate CCA tuning mechanisms. The reason we separate the selection of transmit power and CCA threshold is that the interactions between the two parameters are complex, so concurrent optimization is difficult, if not impractical. Since the transmit powers are the primary factor to optimize, selecting transmit powers first, followed by CCA thresholds, is both practical and effective. While we show in our evaluation that this approach is effective, it does occasionally lead to lost concurrent transmission opportunities. Thus we also introdcue a post-CCA power reallocation algorithm to to further improve spatial reuse.

3.1 Simple Two-Flow Scenario

We begin by considering a simple scenario (Figure 2(a)) where S_1 transmits to R_1 and S_2 transmits to R_2. The SINR at receivers R_1 and R_2 are $SINR_1 = P_1 - L_{11} - P_2 + L_{21}$ and $SINR_2 = P_2 - L_{22} - P_1 + L_{12}$, respectively, where P_i is the transmit power level from S_i to R_i, and L_{ij} is the path loss from S_i to R_j $(i, j \in \{1, 2\})$.[1] Note that independent of the transmit power levels, we have $SINR_1 + SINR_2 = L_{12} + L_{21} - L_{11} - L_{22}$. Power control essentially allocates this sum between the two transmissions, i.e. increasing $SINR_1$ will decrease $SINR_2$. In order to enable concurrent transmission, we need both $SINR_1 \geq SINR_{thrsh}$ and $SINR_2 \geq SINR_{thrsh}$. Note that it may not be possible to satisfy both constraints, so concurrent transmission may be impossible.

We now consider what happens when all nodes use the same power (i.e., *equal power*) or when they use the minimum power level to reach the receiver (i.e., *minimum power*). With equal power, we have $SINR_1 = L_{21} - L_{11}$, and $SINR_2 = L_{12} - L_{22}$. Thus, in Figure 2, equal power performs well in scenario (c) & (d) because $SINR_1 \approx SINR_2$ but poorly in (b) & (e) because $SINR_1 >> SINR_2$. With minimum power, we have $SINR_1 = L_{21} - L_{22}$, and $SINR_1 = L_{12} - L_{11}$. Thus, minimum power performs well in scenario (b) & (e) but poorly in (c) & (d). Intuitively, if the sender is far away from the receiver, the transmission is more likely to be penalized using equal power because its received signal strength is relatively weak. Using minimum power has the opposite problem. If the sender is close to its receiver, it is more likely to be penalized because the interference level is likely to be high.

We constructed the scenarios shown in Figure 2(b)&(d) on a 4-node testbed and used it to compare the performance of different transmission power and CCA threshold settings. We first obtained a baseline throughput by having only one of the sources transmitting. We then ran experiments with both sources active, using the *equal* and *minimum* transmit power settings described above, combined with the *default* 802.11 CCA threshold

[1]We do not assume pathloss obeys the triangle inequality in our algorithms or protocols. However, simulations use this assumption for simplicity.

Table 1: Evaluation result on a 4-node testbed.

Scenario	PC/CCA	LT1(%)	LT2(%)	LT1+LT2
Figure 2(d)	Equal/High	81.4	96.6	**178.0**
	Equal/Def	54.8	60.8	**115.6**
	Min/Def	11.2	102.0	**113.2**
	Protocol	94.3	90.0	**184.3**
Figure 2(b)	Equal/High	20.8	98.8	**119.6**
	Equal/Def	36.4	81.5	**117.9**
	Min/Def	98.8	85.9	**184.7**
	Protocol	94.1	93.3	**187.4**

and a *high* CCA threshold that prevents transmitting nodes from deferring to each other. We also used our proposed protocol for power and CCA threshold selection. The details of our protocol and the experimental setup are described in Sections 4 and 5.

The results, as a percentage of baseline, are shown in Table 1. We see that our protocol enables concurrent transmissions in both scenarios, while neither equal power nor minimum power work well in both scenarios. Specifically, our power control mechanism alleviates the hidden terminal problem caused by minimum power in scenario (d), by equal power in scenario (b), and also solves the exposed terminal problem caused by default CCA threshold in scenario (d).

3.2 Centralized Algorithm

We now present the greedy, iterative power control algorithm that forms the basis of our distributed protocol. It assumes that every node has complete knowledge of the network topology (i.e. path loss between nodes) and current configuration (i.e., power levels used by sources). The basic idea of the algorithm is to greedily decrease the number of edges in the conflict graph. We do per-link power management since it offers more opportunities for special reuse than per cell configuration used in earllier work [17]. Thus, for example, in an infrastructure network, each client contributes two links to the network, one from itself to the AP, and the other from the AP to itself.

In the previous section, we showed how to set the ratios of transmit power between two links to allow simultaneous transmission, if possible. However, if there are multiple links, the choice made for one link may not be compatible with the choices for other links. Thus, we use a greedy algorithm that iteratively allows more concurrent transmissions by removing edges from the conflict graph. We use the following notations in Algorithm 1. $src(t), dst(t)$ are the source and destination of link t while $P(t)$ is the power level currently used on that link. For any two nodes n, n', $L(n, n')$ is the path loss from n to n'. We assume that each wireless device has a range of discrete tunable power levels, e.g. 0dBm to 20dBm; in practice, power range is typically limited by noise consideration (lowerbound) and power restrictions / hardware limitation (upperbound).

In each iteration, and for each link t, the algorithm examines the power level used on all other links (line 6) and the topology (line 7-10) to determine what power level would allow simultaneous transmission with the other links (line 11-12). It then picks the power level that can remove the most edges from the conflict graph (line 18). The new power level will be used if it allows more concurrent transmissions than that in the previous iteration (line 19-21). In each iteration, the number of edges in the conflict graph decreases, and the algorithm converges when no more edges can be removed from the conflict graph. Also, by using this algorithm, the source that needs maximum power level will hit the power limit, and then all other sources will keep an appropriate ratio to that source. Simulation results for this algorithm (omitted for space reasons) show that although the algorithm is greedy, its performance is very close to the upper bound.

Algorithm 1 Iterative Power Control Algorithm

```
1.  while not stable do
2.    /* For each link t, use v[i] to determine how often
      concurrent transmission is possible with power
      level i */
3.    for all t do
4.      clear(v)
5.      for all t' ≠ t do
6.        P' ← P(t')
7.        L11 ← L(src(t), dst(t))
8.        L12 ← L(src(t), dst(t'))
9.        L21 ← L(src(t'), dst(t))
10.       L22 ← L(src(t'), dst(t'))
11.       Pmin ← L12 − L22 − SINRthrsh + P'
12.       Pmax ← L11 − L21 + SINRthrsh + P'
13.       /* [Pmin, Pmax] is the range of P(t) such that
          t and t' can transmit together */
14.       for i = Pmin to Pmax do
15.         v[i]++
16.       end for
17.     end for
18.     Find the Pm such that v[Pm] is maximum.
        /* last[t] is used to ensure convergence: change
        power level only when more concurrent trans-
        missions are allowed */
19.     if v[Pm] > last[t] then
20.       P(t) ← Pm
21.       last[t] ← v[Pm]
22.     end if
23.   end for
24. end while
```

3.3 Interactions with Carrier Sense

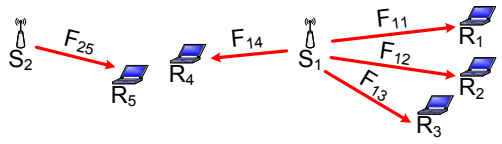


Figure 3: Example of Power Reallocation.

Algorithm 1 did not consider MAC layer effects such as carrier sense. Past work has however noted the importance of CCA tuning when doing power control [17, 1]. We now describe candidate CCA tuning mechanisms, leveraging earlier work [7, 21].

In Alpha [7], every source uses a fixed product α (product in watts, or sum in dB) for power level and CCA threshold. Having every node uses a same alpha ensures the symmetry property [17], where either both nodes hear each other or neither can hear each other. In Echos [21], every source picks a CCA threshold that allows it to hear all the transmissions that interfere with its current transmission. Simulations (Section 7) show that Echos can lead to starvation for some transmissions. The reason is that each node in Echos greedily optimizes for its own transmissions. To address this problem, we introduce Altruistic Echos (AEchos). It is similar to Echos, but sets the CCA threshold to hear all the transmission that interfere with the current transmission or will be interfered with by the current transmission.

Both Echos and AEchos use localized decisions and can easily be incorporated into the power management protocol. Alpha is more complex to integrate since it requires network-wide agreement on α. In our evaluation, we manually set the α.

3.4 Post-CCA Power Reallocation

While selecting power levels first and then CCA thresholds is both practical and effective, it does occasionally lead to lost concurrent transmission opportunities. Figure 3 shows an example for a one-hop network. Receiver R_4 is in a bad location since flow F_{14} interferes with F_{25}, but otherwise all flows can transmit simultaneously. Suppose that Algorithm 1 assigns the same power level from S_1 to all its receivers. S_2 can now either use a high CCA threshold that causes R_4 to starve, or it can use a low CCA level that wastes the spatial reuse opportunities with F_{11}, F_{12}, F_{13}. A better solution is to have S_1 reallocate power levels to its receivers: by using a lower power on F_{11}, F_{12}, F_{13}, S_2 can set its CCA threshold to defer to F_{14} but ignore F_{11}, F_{12}, F_{13}, thus allowing spatial reuse with F_{25} without starving R_4. Power reallocation is possible because the iterative power control algorithm in fact produces a range of power levels that result in the same conflict graph.

The basic idea behind power reallocation is to assign higher power levels to receivers that experience higher level of interference, e.g. R_4 in the above example, while keeping the power levels within the ranges produced by the iterative algorithm. The algorithm is executed locally on each sender, but only when the sender has multiple receivers, e.g. in infrastructure networks, only APs need to execute it. To reallocate power levels, the sender first sorts all its receivers by the level of interference they experience (high to low) and then tries to improve spatial reuse by reallocate the transmit power according to the sorted order. It always keeps the transmit power within the range produced by the iterative algorithm, even this means that concurrent transmission opportunities cannot be increased. Note that the effect of power reallocation, i.e. improved spatial reuse, will take place in the next iteration when other senders adjust their CCA thresholds accordingly.

4. DISTRIBUTED PROTOCOL

In this section, we present the distributed power management protocol based on the algorithm described in Section 3. We describe topology information collection and packet reception and transmission processing.

4.1 Data Collection

In order to create a local interference graph, nodes need information about the path loss on nearby links. This is obtained by having each source insert transmit power and path loss information in each packet, specifically, the power level used for the current packet, the path loss measurement to the destination, the path loss measurement to a randomly chosen third node, and a bit to indicate whether this is a high power frame (see below). The power level used for this packet will be used by the receiver to calculate the path loss. The sender's path loss measurement to the destination will be used by the receiver to help calibrate the power levels of the cards (Section 5.5). Finally, by including the path loss to a third node, we are effectively creating a gossiping channel that allows a source to learn the path loss of links they are not part of. This is used by the source to estimate the interference at its destinations. The overhead of extra information is less than 1% for a full-sized data packet.

Let us illustrate the data collection phase using the example in Figure 2(a), S_1 needs the following information to determine power level: $L_{11}, L_{12}, L_{21}, L_{22}, P_2$. Then, P_2 can be extracted from packet when S_2 is transmitting. L_{11} can be measured when R_1 is transmitting, L_{12} can be measured when R_2 is transmitting, and L_{21} can be measured by R_2 when S_2 is transmitting and sent to S_1. Finally, L_{22} can be measured and inserted into a packet by S_2.

When a node uses a low power level, other transmitters may not be able decode its packets and collect the

above information. To avoid this problem, nodes periodically (e.g. every 2 seconds) transmit announcements at the maximum power level. These announcements can be piggybacked on other packets, such as beacons or ACKs. A similar problem happens when a source uses a high CCA threshold and has a lot of packets to send: it will transmit a significant fraction of the time without deferring to other sources, thus preventing it from overhearing packets from other sources. Though this does not affect performance after the protocol converges, it will affect its ability to change its view of the topology, e.g. when a node moves or joins the network. In order to avoid this problem, each node periodically set its CCA threshold to low level (e.g. 20ms every second).

Each node in the system operates in promiscuous / monitor mode and upon receiving a packet, the node updates its topology information and configuration. For each of its active flows, a source node keeps several lists: T_{sd} to store current transmit power level, current CCA threshold and measured path loss for the source to the flow's destination; T_{so} to store path loss from the source to other nodes; T_{do} to store path loss from the destination to other nodes; and T_{oo} to store the power level, and path loss of other flows.

4.2 Packet Transmission/Reception

When node S_1 is about to transmit a data or control frame to destination R_1, S_1 will first search for the power level, CCA threshold, and path loss for R_1. If such information is unavailable, it will use the default power level and CCA threshold, and it inserts an invalid value for path loss in the packet. This should only happen for the first frame between the sender and receiver, e.g. an association request in infrastructure network. After a few packet exchanges, S_1 will have initial measurements for power level, CCA threshold, and path loss, and will use those for the transmission. In addition, S_1 will also piggyback power level, path loss, and the picked entry from the T_{so} list onto the frame. The frame is then added into the device queue for transmission.

Upon receiving a packet from R_1, node S_1 measures the RSS of that packet and it extracts power level, path loss from R_1 to another node, and path loss from R_1 to its destination from the frame. The path loss from R_1 to S_1 is calculated by subtracting RSS from power level used for the packet. Then, S_1 will determine which lists to update. If the packet is destined to S_1, or the packet is from its sender to another receiver, then it will update the T_{sd} and T_{do} lists. Otherwise, the node will update the T_{so} and T_{oo} lists. If there is an update to any of the lists, S_1 will recalculate the configurations for all its destinations.

4.3 Absolute Transmit Power

Algorithm 1 presented in Section 3 determines the *ratios* between the transmit powers for all the wireless links, i.e. uniformly shifting the power levels up or down does not affect the results. We also noted that, in practice, the range is limited by noise consideration (low end) and power limitations (high end). While the optimization algorithm is not sensitive to absolute power levels, the protocol generally benefits from using high power levels. The reason is that this increases the amount of topology information that nodes can collect by overhearing packets. As a result, our protocol first calculates ratios and then assigns the maximum power level (usually 20dBm) to the link requiring the highest power. The power for the other links are then derived using the ratios.

5. IMPLEMENTATION

We now present an implementation of the protocol using commercial wireless hardware. We also discuss several practical challenges including protocol stability and device calibration.

5.1 Ideal System Requirements

The following is a list of ideal hardware/firmware features needed to run our protocol. Since these features are not well supported in many current devices, we describe design alternatives and "work-arounds" in the following sections.

Per-packet Transmit Power and CCA Threshold Control. Control over transmit power and CCA threshold is central to our design. To the best of our knowledge, the only publicly available solution that supports this is the Intel 2915/2200 card with AP firmware and driver. Also, since our protocol works on a per-link basis, these controls need to operate on a packet granularity, at least for senders with multiple receivers. Unfortunately, the Intel 2915/2200 AP firmware does not support these functions *stably* on a per-packet basis. We believe that this weak support is a result of a lack of demand and that future cards should be able to support per-packet reconfigurability.

Receiver Threshold Control. The receiver threshold is used by a radio to determine when to decode an incoming signal. If a signal is below this threshold, the radio is not activated. On Intel 2915/2200 hardware, the receiver threshold is coupled with CCA threshold [23]. As a result, when a node uses a high CCA threshold, it can no longer decode many low power transmissions, limiting its ability to gather topology information in our protocol. Even worse, the node can miss critical packets, e.g. an AP may miss association requests from a new client even if it is sent using a high power level. We want the receiver threshold to be set independently from the CCA threshold. Again, this should be possible in future cards.

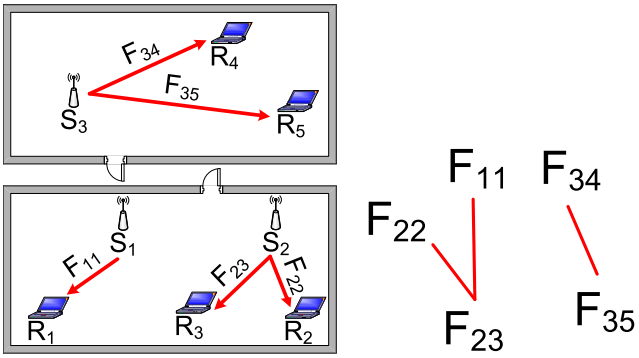
Accurate Signal Strength Measurements and Transmit Power Settings. Our protocol depends on accurate RSS measurements and accurate control of the transmit power. Our experience shows that the RSSI readings from the Atheros cards and the transmit power setting of the Intel cards we used are fairly linear. However, our experience, confirmed by earlier work [8][12], shows that wireless cards are uncalibrated, both in terms of transmit power levels and RSSI readings. This is the case even for cards of the same model from the same vendor. As a result, changes in these values are measured accurately but the absolute values of the readings cannot be compared across cards. Since nodes exchange both transmit power and RSSI values in our protocol, we need cards to be reasonably calibrated. Calibration by the manufacturers only solves part of the problem, because cards can become uncalibrated over time due of drift. Thus automated calibration is necessary and we describe our mechanism in Section 5.5.

5.2 Implementation Setup

We implemented our power control protocol with AEchos CCA tuning mechanism (the choice is based on simulation results in Section 7) in Linux using nodes equipped with one Intel 2915 card and two Atheros 5212 cards. This setup is needed so we can work around the shortcomings of the current commercial cards. The Intel card supports transmit power and CCA tuning and it is used as the primary transmission interface. To deal with the receiver threshold problem, one Atheros card, using a default receiver threshold, is used to monitor all ongoing traffic. This allows us to collect topology information, which is then used to set the txpower and CCA threshold on the Intel card.

When measuring the performance for bidirectional traffic, e.g. TCP, both senders and receivers must be able to control their transmit power and CCA threshold. To achieve this, nodes use the Intel card (in AP mode) for transmission (providing control over power and CCA threshold) and the second Atheros card for receiving (capture effect presents); we realize this set up by configuring the routing tables appropriately. Note however that this set up only allows a sender to talk to one receiver at a time since the Atheros card of the sender can only be associated with the Intel card (which needs to run in AP mode) of one receiver.

For experiments that use multiple receivers, we simplify the previous setup by having receivers not use their Intel card. Instead, all receivers transmit and receive with their Atheros cards. We use one-directional UDP in those experiments so that the lack of transmit power and CCA control on receivers will not affect the results. Another problem with the multi-receiver set up is that we do not have per-packet control over the transmit power and CCA threshold, so we cannot easily interleave transmission to multiple receivers. To overcome this problem, we switch between receivers at a coarse granularity: the sender transmit for 10 seconds to each of its receivers in turn. All possible sender-receiver combinations are enumerated, and the throughput of each flow is calculated as the mean throughput of that flow across all combinations (i.e. the full duration of the experiment).



(a) Topology and association (b) Conflict graph

Figure 4: The topology and association of our experiments with 8 nodes

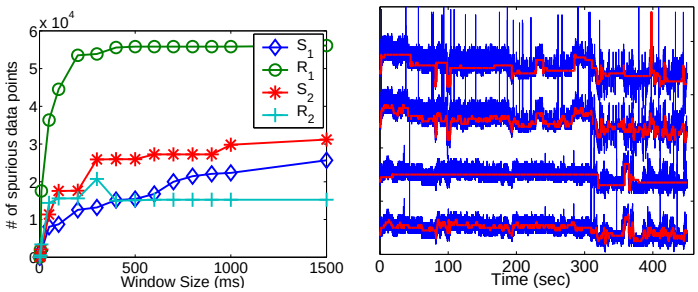
Note that our protocol does not apply to link-layer ACKs. There are two implications. First, since the wireless card sends the ACKs, we cannot control the transmit power that is used (note that ACKs do not use carrier sense so the CCA threshold is not relevant). Second, since link-level ACKs are not delivered to the device driver, they cannot be used to update the topology information. This loss of information is especially critical in unidiectional tests. We overcome this problem by exchanging several packets between senders and receivers at the start of each experiment so all nodes have relevant topology information.

All our experiments use the setup in Figure 4.

5.3 Smoothing RSSI Readings

RSSI readings will always vary over time due to noise, interference and environmental changes. We found that many variation in RSSI readings tend to have a short duration. In our protocol, stability of the RSSI readings is more important than the accuracy of individual readings, so it is advantageous to filter out these variations even if they reflects actual signal strength changes and are not an artifact of the measurement. The reason is that rapidly changing RSSI readings could lead to fluctuations of signal strengths throughout the networks and possibly network instability.

A common approach to smoothing RSSI readings is to take a moving average [13]. However, moving averages of RSSI can still fluctuate greatly, even with a large averaging window of 1 second. Since most RSSI variation is on short time-scales, we take the approach



(a) Effect of Window Size (b) Effect of smoothing

Figure 5: Characterizing the Effect of Filtering

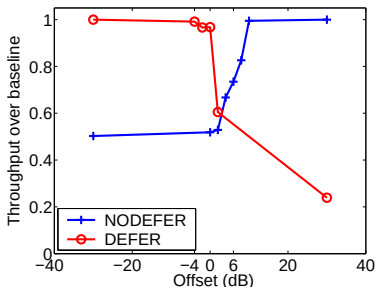


Figure 6: Assessment of CCA thresholds

of removing short-lived changes from RSSI readings, so they do not contribute to the moving average. We do this by filtering out all RSSI values that (1) that deviates by more than 2dB from the current moving average and (2) is short lived. The deviating RSSI reading is considered to be short-lived, if within the averaging window, there exists a time interval K, where all RSSI values in the interval fall within 2dB of the moving average. We set the value of K to be 10% of the averaging window. Otherwise, the change is considered long-lived and it is incorporated into the average.

In order to determine the appropriate window size for filtering readings, we carry out the following experiment. Node S_3 transmits and S_1, S_2, R_1, and R_2 record the RSSI readings. Figure 5(a) shows how many readings are discarded at the four receivers. Note that larger windows limit the rate at which we adjust to long-lived changes. As a result, we use a 500ms window in our protocol since it is the smallest window that removes most spurious data points. Figure 5(b) shows the result of applying our filtering algorithm to the RSSI readings from two RSSI measurement streams. The first and third curves result from our smoothing algorithm with a 500ms window, and the second and fourth curves are from a simple moving average with a 1 second window for comparison. The result shows that the our filtered readings are both stable and reasonable. We will also show later that the filtering does not hurt throughput.

5.4 CCA Offset and Transmit Power Granularity

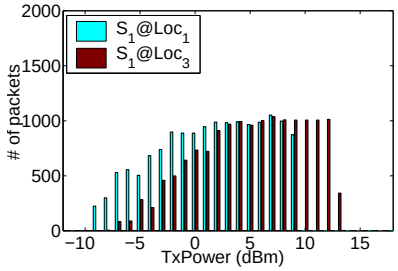
Measurement noise, calibration accuracy and other factors mean that small changes in CCA threshold and transmit power will have little impact on system behavior. We explore this granularity issue, focusing on identifying what CCA thresholds must be used to reliably defer or ignore competing signals and how power levels must be adapted to accomodate these CCA thresholds.

To determine how low S_1 should set its CCA threshold to correctly defer to another transmission, we let F_{22} transmit with a fixed low CCA threshold, and set the transmit power of F_{11} to interfere with F_{22} (Figure 4). Figure 6(DEFER) shows F_{22}'s throughput for different CCA threshold settings on S_1. The throughput is plotted as a ratio to the throughput of F_{22} when S_1 uses a default low CCA threshold and the offset is plotted as an offset from the average RSS from S_2. It shows that with an offset -4, the ratio is close to one. This indicates that in order for S_1 to defer to transmission F_{22}, the CCA threshold on S_1 should be set 4dB lower than the interfering signal. To measure how high the CCA thresholds need to be to ignore a signal, we let F_{22} transmit with a fixed high CCA threshold, and F_{11} and F_{22} are configured so that they can transmit simultaneously. The baseline is the throughput of F_{11} when S_1 uses a very high CCA threshold. Figure 6(NODEFER) shows the ratio of F_{11}'s throughput relative to this baseline for different CCA thresholds on S_1. It shows that the CCA threshold need to be 6dB higher than average RSS to achieve full spatial reuse.

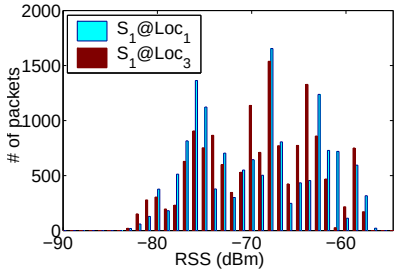
A related issue is what the granularity should transmit power settings be. This is especially important for the reallocation algorithm in Section 3.4 since it may be necessary to distribute a number of values within a limited range. While the transmit power can be any integer, in practice, the useful transmit power spacing depends on CCA granularity, which is essentially the difference between the two CCA offsets, e.g. 10 dB in this case. For example, the CCA threshold on S_1 should be $P_{22} - L(S1, S2) - 4$ to defer to transmission F_{22}, where $L(S1, S2)$ is pathloss between S_1 and S_2. And at the same time, the CCA threshold should be $P_{23} - L(S1, S2) + 6$ to ignore transmission F_{23}. Thus $P_{22} - L(S1, S2) - 4 = P_{23} - L(S1, S2) + 6$, or P_{22} should be 10 dB higher than P_{23}. We carried out similar experiments on other senders and all had a granulatity of less than 10dB. Thus we use a spacing of 10dB in our experiments, so given the txpower range of our cards, each sender can support 3 receivers without losing spatial reuse.

5.5 Calibration

Our protocol relies on both transmit power settings and RSSI readings on network cards to be accurate.



(a) TxPower



(b) RSS

Figure 7: TxPower and RSS Histograms of S_1's at two different locations

Even if hardware manufactures calibrate network interfaces, the devices can become uncalibrated over time [8]. While manual calibration using a signal analyzer and generator is possible [8], this is not a practical solution is most deployments. In this section, we describe an automated calibration mechanism.

As observed in previous work, the relationship between output power and transmit power value on each card is roughly linear but there is typically an offset; offsets of up to 20 dB have been observed in [8]. The relationship between actual RSS and RSSI readings is similar and offsets of up to 10dB have been observed in [12]. To formalize the calibration problem, let $\Delta txpow_i$ and $\Delta RSSI_i$ denote the difference between actual and reported transmit power levels and RSSI readings at node i, respectively. For our protocol, we need to calibrate the cards relative to one particular node i, i.e. solve $\Delta txpow_j - \Delta txpow_i$ and $\Delta RSSI_j - \Delta RSSI_i$ for any node j. Put it another way, after calibration, we only need to know the differences in the offsets between all nodes, and share the same unknown $\Delta txpow$ and $\Delta RSSI$. The two unknowns will not affect the result of our protocol because: 1) every pathloss will have an offset of $\Delta RSSI$, and this does not affect the calculation of SINR for either flow, 2) every txpower setting will have an offset of $\Delta txpower$, which doesn't change SINRs nor the conflict graph, 3) $\Delta RSSI$ and $\Delta txpower$ does not need to be equal because CCA thresholds are set based on observed RSS, which already includes an offset of $\Delta txpower$.

We calibrate the RSSI readings based on the assumption that receiver sensitivity of all cards are the same.

Table 2: RSSI offset from R_1 (in dB) with the different cards

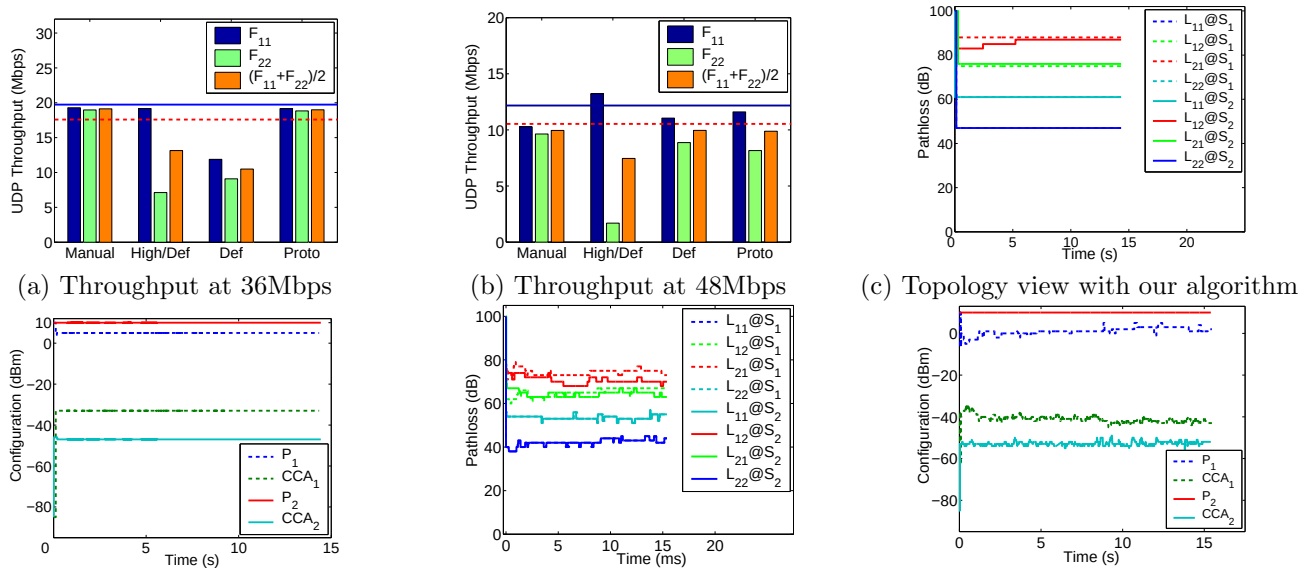
Experiment	S_1	R_1	S_2	R_2
1	1	0	2	0
2	1	0	-1	2
3	1	0	3	0

We let one node send packets with different transmit power levels and all other nodes record the RSSI of the packets they received. For nodes that cannot receive the packets at low transmit powers, the histogram of the RSSI readings of the first N received packets should be independent of their location, but depend only on the difference between their $\Delta RSSI_i$'s. In contrast, the histogram of the transmit power level of the first N received packets is determined by its location, i.e. pathloss. We compare the RSS histograms with different offsets, and choose the offset with the closest distance, i.e. $argmin_k \sum_i |P_i - Q_{i+k}|$ for two histograms P and Q. Figure 7 shows an example of transmit power and RSS histograms for the same device at two different locations with $N = 15000$. It shows that even though the device has a different txpower histograms at two locations, i.e. different pathlosses to two locations, the RSS histograms are roughly the same. Note that if a node can receive packets from the lowest power levels, it should be excluded because the histogram does not reflect its sensitivity, but its location instead. Also, this mechanism is susceptible to external interferenc and such RSSI calibration should be carried out when external interference is expected to be low.

In our experiments, we used S_3 as the sender to calibrate the RSSI readings of S_1, S_2, R_1, and R_2. We ran the experiments three times, with receivers at different locations. Table 2 shows the offsets of different cards from R_1. Ideally, the offsets would all be consistent, and this is nearly true for experiment 1 and 3. However, experiment 2 produces a slightly different callibration, which might be caused by external interference during the experiment. Thus we use the RSSI offsets from experiment 1 on each node. Also, compared with the calibration errors observed in previous work, the calibration errors in our testbed are small because all the wireless devices are newly purchased.

After the RSSI readings are calibrated, we calibrate the transmit power levels based on the law of channel reciprocity, i.e. the pathloss from node A to B is the same as that from B to A at each point in time. This means that nodes can detect the offset between their transmit powers by exchanging their views of the pathloss between them (Section 4.1); the difference between the two reported pathloss values is the offset.

6. EXPERIMENTAL EVALUATION



(a) Throughput at 36Mbps (b) Throughput at 48Mbps (c) Topology view with our algorithm

(d) Configurations with our algorithm (e) Topology view without filtering (f) Configurations without filtering

Figure 8: UDP Experiments with 4 nodes, i.e. 2 flows F_{11} and F_{22}.

In this section, we evaluate our protocol on a 8-node indoor testbed, as shown in Figure 4(a). The nodes are located in two rooms on opposite sides of a hallway.

6.1 UDP Throughput & Protocol Behavior

To evaluate how well RSSI smoothing and calibration work, we carry out an experiment involving four nodes, S_1, R_1, S_2, R_2, forming two UDP flows F_{11} and F_{22}, and using the UDP setup. We compare our protocol with several other mechanisms, 1) Manual: manually tuned txpower with high CCA threshold, to obtain similar throughput for both flows, this shows the best that our protocol can achieve, 2) High/Def: default transmit power with a high CCA threshold, which shows whether two flows interfere or not, and 3) Default: default transmit power and CCA threshold. Transmit rates are set manually, and we show results from 36Mbps and 48Mbps, since these rates result in different conflict graph. All throughputs as shown are an average over 4 runs.

Throughput. Figure 8(a) shows the UDP throughput for the data rate of 36Mbps. The horizontal lines shows the throughput when only one flow is active. The solid line is for a high CCA threshold, which measures the maximum possible throughput, and the dashed line is for the default CCA threshold, which shows impact of other traffic in our test environment. When both flows are active, they cannot achieve full throughput when using the same power level, since F_{11} interferes with F_{22} (but not vice versa) as shown by the High/Def results. Simultaneous transmission is possible if F_{11} use a lower power level, as shown in Manual. The throughput of our protocol is very close to the upper bound of Manual. For Default, note that even with the low, default CCA thresholds, when both transmissions can hear each other, throughput is not exactly fairly shared; this is probably caused by collisions that affect contention window sizes and thus fairness.

Table 3: TCP Performance

Data Rate	Pwr Ctl	LT1(%)	LT2(%)
36	Bi	94.6	90.7
	Uni	96.3	8.6
48	Bi	91.7	87.7
	Uni	90.6	85.6

Figure 8(b) shows the UDP throughout for the data rate of 48Mbps. The solid and dashed lines are similar to those in Figure 8, but the throughputs are halved because, at 48Mbps, both flows always interfere with each other. From the High/Def bar, we see that F_{11} has a better SINR than F_{22} when nodes use the same power. The throughput of our protocol is roughly the same as the default configuration. Fairness is slightly worse than Manual since our CCA threshold selection is somewhat prone to noise in the RSSI values. These results show that our protocol correctly identifies when flows interfere and that the throughputs of our protocol are very close to the upper bound.

Stability. Figure 8(c) shows the view of the topology at nodes S_1 and S_2 over time, and Figure 8(d) shows the transmit powers and CCA thresholds they choose. Compared with the results without filtering, Figure 8(e)&(f) respectively, our protocol is quite stable and the topology views at different nodes are roughly consistent.

Table 4: Convergence Time after Sudden Movement

Movement	Without Low CCA (seconds)	With Low CCA (seconds)
R_1 away from S_1	15.5	1.2
R_1 to S_1	0.8	0.7

6.2 TCP Throughput & Bidirectional Traffic

In this experiment, we run TCP bulk transfers from S_1 to R_1 and S_2 to R_2 at the same time. Table 3 presents the TCP throughput for data rate of 36Mbps and 48Mbps as a percentage of the throughput achieved when one transmission is active. "Bi" power protocol refers to a setup where we perform power control and CCA tuning on both link between the senders and receivers, while the "Uni" results are for a setup where we only perform power control on the sender. The fact that TCP performs well at 36Mbps in the Bi setup but not in the Uni setup shows that although TCP ACKs are small, their transmissions can have a significant impact. At 48Mbps, the throughputs in the Bi and Uni setups are almost the same since all nodes use low CCA thresholds.

6.3 Mobility & Convergence

We evaluate how quickly our protocol can converge after a sudden node movement. Since bidirectional traffic is necessary for the nodes to rebuild the topology, we use the TCP setup in this experiment, but since TCP may interact with packet losses due to movement, we use bidirectional UDP traffic instead. Table 4 shows the convergence time after moving R_1 close to or away from S_1. It is calculated as the time from the earliest observed change in topology to the last change plus an additional 500ms delay from the smoothing algorithm. Note that the results exclude the duration between the actual movement and when the node receives a first packet after the movement.

The results show that periodically lowering the CCA threshold, as described in Section 4, is necessary to deal with such changes. Without this optimization, the convergence time for R_1 moving away from S_1 is very long because it requires a change from high CCA to low CCA and such change is difficult when the flows are saturating the network. Convergence time is much shorter when R_1 is moving towards S_1 in both cases because a change from low CCA to high CCA is easier.

6.4 More Complex Scenarios

In this experiment, we use all 8 nodes in our testbed. A manually generated pair-wise conflict graph for the 36Mbps data rate is shown in Figure 4(b). The throughput of each flow for data rate of 36Mbps is shown in Figure 9. Given the conflict graph, our protocol achieves

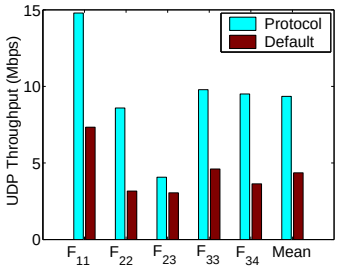


Figure 9: The throughputs for 8 node setup.

almost optimal throughput, which is 200% better than the default configuration.

First, let us look at F_{11}, F_{22} and F_{23}. In this experiment, F_{11} interferes with F_{23} but F_{22} does not. Thus, our protocol makes F_{22} use a lower power level, which allows F_{23} to chose a CCA threshold that allows concurrent transmission with F_{22} but defers to F_{11}. The transmit power spacing of 10dB works in this case and the throughput of F_{23} is only slightly lower than expected. Second, F_{34} and F_{35} can both transmit simultaneously with all other flows. At S_3, F_{35} uses a power level of 10dBm and F_{34} uses 0dBm. Though F_{35} uses a high power level, it gets slightly lower throughput than expected. This is probably because the pair-wise interference assumption does not quite hold, since F_{35} can achieve full throughput when only one of the flows F_{11} and F_{23} is active. Also, the performance of the default configuration is better than expected. The reason is probably that the nodes are in another room, and the RSSI readings can sometimes drop below the default CCA threshold due to changes in the environment.

7. OPNET SIMULATION

We now use the OPNET simulator to study several properties of our protocol: network capacity, fairness, convergence, and performance in large grids. We use two node placement models. 1) Clustered placement: APs (sources) are uniformly distributed, and the clients (recivers) are uniformly distributed in a disk of a chosen radius center at a randomly chosen AP, and 2) Random placement: APs and clients are uniformly distributed, and clients associate with the closest AP. Since the results from the random placement model are similar to those from the clustered model, we only present the results for the clustered model. In our simulations, all APs operate on the same channel and use the same data rate. We simulate 10 APs and 10 clients within a 100m×100m grid, placed according to the clustered model. We vary the cluster radii from 3m to 21m. The results for each radii setting are the average of three different random topologies. The start time for traffic flows is uniformly distributed in the first 20 seconds and we run each experiment for 20 minutes. Traffic is generated using an exponential on-off process with a traffic

Table 5: Network Capacity (Mbps) with different radius size (m)

CCA	PC	3	9	15	21
Echos	Iter	**31.4**	**28.1**	**17.9**	**15.2**
	Min	29.9	24.2	15.9	8.70
	Equal	29.0	19.3	13.4	9.46
AEchos	Iter	**30.8**	**27.6**	**17.2**	**12.7**
	Min	30.2	19.8	14.1	8.37
	Equal	25.3	18.0	11.3	7.00
Default	Iter	31.4	23.5	14.0	8.15
	Min	**30.3**	**22.5**	**13.8**	**7.61**
	Equal	**6.40**	**6.31**	**6.12**	**5.96**
Alpha	Iter	32.0	28.4	9.60	8.60
	Min	30.3	24.1	12.5	6.48
	Equal	28.2	18.7	12.1	7.75
No	Iter	16.7	11.8	6.51	4.32
	Min	15.6	11.7	5.37	3.09
	Equal	15.2	9.05	2.98	2.49

demand for each node of about 2Mbps.

We simulate all combinations of CCA tuning and power control. For power control, the choices are our iterative algorithm, the minimum power to reach a receiver, and all nodes using equal power. For CCA tuning, the choices are Alpha tuning, Echos, AEchos, default CCA threshold and disabled carrier sense (suggested in [10]). Equal power combined with default CCA is essentially the default configuration in 802.11. For Alpha tuning, a relatively high α is aggressive and would lead to more collisions and a relatively low α is conservative and would lead to more exposed terminal problems. We pick the α manually such that the lowest link throughput would be similar to what we observe using iterative power control with AEchos.

Network Capacity. Table 5 presents the simulation results for network capacity (total throughput across all nodes). For power control, iterative power control improves capacity by 2 to 75% over minimum power, and by 8% to 82% over equal power. This improvement also increases as the average radius size increases, i.e. more diversity in client-AP distance. Minimum power with default CCA threshold works well when the radius is short, but the performance degrades when the radius is long. For CCA tuning, Echos provides the best capacity when the radius is longer than 12m, and Alpha has the best capacity when radius is short. The reason that Alpha's performace degrades with the radius is that the iterative power control does not consider the CCA tuning mechanism, and can create asymmetric physical layer interference. For example, one link may use low power to enable concurrent transmission with another link, while other links uses high power level, causing asymmetric interference. Thus, if the α is set aggressively, then a low power link can starve. As a result, we set α more conservatively to prevent starvation. However, this causes the network capacity to decrease dramatically. The results also show that AEchos performs similarly to Echos, with the difference ranging from 2 to 20% for iterative. When equal power and default CCA threshold are used, the simulated regions is small enough that all sources defers to each other. Thus, the network throughput is about the same, independent of the radius. Disabling carrier sense does not work very well in the simulation, since the traffic demand is high and many collisions happen. However, consistent with the observation in [10], the throughput can be higher than using the default configurations. We also changed the number of APs, while keeping the ratio of AP-AP distance to client-AP distance to about 4.5. The results from this setup are consistent with the above. This setup did show that the improvement of iterative approach increases with the number of APs.

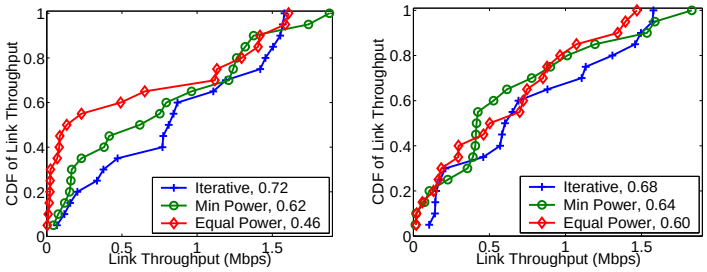
Table 6: Protocol Convergence Time (sec)

# AP	WO/ Low CCA	W/ Low CCA	Low Traffic
10	39	25	12
20	138	121	37

Fairness. Figure 10 shows the CDF of link throughput for a radius setting of 15m. The numbers in legend are the Jain fairness indices for each curve. In Echos (a) and AEchos (b), the curves for the iterative algorithm are roughly to the right of the other two curves, indicating that it provides better throughput for all links. Iterative also has a better fairness index. In general, Echos has better throughput than AEchos, but we observed that Echos results in starvation for some nodes. Starvation was especially a problem for Echos in large radii. AEchos provides a reasonable balance between throughput and avoiding starvation.

Convergence. Table 6 shows the convergence times for iterative power control with and without periodic low CCA threshold for different numbers of APs. The time is measured from the start of the experiment until the configuration is stable. Remember that flows are started uniformly during the first 20 seconds, so convergence may be slow during that period. It shows that the convergence time of the protocol can be reduced by periodically lowering the CCA threshold. However, the convergence time can still be more than two minutes. This is because the duration of low CCA is only 20ms in our protocol, and when the traffic demand is high, collisions can prevent nodes from successfully overhearing enough traffic to collect topology information. At lower traffic rates, the convergence time is significantly lower (last column in Table 6).

Periodic Announcement & Large Grid. Figure 11 shows the CDF of link throughput with and without periodic announcement. The simulation uses



(a) Echos (b) AEchos

Figure 10: CDF of link throughput

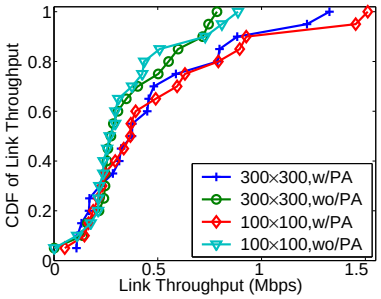


Figure 11: Periodic Announcement & Large Grid

the iterative algorithm and AEchos. We see that link throughputs are much improved with periodic announcements; the network throughput is improved by about 47%. We also ran the same experiment for a 300×300 grid. In that case, not all nodes can hear each other so their local topology graph is not complete. Figure 11 shows that this only slightly reduces link throughput, showing that our protocol can also work in larger areas.

8. RELATED WORK

A wide range of work has explored techniques for tuning transmit power and/or CCA threshold to improve performance.

Transmit Power Tuning. The authors in [1] propose to increase spatial reuse in dense wireless deployments by reducing transmit power levels. The idea is that this will reduce interference, as suggested by the circle model. Our approach is based on a more realistic model of radio behavior. The authors also observed that minimizing transmit power can lead to starvation.

In [4], the authors measure the performance of power control in wireless networks, and identify three cases: 1) overlapping, where aggregate throughput cannot be increased by power control, 2) hidden-terminal, where power control can help to ensure fairness, 3) potentially disjoint, where power control can allow concurrent transmission. Their results are consistent with our observations. Their work did not consider the interaction with the CCA threshold selection.

CCA Threshold Tuning. Echos [21] tunes the CCA threshold while using fixed power levels. The idea is that sources delay transmission if an ongoing transmission will interfere. In our simulation, we found that Echos leads to starvation in some scenarios and we propose an altruistic variant that avoids starvation at the cost of slightly reduced network throughput.

Joint CCA/Transmit Power Tuning. A few efforts, have explored joint CCA/transmit power tuning to maximize performance. However, existing efforts have assumed all nodes [15] or all nodes in a cell [17] use the same configuration. These approaches work poorly when there is diversity in client-AP distances. Our protocol operates on a per-link basis and can handle such diversity both within a cell and between cells.

Both [17] and [7] tune the CCA by keeping the product of power level and CCA threshold a constant. We found that this approach, which we called alpha, did not work well in certain scenarios where alpha need to be set conservatively to avoid startvation, while wasting many spatial reuse opportunities. Also, in the experiments in [17], the txpower and CCA thresholds are calculated offline. We deployed the protocol in an online fashion, and addressed several practical issues.

Practicality of Power Control. [20] studies the feasibility of fine-grained power control based on RSSI readings. The paper shows that under certain senarios, fine-grained power control is not feasible, and suggests using a more coarse-grained power control in these scenarios. Their work is complimentary to our work and can be added to our protocol. The authors in [5] study the interactions between channel selection, user association, and power control. They observed that channel selection is important for power control to work. Also, since in their experiments, APs are carefully deployed to maximize coverage, power control is not as critical.

9. CONCLUSION

We presented a practical interference-aware power management protocol. In the protocol, each node inserts signal strength information into its packets. This allows nodes to measure the path loss on nearby links by monitoring traffic. Based on this information, nodes then execute a power control algorithm that iteratively increases the number of concurrent transmissions that can take place. We introduce an altruistic version of the Echos CCA tuning algorithm to avoid starvation. We implemented the protocol in Linux and addressed several practical challenges. The experimental results from a 8-node platform shows that our protocol works well in practice. Our evaluation of the protocol using the OPNET simulator shows that it improves network throughput by 22% to 87% compared with only tuning the CCA threshold and using default transmit power, and by 2% to 67% compared with using minimum power

levels. The protocol also improves performance for low-throughput links.

Two areas that we did not explicitly address are the protocol's failure modes and its support for rate adaptation. An example failure mode is that the protocol may react to RSSI fluctuations and create asymmetric links, where one of the sources thinks concurrent transmission is possible and the other thinks it is impossible. We believe it is possible to have nodes monitor achieved and expected throughputs and trigger recovery processes when expectations are not met. To handle data rates, our protocol must be adapted to produce conflict graphs for all considered data rates and initiate data rate changes at a coarser time scale than in current systems. We leave these design issues to future work.

10. REFERENCES

[1] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of the MobiCom*, pages 185–199, Cologne, Germany, aug 2005.

[2] Y. Bejerano, S.-J. Han, and L. E. Li. Fairness and load balancing in wireless lans using association control. In *Proceedings of MobiCom*, pages 315–329. ACM Press, 2004.

[3] G. Brar, D. M. Blough, and P. Santi. Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In *Proceedings of MobiCom*, pages 2–13, 2006.

[4] I. Broustis, J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos. Implications of power control in wireless networks: A quantitative study. In *PAM*, 2007.

[5] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. Mdg: measurement-driven guidelines for 802.11 wlan design. In *MobiCom '07*, pages 254–265. ACM, 2007.

[6] S. M. Das, D. Koutsonikolas, Y. C. Hu, and D. Peroulis. Characterizing multi-way interference in wireless mesh networks. In *ACM MobiCom International Workshop WiNTECH*, 2006.

[7] J. A. Fuemmeler, N. H. Vaidya, and V. V. Veeravalli. Selecting transmit powers and carrier sense thresholds for csma protocols. In *Technical Report, UIUC*, 2004.

[8] S. Ganu, H. Kremo, R. Howard, and I. Seskar. Addressing repeatability in wireless experiments using orbit testbed. In *TRIDENTCOM '05*, pages 153–160, Washington, DC, USA, 2005. IEEE Computer Society.

[9] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. In *IEEE Transactions on Information Theory, vol. IT-46*, pages 388–404, 2000.

[10] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan. Understanding the real-world performance of carrier sense. In *ACM SIGCOMM Workshop E-WIND*, August 2005.

[11] R. A. Jones. netperf: A network performance benchmark. In *Information Networks Division, Hewlett-Packard Co.*, 1993.

[12] G. Judd and P. Steenkiste. Using emulation to understand and improve wireless networks and applications. In *NSDI*, May 2005.

[13] G. Judd, X. Wang, and P. Steenkiste. Low-overhead channel-aware rate adaptation. In *MobiCom '07*, pages 354–357. ACM, 2007.

[14] V. Kawadia and P. R. Kumar. Principles and protocols for power control in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, Vol I, 2005.

[15] T.-S. Kim, J. C. Hou, and H. Lim. Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks. In *Proceedings of MobiCom*, pages 366–377. ACM Press, 2006.

[16] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *WinTECH '07*, pages 19–26. ACM, 2007.

[17] V. Mhatre, K. Papagiannaki, and F. Baccelli. Interference mitigation through power control in high density 802.11 wlans. In *Infocom*, 2007.

[18] 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 *Proceedings of IMC*, 2005.

[19] N. Poojary, S. Krishnamurthy, and S. Dao. Medium access control in a network of ad hoc mobile nodes with heterogeneous power capabilities, 2001.

[20] V. Shrivastava, D. Agrawal, A. Mishra, S. Banerjee, and T. Nadeem. Understanding the limitations of transmit power control for indoor wlans. In *IMC '07*, pages 351–364. ACM, 2007.

[21] A. Vasan, R. Ramjee, and T. Y. C. Woo. Echos - enhanced capacity 802.11 hotspots. In *Infocom*, pages 1562–1572, 2005.

[22] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of the IEEE EmNetS-II*, May 2005.

[23] J. Zhu, B. Metzler, X. Guo, and Y. Liu. Adaptive csma for scalable network capacity in high-density wlan: a hardware prototyping approach. In *Proceedings of Infocom*, 2006.