PIE in the Sky: Online Passive Interference Estimation for Enterprise WLANs

Vivek Shrivastava
Nokia Research Center *

Shravan Rayanchu, Suman Banerjee
University of Wisconsin-Madison

Konstantina Papagiannaki
Intel Labs, Pittsburgh

Abstract

Trends in enterprise WLAN usage and deployment point to the need for tools that can capture interference in real time. A tool for interference estimation can not only enable WLAN managers to improve network performance by dynamically adjusting operating parameters like the channel of operation and transmit power of access points, but also diagnose and potentially proactively fix problems. In this paper, we present the design, implementation, and evaluation of a Passive Interference Estimator (PIE) that can dynamically generate fine-grained interference estimates across an entire WLAN. PIE introduces no measurement traffic, and yet provides an accurate estimate of WLAN interference tracking changes caused by client mobility, dynamic traffic loads, and varying channel conditions. Our experiments conducted on two different testbeds, using both controlled and real traffic patterns, show that PIE is not only able to provide high accuracy but also operate beyond the limitations of prior tools. It helps with performance diagnosis and real-time WLAN optimization, we describe its use in multiple WLAN optimization applications: channel assignment, transmit power control, and data scheduling.

1 Introduction

Radio interference remains a key performance bottleneck for enterprise WLANs [25]. In spite of significant progress in planning, deploying, and managing enterprise WLANs, administrators today have very tools that can help them understand how much interference exists in their network, and how interference patterns evolve over time. Building an on-line tool for enterprise-wide WLAN interference estimation is particularly challenging, because interference is highly dynamic in nature. Each time a new client arrives, departs, moves, or changes its traffic pattern, the number of other nodes in the network it interferes with (and the degree to which it interferes) changes. Further, wireless channel conditions are never static but continuously evolve with changes in the environment, e.g., even with the opening or closing of a door, people walking, etc.

The goal of this paper is to answer the following question: *Given an enterprise WLAN consisting of a number of Access Points (APs) and mobile clients, compute its real-time conflict graph, i.e., identify the precise set of nodes that interfere with each other and the degree to which they do so at any specified point of time.*

Applications of interference estimation: This problem of interference estimation is fundamental to understanding the behavior of any wireless network. Further, interference estimates and the conflict graph serve as important inputs to many WLAN configuration problems, e.g., channel assignment for each AP, transmit power selection, and even emerging strategies for data scheduling across the enterprise WLAN [22].

A number of research efforts have made significant progress toward this tool building goal. Prior techniques for interference estimation mainly employ active probing (interference maps [15] and micro-probing [3]) and suffer from three main problems: a) they incur moderate to significant measurement overhead and cannot be employed to continuously obtain interference information across time, b) they offer limited visibility into the root cause of interference, c) they often require specific client modifications. While some recent work has also explored the potential for passive interference estimation, it is mostly limited to offline trace collection and analysis, and thus cannot be employed in real time.

In this paper, we explore an alternate design for a practical online interference estimation mechanism, one that does not impose any active measurement traffic on the WLAN. It is completely passive in nature, and estimates interference by simply observing ongoing traffic at the different APs. Specifically, we present the design, implementation, and detailed evaluation of a Passive Interference Estimator (PIE) system.

Our work is inspired by two key passive WLAN monitoring approaches proposed earlier: Jigsaw [8, 9] and WIT [13]. These systems provide us with two useful building blocks: (i) a platform for capturing wireless traffic and merging traces collected from different vantage points and (ii) specific tools to infer interesting properties about the 802.11 network from such merged traffic traces. However, both these research efforts stop short of addressing our goal of designing a real-time interference estimation tool. The key features of PIE are:

1. It captures dynamic interference information quickly and robustly: PIE captures interference information across the entire WLAN within a few hundred milliseconds. It can effectively identify the real interfer-

*Vivek Shrivastava worked on this project as a PhD student at UW-Madison

ers when multiple overlapping transmitters are present.

2. It uses real traffic patterns: PIE is passive; it estimates interference using actual traffic patterns in the network, capturing the effects of bit rate adaptation, varying packet sizes, and traffic burstiness.

3. It has low overhead and causes no downtime: Being passive, PIE does not take away wireless bandwidth resources from users.

4. It does not require client modifications: The PIE mechanism is implemented at the APs and a central controller placed within the enterprise wired network. No client modifications are required.

PIE relies on the accurate timestamping of transmissions by the AP. These timestamps could be reported accurately by the firmware of the AP's wireless card. However, most off-the-shelf wireless cards do not expose this functionality and hence in our current implementation we use a second card at the AP to gather accurate timestamps of wireless transmissions.

Key contributions

This paper makes the following key contributions:

• We identify the key requirements for a practical interference estimation mechanism. We then carefully design PIE to meet those requirements and report various design choices to infer interference in real time.

• We evaluate the accuracy and agility of PIE using both controlled experiments as well as by playing back real traffic traces. For 95% of the links, PIE achieves accuracy comparable to the state-of-the-art technique of bandwidth tests (see §2). We further show that PIE can efficiently track the changing interference patterns caused by client mobility, variable transmission rates and varying traffic loads. Results from our playback of real traces indicate that PIE can converge to the correct interference estimate within 540 ms, 700 ms and 900 ms for heavy, medium and low traffic load periods. This represents up to 300× of speed up over bandwidth tests.

• *Demonstrate the utility of PIE in interference mitigation mechanisms:* We show the usefulness of PIE by integrating it with three interference mitigation mechanisms 1) Centralized scheduling, 2) Transmit power control and 3) Channel assignment. We show that real-time conflict information provided by PIE can enhance the performance of such mechanisms and outperform bandwidth tests under dynamic settings.

• *Employ PIE to uncover performance issues in two production WLANs:* We use PIE to monitor two production WLANs. We show that PIE can correctly infer subtle performance issues like asymmetric channel access and hidden terminal problems.

The rest of the paper is organized as follows. §2 discusses the current state of art in wireless interference estimation. The fundamental principles behind PIE are described in § 3. We present the design and operation of PIE in §4. We evaluate and validate our mechanism in §5. Finally we conclude in §6.

	Interference maps [15]	Microprobing [3]	CMAP [24]	PIE
No client mods	×	√	×	√
Online	×	√	√	√
Zero downtime	√	×	√	√
Real traffic	×	×	√	√
No wireless control traffic	×	×	×	√

Table 1: Comparing PIE with other interference estimation mechanisms.

2 Related work

We classify prior interference estimation and wireless monitoring efforts into the following categories.

Interference estimation tools : Bandwidth test mechanisms [16, 15] systematically transmit a simultaneous burst of traffic along each pair of AP-client links and observe how the aggregate throughput differs from the throughput achieved by each link operating in isolation. Recently, Ahmed et al. [3, 4] proposed the use of micro-experiments, each lasting less than a millisecond, to detect different kinds of conflict between WLAN nodes. Such mechanisms require network downtime and must rely on certain traffic pattern to test the interfering links, which may be deviant from real traffic scenarios.

CMAP [24] is a technique designed to solve exposed terminal problem using passive conflict graphs. However, it requires the interferers to be in the communication range of the receiver and will miss conflicts in which the interferer is outside the communication range but inside the interference range. Further, it requires driver level modifications to both APs and clients. Given that CMAP relies on modified clients, it is better able to infer uplink conflicts as well. However, since the fraction of uplink traffic might be limited (as reported for some enterprise WLANs [22]), we take the penalty of missing some uplink conflicts in order to avoid client modifications. Table 1 presents a comparison of our design of PIE with some prior proposed interference estimation tools.

Wireless monitoring studies: Researchers have recently conducted several studies to understand the performance of different 802.11 networks using trace collection, followed by empirical analysis. Each system is designed to analyze specific aspects of an 802.11 wireless network, ranging from physical and link-level behavior [21, 2, 24], client coverage [7], to understanding the performance of TCP/IP in wireless environments [9]. However, most of these mechanisms are geared towards offline analysis of wireless traces to derive interesting measures for their target 802.11 network. Recently, a short paper [6] proposed a machine learning approach to infer high-level interference. However, the proposed technique provides limited visibility and does not capture all types of interference.

Finally, WIT [13] and Jigsaw [8] are interesting measurement studies that have influenced some of the design decisions in PIE . In WIT, traces are captured using 5 sniffers in a wireless network and a state machine based learning approach is proposed to study the performance of the 802.11 MAC protocol in a practical deployment. Jigsaw deploys a large wireless monitoring infrastructure consisting of 150 sniffers to monitor a production WLAN and performs a cross-layer analysis to diagnose performance problems. Both these mechanisms present excellent insights into the functioning of a 802.11 network, but unlike PIE, they do not focus on evaluating the accuracy and agility of their interference estimation mechanisms, especially under interference settings that can arise due to client mobility and the use of bit rate adaption mechanisms. Also, they do not discuss the integration of their interference estimation mechanisms with applications like power control and channel assignment.

3 Interference estimation in PIE

Interference in an enterprise WLAN can be broadly classified into two categories: (a) sender-side interference caused by carrier sensing between two transmitters, and (b) receiver-side interference caused by collision at the receiver. While carrier sensing determines how the transmitters share the wireless medium, collision-induced interference determines whether transmissions are successfully decoded at the intended receiver. The goal of PIE is to identify both of these interference properties in a non-intrusive manner. We now explain the intuition behind PIE with the help of a simple example.

Intuition behind PIE: Consider a scenario from an enterprise WLAN (shown in Figure 1) where APs A and B are far enough apart such that they cannot carrier sense (CS) each other. Assume that two clients C_A and C_B are associated to APs A and B respectively. Suppose some downlink packets are being enqueued and being transmitted by APs A and B, for transmission to their respective clients, C_A and C_B. The APs follow the regular 802.11 carrier sensing mechanism, and transmit to their clients whenever possible.

In PIE, APs A and B periodically send their frame transmission timestamps to the controller. Further, the frames are tagged with their reception status indicating whether this frame transmission was successful or not (i.e., whether the AP has received an ACK for this frame or not). The controller parses these timestamps and identifies the four scenarios shown in Figure 1(b). Looking at scenarios 1 and 2, the controller observes that frame transmissions from A and B (denoted by P_A and P_B) overlap in both directions, indicating that A and B do not defer to each other, and hence are not within carrier sense range. Additionally, the controller can also infer that whenever a transmission for client C_B overlaps with a transmission by AP A, then C_B is not able to decode the transmission (i.e., P_B is lost). On the other hand, transmissions for C_A are not lost despite overlapping transmissions by AP B. Hence the controller concludes that AP A interferes with link (B, C_B) but B does not interfere with (A, C_A). The controller can then use this information to efficiently mitigate interference for C_B. For example, it can perform downlink data scheduling [22] and allocate different time slots to (A, C_A) and (B, C_B). Alternatively, the controller can also assign different channels to APs A and B, thereby allowing both transmissions to proceed simultaneously without any interference. As this example demonstrates, having accurate interference estimates could enable the controller to improve client performance in an enterprise WLAN by employing interference mitigation mechanisms effectively. We now give a detailed explanation of how PIE identifies these interference properties in a non-intrusive manner.

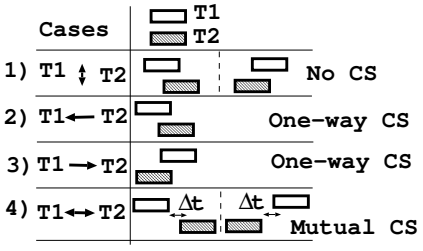

Figure 2: Detecting the carrier sense relationship between two links on the basis of timestamps of transmissions by the two transmitters A and B. Timestamps refer to the MAC timestamp of wireless frames as reported by the wireless card.

3.1 Estimating carrier sense (CS) interference

PIE identifies the carrier sense relationships based on the order in which competing transmitters access the wireless channel. Figure 2 shows the possible order of channel access for different carrier sensing relationships. As shown, there can be four cases of channel access:

(a) Overlapping frame transmissions (Cases 1, 2 and 3): Case 1) When two competing transmitters are not in carrier sensing range, they can access the channel in any order and hence the controller would observe that their frames overlap in both directions. Case 2,3) In case of one-way carrier sensing, the frames will only overlap in one direction. For example, if $T_1 \leftarrow T_2$ (i.e., T_1 carrier senses T_2) then T_1 will defer for T_2's transmissions. However, T_2 will not defer for T_1's transmissions, and would transmit even if T_1's frame is still in air. Hence the controller should only observe overlaps when T_1's transmission is already in the air and is overlapped by a later T_2 transmission.

(b) Non overlapping transmissions (Case 4): If both the transmitters can mutually carrier sense each other,

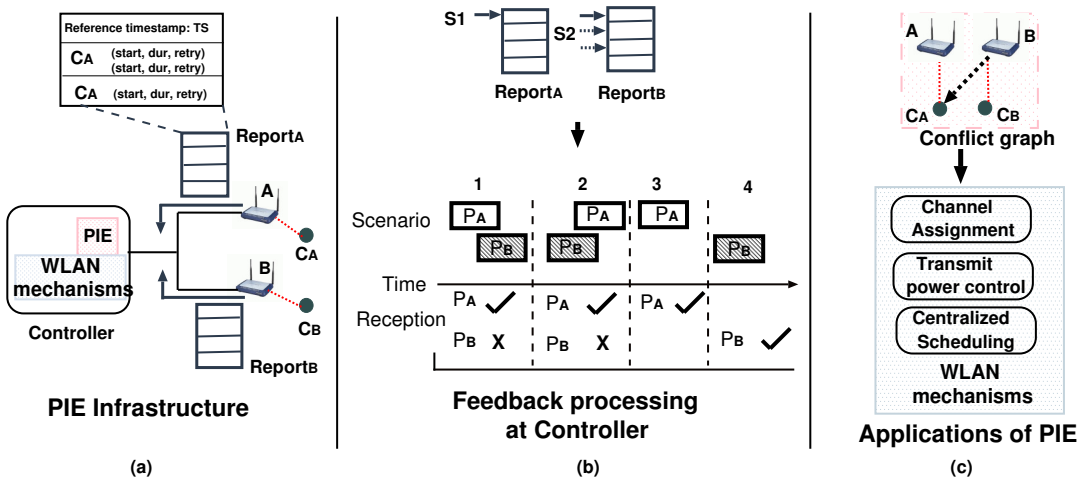

Figure 1: Overview of PIE, showing the overall infrastructure, the feedback processing performed at the Controller and the integration of PIE with channel assignment and scheduling. The detection of conflict between AP B and client C2A i) places the two APs in separate channels when channel assignment is performed, or ii) serializes the transmissions between AP A and B.

the controller should not see any overlaps as carrier sensing will serialize their frame transmissions. However, we note that non-overlapping transmissions may also be observed in scenarios where the two transmitters do not simultaneously contend for the channel, and transmit their frames one after another due to their specific traffic patterns. In such a scenario, it is difficult to make any inference regarding carrier sense relationship of the two transmitters. In order to distinguish the cases where transmitters are actually contending for the medium, we use the mechanism outlined in [13]. The controller labels a pair of frames as being transmitted by c̈ontending" transmitters if their starting timestamps are within a time interval γ, where γ is the total time that can be spent by competing transmitters performing back-off. Although all traffic within the γ interval may not contend for the channel, this heuristic was shown to be effective for practical settings [13]. We use a value of $\gamma = 28 + 320\mu s$ (DIFS + Max back-off period for 802.11g). The pseudo-code for estimating carrier sense properties in PIE is shown in Algorithm 1 (Procedure ComputeCS).

3.2 Estimating collision induced interference

PIE identifies collision-induced interference at the receiver by computing the probability of a frame loss at the receiver when it overlaps with a simultaneous transmission from a competing transmitter. Intuitively, the extent of interference is directly proportional to the probability of losing overlapping frames. Note that this allows PIE to maintain a continuous interference model, where the extent of interference can be any value between 0 and 1. Such a model is better suited for realistic environments where the binary model of interference may not suffice. On the basis of this observation, in PIE, we use the Link

Algorithm 1 PIE : CS and INT computation

Procedure ComputeCS:
Inputs: number of frames in contention n_c, number of case (3) overlaps n_f, and number of case (2) overlaps n_r, cs threshold δ_t (δ_t = 0.8 in our implementation)
$n_o = n_f + n_r$
$n_n = n_c - n_o$
if ($\frac{n_n}{n_c} > \delta_t$) **then**
  /* case 4 (A and B sense each other) */
  return $\frac{n_n}{n_c}$
else if ($\frac{n_o}{n_c} > \delta_t$) **then** /* sufficient overlaps to compute prob */
  if ($\frac{n_r}{n_c} > \delta_t$) **then**
    /* cases 3 (A senses B) */
    return $\frac{n_r}{n_c}$
  else
    /* case 1 (A and B do not sense each other) */
    return $\frac{n_n}{n_c}$
else
  /* inconclusive (wait for more samples) */
  return $-$
Procedure ComputeINT:
Inputs: total number of frames n_p, number of frames lost n_l, number of overlapping frames n_o, number of overlapping frames lost n_{ol}, overlapping packets threshold β_t (β_t = 20 in our implementation)
if ($n_o > \beta_t$) **then**
  $l_{iso} = (n_l - n_{ol})/(n_p - n_o)$ /*loss in isolation*/
  $l_{int} = n_{ol}/n_o$ /*loss under interference */
  LIR = $(1 - l_{int})/(1 - l_{iso})$
  return LIR
else
  /* inconclusive (wait for more samples) */
  return $(-)$

Interference Ratio (LIR) described below, as the metric to quantify interference for a link.

Link Interference Ratio (LIR): For a pair of interfering links, LIR captures the loss in performance observed when the two links are interfering, as opposed to operating in isolation. Consider a link (A, B) and its interferer C. We measure D_{AB}, the delivery probability of the link (A, B) in isolation (A is active, C is inactive). We then measure D^C_{AB}, the delivery probability of the link when

interferer C is also active with A. The LIR is given by:

$$LIR = D_{AB}^{C}/D_{AB} \quad (1)$$

LIR takes values between 0 and 1. LIR of 0 means that link (A, B) cannot deliver frames in the presence of C, while LIR of 1 means that C does not impact link (A, B). LIR values between 0 and 1 indicate the extent of interference on link (A, B) by interferer C. When A and C are in carrier sense range, LIR will be equal to 1, since the interferer C is able to share the channel with the transmitter A without causing any decrease in the delivery ratio of link (A, B) [1]. The pseudo-code for estimating interference is shown in Algorithm 1 (Procedure ComputeINT). PIE requires a certain threshold of overlap packets (β_t) to accurately estimate the loss rate under interference. We use $\beta_t = 40$ for our implementation as it is the smallest threshold that yields stable interference estimates under diverse experimental scenarios.

Handling simultaneous overlaps from multiple interferers: A client packet may overlap with multiple simultaneous transmissions from potential interferers. In such a scenario, the packet overlap and its subsequent loss or success is attributed to each overlapping interferer. Further transmission diversity will allow PIE to observe events that will distinguish the true interferer from the nodes that happened to transmit at the same time (future overlapping transmissions by false interferers will not lead to loss). As we show later in our evaluation in §5.1.3, there is significant diversity in wireless transmissions in realistic settings to allow PIE to operate efficiently in practice.

4 PIE Design and Operation

In this section, we describe the design and operation of PIE. A schematic overview of the overall design can be seen in Figure 1. PIE has the following three components.

Sniffing at the APs: In our current implementation of PIE sniffing of the wireless medium is limited to the APs in the enterprise WLAN. This allows us to avoid the additional overhead associated with the deployment and management of extra sniffers in the enterprise building. However, sniffing solely at the APs might result in reduced coverage of uplink client traffic, as compared to a dense sniffer deployment (e.g., as in Jigsaw [8]). In order to overcome this limitation, we employ the finite state mechanisms outlined in [13] (based on 802.11 states) to infer some of the missing client transmissions. We note that even with such mechanisms, it is difficult to capture all uplink client transmissions using monitors at the AP, and hence PIE may not be able to detect all uplink conflicts accurately. However, we accept this penalty of missing some uplink client conflicts in order to avoid deploying additional monitors.

PIE requires accurate timestamp information for accurate interference estimation. However, due to limitations of the existing Atheros driver and firmware, it is difficult to extract the exact time at which a packet is transmitted over the medium[2]. In order to overcome this problem, in our implementation of PIE, APs are equipped with two radios: one radio is used for normal packet transmissions and receptions, while the other radio is used for capturing packets on the wireless medium. The Atheros driver timestamps every frame that is received over the interface using an on-board 64-bit microsecond resolution timer. Thus a second radio that captures packets can record the exact timestamp of the packet transmission. Moreover, the proximity of the two radios ensures that the second radio receives the majority of frames transmitted by the AP due to capture effect.

Synchronization of clocks at the APs: PIE needs the APs to synchronize their clocks so that the controller can compare their packet transmission reports and determine the extent of overlap between any two transmissions reported by the APs. Further, time synchronization should be tight to allow accurate 802.11 analysis, on the order of 20-30 μs [8]. Prior mechanisms for 802.11 analysis [8, 9, 13, 26] synchronized the APs by finding common beacon packets in their transmission reports. However, performing such offline synchronization at the controller can be time consuming, and impractical for a real time interference estimation mechanism. To synchronize the clocks across the APs, we use the time synchronization protocol implemented by the Atheros driver [1]. As part of the protocol, the AP embeds a 64-bit microsecond granularity time stamp in every beacon frame, and the nodes that listen to the AP adjust their local clock based on this broadcast timestamp [12]. In order to make this synchronization seamless, we set up a virtual ad hoc interface on the second radio of each AP. Now all the APs that join the ad hoc network, synchronize themselves in real time using the beacons of the reference AP for the network. This approach has two key benefits:1) it is an online mechanism, meaning the nodes synchronize their clocks every time beacons are received from neighboring nodes, and, 2) it is transitive in nature, and works as long as the network is not partitioned.

[1] Note that this measure of LIR differs slightly from the interference metric proposed in [16], that relies on effective throughput and not delivery probability. However, throughput based LIR is ambiguous for carrier sensing scenarios, where a LIR value of 0.5 could mean 50% loss or carrier sensing. Hence we use delivery probability as it provides greater clarity into the LIR values in all scenarios.

[2] This is because once the driver passes the packet to the firmware, a variable delay is introduced based on the length of the firmware transmit queue and the amount of time the radio performs carrier sensing/back-off. Further, retry and other 802.11 packets (like beacons) are handled solely by the firmware, making timestamp estimation more challenging.

Section	Objective	Topology	Observation
§ 5.1.1	Accuracy of PIE for canonical topology	2-link (Hidden / Exposed / Normal) topology	PIE is accurate within ± 0.1 of ground truth for 95% of scenarios
§ 5.1.2	Accuracy of PIE under client mobility, variable bit rates and packet sizes	2-link (Hidden) topology	PIE is able to track the changing interference patterns in real-time ($\sim$ 100 ms)
§ 5.1.3	Evaluate accuracy with multiple simultaneous transmitters	15-node topology	PIE is accurate when transmitters overlap less than 75% of time
§ 5.2.2,5.3	Convergence time of PIE under real trace-based traffic replay	15-node topology	Median convergence time is 400, 600, 720 ms for heavy, medium and light client traffic
§ 5.4.1,5.4.2 § 5.4.3	Performance of channel assignment, power control & scheduling with PIE	15-node topology	Outperforms bandwidth tests in dynamic cases (1.25×, 1.50× gain in goodput, fairness)
§ 5.4.4	Performance diagnostics in two production WLANs	386 & 464 AP-client links	8-11% links suffer from hidden terminals and 20% links show rate anomaly problems

Table 2: Summary of evaluation results.

Collecting and processing feedback from the APs: In PIE the Controller periodically polls the APs for their transmission reports. The granularity of polling is a tunable parameter, which can be determined empirically. Lower polling periods will enable PIE to update interference estimates faster. On the other hand, increasing the polling period allows APs to sample more packets per transmission report, increasing the accuracy of interference estimates. We evaluate this tradeoff in §5 and show that a polling period of at least $\sim$100 ms is needed to achieve good accuracy for PIE . Feedback processing at the Controller takes O(m^2n) time, where m is the number of APs and n is the number of packets per AP[3].

Handling multi-rate links: The exact impact of an interferer on a transmitter-receiver pair also depends on the physical layer bit rate being used by the transmitter. PIE tags the LIR value for each link-interferer pair with the bit rate being used for packet transmission on the wireless link. During the computation of LIR values as described in §3.2, overlap and isolation losses are recorded separately for each physical layer data rate and then the corresponding LIR value is computed for each rate. The Controller maintains a two-level lookup table for LIR values, where the first level is indexed by the link-interferer pair and the second level provides values for different rates used by the link for the given interferer. This data structure can also be extended for tagging conflicts with the transmit power level of the interferer, allowing the Controller to estimate the level of conflict under different power levels.

Interaction with external interference: External interference can be caused by non-enterprise wireless traffic and/or non-WiFi traffic (like microwaves). In the first case, if the non-enterprise traffic source is visible to any enterprise AP, its transmission timestamps would be reported to the PIE controller, which could then use the normal procedure to detect if the external source is causing any problems for the enterprise clients. In the second case, when the external interferer is not visible (like a non-WiFi source or a hidden external WiFi source) to any enterprise AP, PIE would not be able to identify the source of interference.

[3]Since the transmission report by each AP is already sorted, the overhead of merging at the Controller is small.

5 Evaluation of PIE

We divide the evaluation section into three distinct subsections. First, we demonstrate that PIE accurately captures interference in real time. We do so by comparing PIE with bandwidth tests. Next, we measure the time taken by PIE to converge to accurate interference estimates, under both controlled traffic loads and realistic trace replay on the wireless testbed. Lastly, we integrate PIE with a number of real time WLAN optimization mechanisms to offer evidence that PIE is useful for real-time problem diagnosis on a WLAN.

We evaluate PIE on two different testbeds. We run our central controller on a standard Linux PC (3.33 GHz dual core Pentium IV, 2 GB DRAM) (in about 3,000 lines of C code and a few hundred lines of Perl script), and Soekris (Testbed 1) as well as VIA-based (Testbed 2) wireless APs, modified slightly to improve path latencies. Each node in the two testbeds is equipped with two Atheros AR5212 chipset wireless NICs. We use saturated UDP traffic for our experiments unless otherwise specified.

Summary: A summary of the results presented in this section is shown in Table 2. Our results show that (i) PIE accurately estimates LIR under different carrier sensing and interference relationships, (ii) PIE can handle client mobility, variable bit rates and packet sizes, (iii) PIE is able to distinguish between multiple interferers when overlap in transmissions is less than 75%, (iv) PIE converges within 100 ms for saturated traffic, and within 400 ms, 600 ms and 720 ms when heavy, medium and light activity traffic periods are replayed from a real trace, (v) PIE enables WLAN applications to perform efficiently in dynamic scenarios, (vi) PIE can identify performance problems in hidden terminals and rate anomaly in production WLANs.

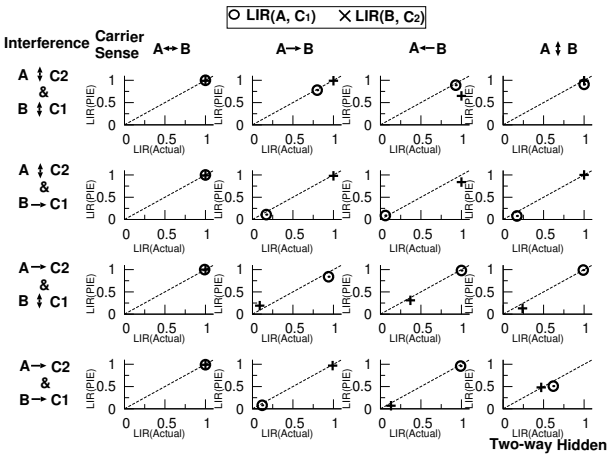

Figure 3: Scatter plots comparing the LIR values of PIE with the ground truth computed using unicast bandwidth test for all possible combinations for carrier sensing and interference relationships that can occur in a two link canonical topology. Packet size and data rate was fixed at 1400 bytes and 6M respectively. Note that for all scenarios, the value computed by PIE is close to the value reported by bandwidth test, as indicated by the proximity of these values to the x=y line in the plots.

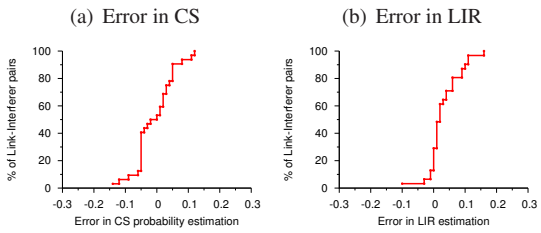

Figure 4: Distribution of error in predicting (a) Carrier Sense probability, and (b) LIR value as compared to the ground truth computed using unicast bandwidth tests, for the sixteen canonical scenarios outlined in Figure 3.

5.1 Accuracy of PIE

We evaluate PIE's accuracy using two different methods. First, we construct all possible conflict scenarios using a canonical two link topology. This experiment serves as our controlled experiment that allows us to assess accuracy and focus on the underlying phenomena causing any discrepancies between PIE and bandwidth tests. Second, we generalize our findings across a large-scale testbed, quantifying PIE's overall accuracy. Overall accuracy is further evaluated across a number of dimensions that take into account diverse transmission rates, packets sizes, interference scenarios, and density.

Metrics for comparison: Both experiments are evaluated according to the Link Interference Ratio (LIR) described in §3.2. LIR is the ratio of the frame delivery probability [4] of a link (A, B) under interference from C and in isolation (D^C_{AB}/D_{AB}).

Compared schemes: We compare three approaches that measure LIR with differing levels of overhead.

1) Unicast bandwidth tests (Ground truth): The conventional approach, is to use unicast bandwidth tests (UBT) to determine the impact of an interferer on a link [16]. In unicast bandwidth tests, A transmits unicast packets to B in isolation and under interference from C. We then report LIR as the ratio of frame delivery probabilities under the two scenarios. This is an accurate test to determine LIR as it uses unicast traffic, which takes into account the impact of C on the receiver (data packet collisions) and the sender (ack collisions). Henceforth, we use the LIR value reported by unicast bandwidth tests as the g̈round truth" in our experiments. Note that UBT incurs significant overhead – it takes O(n^4) measurements to compute a conflict graph for a n node topology, and hence is not practical to use under dynamic wireless environments.

2) Broadcast bandwidth tests : In broadcast bandwidth tests (BBT), broadcast traffic from A to B is used to compute the frame delivery ratios, both in isolation and under interference from C. This method was proposed as a relatively fast way to measure interference relationships among a large number of links [16]. Broadcast tests can compute the conflict graph for a topology of n nodes using O(n^2) measurements (as opposed to O(n^4) for UBT). However, broadcast tests do not take data-ack collisions into account and hence may be inaccurate in some scenarios.

3) PIE : PIE computes the LIR value in a passive fashion by determining the conditional loss probability of packets on link (A, B) that are interfered by interferer C. A packet P_i on link (A, B) is considered to be interfered if it overlaps with a transmission from interferer C that leads to packet loss. The LIR in this case is computed by passively observing the events in the wireless medium as recorded at the controller. Psuedocode for PIE is shown in Algorithm 1 (function ComputeINT).

In what follows, all experiments are performed using 802.11a (except the live WLAN measurements in §5.4.4), to prevent interference from the co-located department WLAN that uses 802.11g. Furthermore, the PIE measurements are collected passively through the observation of the probe traffic generated by the bandwidth tests.

5.1.1 Static interference settings

We start by comparing the LIR generated by the three mechanisms for different canonical scenarios, as shown in Figure 3. In order to have a fair comparison, we first evaluate the accuracy of PIE under static data rate (6Mbps) and packet size (1400 bytes) settings, as the overhead for computing LIR for dynamic (client mobility, variable rates) can be significant for bandwidth tests. We then relax these constraints and evaluate the performance of PIE under dynamic interference scenarios triggered by client mobility, the use of variable transmission rates and different packet sizes.

[4] 802.11 ACK is included into frame delivery rate for unicast frames

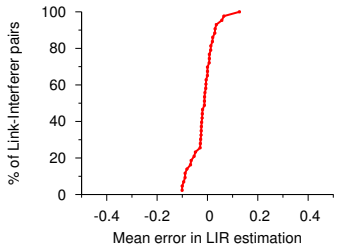

Figure 5: Distribution of error for PIE as compared to LIR values computed using UBT. We note that in 95% of the interference scenarios PIE is within 0.1 of the actual LIR value.

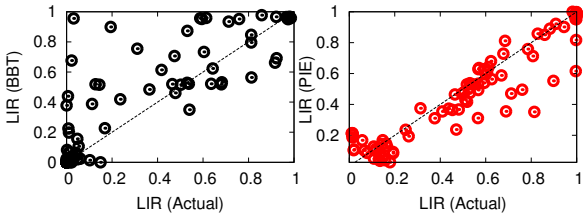

Figure 6: Scatter plot of delivery ratios obtained using bandwidth tests (unicast - LIR(Actual), broadcast - LIR(BBT)) and PIE on 43 link pairs. Note that LIR(BBT) may underestimate the loss rates as it does not take the ACK loss into account.

Controlled experiments: Using a canonical two link topology we benchmark different carrier sensing and interference scenarios. We selectively disable the carrier sensing of transmitters to create the complete set of scenarios. The possible interference relationship between the two links assuming that C_1 is associated with AP A, and C_2 is associated with AP B are as follows: (i) A interferes with C_2 and B interferes with C_1 ($A \rightarrow C_2 \wedge B \rightarrow C_1$), (ii) A interferes with C_2, B does not with C_1 ($A \rightarrow C_2 \wedge B \updownarrow C_1$), (iii) B interferes with C_1, A does not with C_2 ($A \updownarrow C_2 \wedge B \rightarrow C_1$), and (iv) A, and B do not interfere with each others client ($A \updownarrow C_2 \wedge B \updownarrow C_1$). Further, the possible carrier sensing relationship between the two transmitters are: (i) A and B carrier sense each other ($A \leftrightarrow B$), (ii) B carrier senses A ($A \rightarrow B$), (iii) A carrier senses B ($A \leftarrow B$), and (iv) A and B are not in carrier sensing range ($A \updownarrow B$).

Figure 3 compares the LIR values computed by PIE and unicast bandwidth test for the sixteen possible scenarios of carrier sensing and interference between two links. It also identifies cases which correspond to mutual (two-way) and asymmetric (one-way) hidden terminals. As shown in the figure, the LIR estimates of PIE are very close to the values reported by the unicast bandwidth tests. Also, Figure 4 shows the distribution of error in estimating carrier sense probability and LIR values for these different scenarios. As clear from the figure, PIE is able to estimate the carrier sensing and LIR values with good accuracy (± 0.15) for all scenarios. Note that identifying both carrier sensing and LIR values accurately can characterize client performance under any scenario. For instance, in the scenario where the interference relationship is $A \rightarrow C_2 \wedge B \rightarrow C_1$, the links can achieve similar throughputs when they are carrier sensing and sharing the channel ($A \leftrightarrow B$) or when they are not carrier sensing (two-way hidden terminal) and there is close to 40% loss rate for the links. PIE can provide this greater visibility, as to which phenomenon is actually taking place, which can then be used by interference mitigation mechanisms.

Accuracy in larger testbed: We repeat the experiments reported in Figure 3 for a large number of link pairs in our testbed, comprising 30 nodes spread across five floors of our department building. We select links whose delivery ratio in isolation is greater than 0.9 in both directions [3][5]. Figure 5 compares the values of LIR achieved using unicast bandwidth test and PIE for 43 interference scenarios. We note that for 95% of the interference scenarios, PIE is within 0.1 of the actual LIR value. We experimented with different convergence thresholds and found that convergence within 0.1 of the actual LIR value was sufficient for practical applications (see §5.4 for performance of such applications).

Finally, we note some inaccuracies that are introduced through approaches like BBT, which aim to collect interference information at low overhead. BBT will mis-estimate when interference impacts the reception of ACKs rather than data packets. Figure 6 does indeed confirm that such cases do exist in reality and that they lead to the underestimation of loss.

5.1.2 Dynamic interference settings

The previous experiments quantified PIE's accuracy as compared to the ground truth generated using unicast bandwidth tests. However, PIE is not only able to accurately capture interference under static conditions, but more importantly, also under dynamic conditions.

Handling client mobility: Any practical interference estimation mechanism must be able to handle client mobility, i.e. it should be able to update the conflict graph in real time to reflect the changing interference patterns that arise due to client movement. In order to evaluate PIE 's ability to handle mobile clients, we perform a micro experiment, where a mobile client is moving away from its AP towards a hidden interferer as shown in Figure 7. In this experiment, the client is moving at a pace of 0.25 m/s [6]. The bottom plot in the Figure shows the signal strength at the client from the AP and the interferer, while the middle and top plots show the throughput of the mobile client and the LIR estimate by PIE at each instant in the experiment. As shown in the Figure, PIE's LIR estimate decreases as the client moves towards the interferer. Furthermore, it closely matches the trend shown

[5] We wanted to consider stable links (high SNR) for analysis. In reality, poor SNR links would rarely be selected during client association to APs.

[6] Normal walking speed for mobile user.

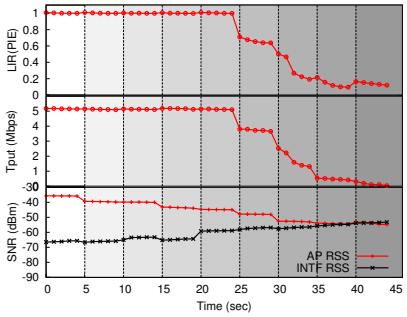

Figure 7: PIE 's ability to track the changing interference patterns for a mobile client. In this experiment, a mobile client is moving away from its AP towards a hidden interferer. The bottom plot shows the signal strength at the client from the AP and the interferer. The middle plot shows the throughput achieved by the client at each instant. The top plot shows the LIR as measured by PIE.

by the instantaneous throughput during the experiment, which confirms PIE's accuracy in predicting the end user performance in dynamic wireless environments.

Variable rate and packet sizes Prior research [24, 5] has shown that the interference properties of wireless links are impacted by the data transmission rate and packet size. In order to evaluate PIE for different packet sizes and data rates, we repeat our canonical experiments with different packet sizes and data rates on multiple interferer-link pairs. To evaluate multiple data rates, we first activate a link in isolation and then activate an interferer, which forces the transmitter to adjust its data rate to minimize losses. We use the default Atheros rate adaptation algorithm, SampleRate. Figure 8 (left) shows the impact of data rate on the delivery ratio of a link (LIR by UBT) and the estimate of LIR generated by PIE for each rate in the experiment.

Next, we fix the data rate and vary the packet size for a link under interference (right plot). As expected, LIR is worse for larger packet sizes, which are prone to more errors. We observe that the combination of data rate and packet size can result in varying interference properties and PIE is able to efficiently identify the impact of interference accurately in each such scenario (confirmed by the agreement with UBT). This also shows that using bandwidth tests or other active measurements may require performing an exponential number of tests with varying packet sizes and rates to determine the interference impact for any given traffic scenario. PIE, on the other hand, can passively determine the extent of interference for each scenario efficiently and accurately.

5.1.3 Classifying interferers accurately

PIE's fundamental operation relies on observing overlap in transmissions and correlating such events with packet loss. One could argue that PIE's accuracy is likely to be affected by scale since the probability of observing overlap in transmissions across the network increases with greater scale. Then the probability of identifying the

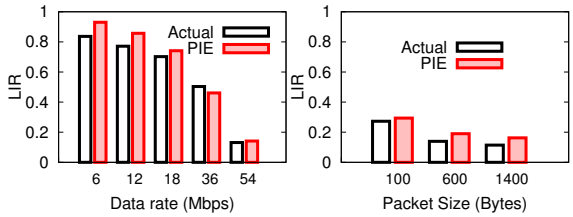

Figure 8: Impact of physical layer data rate and packet size on the delivery ratio of a link in a canonical hidden terminal topology. While varying data rate, packet size is fixed at 1400 bytes, and while varying packet size, data rate is fixed at 24Mbps. Note the significant drop in delivery ratio with rate while the impact of packet size is less pronounced. 90% Confidence intervals were found to be tight and hence are omitted for clarity.

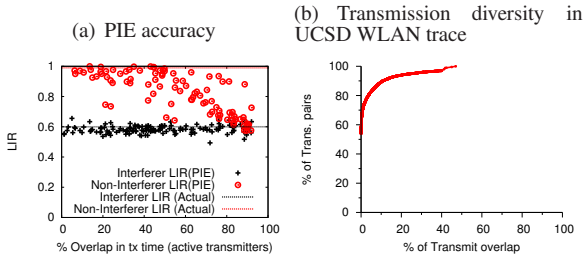

Figure 9: Ability of PIE to identify true interferers from a set of active transmitters. (a) LIR measured by PIE for both the true interferer and the non-interfering transmitter as a function of the overlap in transmission times. If the overlap fraction is less than 75%, PIE can distinguish the false and true interferers accurately. (b) Overlap in transmission times for all wireless transmitter pairs that are active during a one hour time window (2pm - 3pm) in the USCD wireless trace. As clear from the trace, about 90% of the transmitter pairs overlap less than 20% of the times, providing sufficient traffic diversity for PIE.

transmitter responsible for loss becomes much harder. To answer this question we attempt to quantify the success of PIE in correctly identifying an interferer depending on the amount of time that it tends to overlap with the transmitter suffering the loss.

Canonical experiments: Consider a link (A, B) and two interferers C_1 and C_2. We compute the actual LIR of the link under C_1 and C_2 by performing individual unicast bandwidth tests, first with C_1 and then with C_2. According to the unicast tests, the LIR of the link under interference from C_1 and C_2 is 0.6 and 0.99 respectively, indicating substantial interference from C_1 and no interference from C_2. We term C_1 as the interfering transmitter and C_2 as the non-interfering transmitter. Our goal is to evaluate the accuracy of PIE in identifying the interfering (C_1) and non-interfering (C_2) transmitters, when both C_1 and C_2 are activated simultaneously. Both C_1 and C_2 follow a http traffic model, with sleep and active times being drawn from a 802.11 wireless trace [13]. We then identify the time periods (1s) in the experiment with varying overlaps between the transmission times of C_1 and C_2 and measure the LIR values for C_1 and C_2 according to PIE.

Figure 9(a) shows the LIR obtained by PIE for both the interfering (C_1) and non-interfering (C_2) transmitter as a function of the overlap in their wireless transmission times. As expected, when the overlap in transmission times is close to 100%, PIE is unable to distinguish between true and false interferers. When the overlap is less than 60% PIE can distinguish between the false and true interferer. In fact, notice that even for high overlaps (close to 75%), the median loss probability for false interferer is close to 0. Further, as shown in Figure 9(b) more than 90% of the transmitters in a real WLAN trace (UCSD WLAN [8]) overlap less than 20% of the time, indicating rich diversity in transmission patterns for wireless users. Such diversity will enable PIE to function efficiently in realistic deployments.

Multiple interferer experiments: To validate the previous result with multiple interferers, we repeat the aforementioned experiments in a larger topology. In our experiments, we try to emulate the structure of our in-building WLAN by placing one testbed AP node near each production AP in the environment. We present results from a representative topology that randomly distributes client nodes into offices. The topology has 7 APs and 8 clients. Clients connect to the AP with the strongest signal strength. Each transmitter follows a http on-off model for transmitting data with the on and off times derived from the UCSD trace. We classify all interferers for which the UBT LIR is less than 0.8 ($> 20\%$ loss) as strong (interfering) transmitters and the rest are classified as weak (non-interfering) transmitters.

Figure 10 (a) shows the number of strong and weak interferers per client as determined by UBT in our topology. Figure 10 (b) shows the ability of PIE to identify multiple strong and weak interferers in this topology. As shown in the Figure, the LIR values estimated by PIE are within +/- 0.15 of the actual LIR determined by pairwise bandwidth tests using unicast traffic (UBT). Summarizing, PIE is able to accurately identify the exact impact of each interferer on every client in the system even in the presence of multiple simultaneous transmitters. We show the overall impact of such an accurate conflict graph on application level performance for wireless clients in the system in §5.4.

5.2 Agility of PIE

PIE can be integrated in today's centralized WLANs, requiring software-only modifications to the central controller. However, as is apparent from the design section, there are a number of knobs in PIE 's design that are likely to affect its accuracy. In this section, we study appropriate values for the polling interval, and measure PIE's convergence time under varying loads.

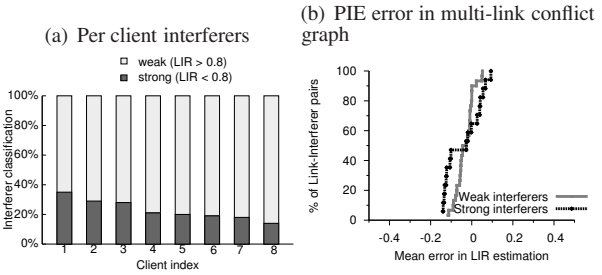

Figure 10: Accuracy of PIE for a 8 client, 7 AP topology. (a) Distribution of strong (LIR < 0.8) and weak (LIR > 0.8) interferers. (b) CDF shows the error in PIEś estimation of LIR for a link-interferer pair as compared to pairwise bandwidth test (UBT). PIE identifies both multiple strong and weak interferers accurately (all estimates are withing +/- 0.15 of UBT LIR values). PIE is able to identify the extent of interference accurately in the presence of multiple strong and weak interferers.

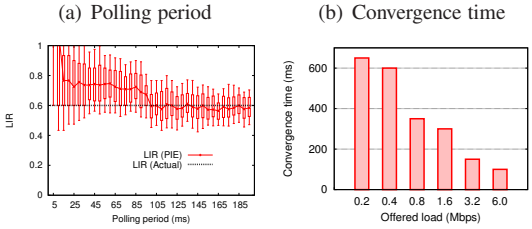

Figure 11: (a) Impact of polling period on the accuracy of the interference measures produced by PIE . LIR value stabilizes for polling periods greater than 100ms. The experiment time was adjusted to ensure same sample size for different polling periods. (b) Convergence time for a canonical hidden terminal link as a function of traffic load on the link and the interferer.

5.2.1 Polling interval

Any online interference estimation mechanism must identify conflicts in real time to be useful. In PIE , the controller periodically polls the APs for transmission summaries and then determines link conflicts. Higher polling periods can provide more information to the controller, thereby improving the quality of interference estimation. However, having a higher polling period also makes the system less responsive, which may be critical to dynamic interference scenarios. Here we evaluate the performance of PIE with different polling periods and determine the minimum period for which PIE can provide stable LIR values. We define a LIR value reported by PIE to be stable when the 90th and 10th percentiles of the LIR estimates differ by less than 0.1 of the mean LIR value. Figure 11 (a) demonstrates that a value of 100 ms provides a good compromise between reactivity and accuracy.

Note that smaller polling periods will also increase the communication overheads for sending traffic reports from the AP to the Controller. Using an average packet size of 600 bytes, and a medium constantly busy at 54 Mbps, the AP in PIE will have to store a summary for 1125 packets. This results in 9 KBytes sent from each AP every 100 ms, i.e. 1 Mbps, easily sustained by the AP.

(a) Light traffic (b) Heavy traffic

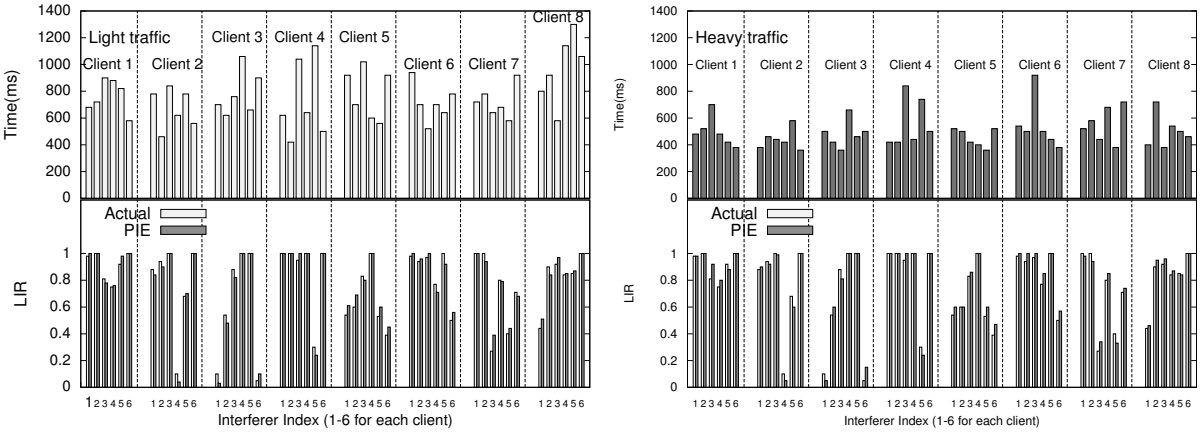

Figure 12: Convergence time and accuracy for PIE on a 7 AP - 8 Client topology under realistic patterns replayed for a period of (a) light client activity and (b) heavy client activity (using TCP). Top part of both figures shows the convergence time for each link-interferer pair and the bottom figure shows its corresponding accuracy when traffic traces are replayed on our representative topology. As shown in the figure, for light (heavy) traffic scenarios, PIE takes 1150ms (650ms) or less for 95% link-interferer pairs to converge within $\pm$ 0.1 of their actual value.

5.2.2 Convergence time

Convergence time is defined as the amount of time taken by PIE to gather sufficient samples to compute an accurate LIR estimate (within $\pm$0.1 of ground truth). Accordingly, the time taken by PIE to converge on an accurate estimate for link interference depends on two key factors: i) the polling period used by PIE to collect statistics from the APs, and ii) the actual amount of traffic that is captured by the APs in a given polling period. We first understand the impact of traffic load on the convergence of PIE by systematically varying the load on the canonical two link topology. Figure 11(b) shows the convergence time for a canonical hidden terminal link as a function of traffic load on the link and the interferer. Both the link and the interferer use a physical data rate of 6Mbps, while the traffic load is varied from 6Mbps (saturated) to 0.2 Mbps (light). Reduction in traffic load leads to longer convergence times because of the reduced frequency of interference events. Note, however, that LIR values would correspond to perceived client performance degradation only under relatively heavy loads, in which case PIE could capture events in 100 ms. In contrast, the measurement overheads of prior bandwidth test based active interference estimation mechanisms (e.g. Interference-maps [15]) is in the range of 20-30 seconds per link-pair [16].

Next, in order to understand the convergence of PIE under realistic traffic patterns, we replay a real WLAN trace [18] on the representative (7AP - 8 Client) topology (described in Section 5.1).

5.3 Experiments with real wireless traces

We now present experimental results on the performance of PIE using the publicly available Sigcomm 2004 traffic traces [18]. The Sigcomm trace was partitioned into heavy, medium, and light periods corresponding to periods with airtime utilization of more than 50%, between 20-50%, and less than 20% respectively, at different times of the conference [19]. In these traces, HTTP transactions were categorized into a series of HTTP sessions. Each session consists of a set of timestamped operations starting with a connect, followed by a series of sends and receives (called transactions), and finally a close. The HTTP sessions are then replayed on our testbed using the mechanism described in [10]. In our experiments, each client emulated the behavior of one real client from the trace, faithfully imitating its HTTP transactions. We use TCP as the underlying transport protocol for trace replay.

Figure 12 shows the convergence time (top plot) and accuracy (bottom plot) of PIE for each link-interferer pair when access patterns from the light and heavy load periods are replayed on the representative topology. As shown in the figure, for light (heavy) trafc scenarios, PIE converges to $\pm$ 0.1 of the actual LIR value within 1150 ms (800 ms) for more than 95% of the link-interferer pairs [7] Further figure 13 shows the distribution of convergence time of PIE for different link-interferer pairs under all three load periods. As expected, the convergence time is smaller for higher activity periods. The median convergence time for the light, medium, and heavy traffic loads are 400 ms, 620 ms, and 700 ms respectively.

[7] We skip detailed results from medium activity periods and instead show only the distribution for medium activity period to save space.

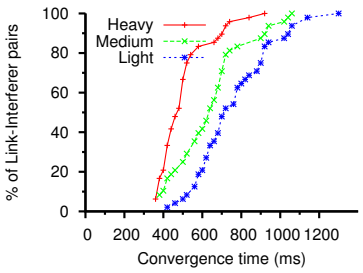

Figure 13: Distribution of convergence time for all link-interferer pairs under realistic traffic scenarios. Traffic scenarios (TCP based) are classified as heavy, medium and light depending on the total traffic load. As expected, PIE's convergence is faster for heavy traffic scenarios (median of 400 ms), followed by medium (median of 620 ms) and light (median of 700) traffic.

5.4 Applying PIE to improve WLAN performance

Being able to track interference in a highly dynamic environment may be considered as an admirable academic exercise. In this section, we will prove that access to such information can better enable a number of real time mechanisms that have been proposed for the performance optimization of wireless networks. To that end, we have integrated PIE with three such mechanisms (channel selection, dynamic packet scheduling, and power control) and tested them on two different testbeds. Our results clearly demonstrate that all these functions become a viable tool in the hands of network operators as long as we can supply reliable interference information in real time.

We use the same 7 AP and 8 client topology that we described in §5. We set the polling period to 1 second as per our observation in §5.2.2, thus capturing interference accurately even under low traffic loads. In mobility experiments, each client moves along a corridor at ∼0.25 m/s. We use UDP traffic for our experiments to measure the performance of PIE with different applications. We also perform experiments with TCP traffic for centralized scheduling application and report the results for the same.

Conflict graph	Mechanism (Num Channels)	System Tput(Mbps)	Jain's Fairness Index
NA	Single (1)	9.2	0.52
NA	LCCS (3)	17.1	0.58
UBT	Conflict aware (3)	24.6	0.72
PIE	**Conflict aware (3)**	**24.9**	**0.71**

Table 3: Performance of conflict-aware channel assignment (using conflict graph generated by PIE and bandwidth tests) as compared with single channel and LCCS (least congested channel search) assignments. Under static conditions, PIE leads to similar results as UBT, offering significant improvement compared to single channel and LCCS assignments. Note that UBT being an active technique has significantly higher measurement overhead and is not practical.

5.4.1 Application I: Channel assignment

Efficiently assigning channels to access points (APs) in an enterprise WLAN can significantly affect the network performance and capacity [14, 20]. We implement a conflict aware channel assignment heuristic (Randomized Compaction), proposed in [20], that takes a conflict graph as input and performs channel assignment with the objective to minimize interference. We compare the performance of the conflict-aware channel assignment scheme when based on the conflict graph generated by PIE and that of unicast bandwidth tests.

Table 3 shows the total system throughput and Jain's fairness index achieved by each channel assignment mechanism. Bandwidth tests are performed with unicast traffic at data rate and packet size of 6Mbps and 1400 bytes. Experiments are performed under static settings for a fair comparison with bandwidth tests. We consider the conflict graph generated by bandwidth tests as the true interference information. Results are averaged over 20 runs. We note that conflict aware channel assignment significantly improves system throughput over LCCS [11] (least congested channel search) and single channel assignments. Moreover, the performance of the heuristic is similar with PIE and bandwidth tests, illustrating PIE's ability to generate high quality conflict graphs in real time.

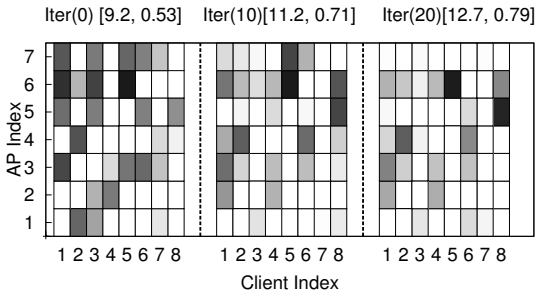

Figure 14: Performance of an iterative power control mechanism that uses PIE. Each matrix represents the conflict graph, with overall capacity (total system throughput in Mbps) and Jain's fairness index listed in the title. Intensity of darkness is proportional to the extent of interference. The final state corresponds to reduced interference, improved overall network capacity and fairness.

5.4.2 Application II: Transmit Power Control

We implement a simple centralized power control heuristic that uses the dynamic conflict information produced by PIE to reduce interference in the system. We measure the performance of the system through LIR_{all}, i.e. the sum of LIR values, for all link-interferer pairs in the system. Our goal is then to maximize this value by iterating over different power levels of the transmitters.

In each iteration of power control, we identify the most dominant interferer, as the AP that sources links with the minimum cumulative LIR. We reduce its transmit power

(by 10mW) and recompute the conflict graph using PIE. If the new conflict graph has lower cumulative LIR, then we discard the new power settings and reduce the power level of the next strongest interferer. In this way, we always move to a new set of power levels only if it increases the overall performance of the system. We quit when there is no improvement in the overall LIR value for 10 iterations.

Figure 14 shows the impact of such a power control mechanism. We present three matrices that capture the interference caused by each AP (row) to each client (column) in the network (the darker the cell, the stronger the interference). The title of each matrix further captures the iteration, the overall network capacity, and the fairness index. The leftmost matrix corresponds to the default power level setting, while the middle and right columns indicate the intermediate and final stages of the power level settings achieved by the aforementioned power control heuristic. We clearly see that our simple power control mechanism reduces the overall conflict in the system (matrix cells get increasingly lighter), while increasing overall network capacity and fairness. The point of this evaluation is not on the power control mechanism itself, since there are a number of solutions that could achieve such an objective more effectively (like [17]). Our focus is to demonstrate the effectiveness of PIE when used for power control.

5.4.3 Application III: Centralized scheduling

Accurate, fast and scalable conflict graph construction is critical for realizing centralized data plane mechanisms. In a recent work on centralized data path scheduling (Centaur [22]), authors relied on micro-probing [3], an online mechanism that performs micro experiments to determine link conflicts. Although micro-probing can generate an accurate conflict graph in very short time scales (4 seconds for a 10 link topology), it may still be inefficient in high mobility scenarios, especially given the need for silencing the network during the measurement of the conflict graph. We re-evaluate the performance of Centaur using the conflict graph generated by PIE and contrast it to bandwidth tests for consistency. We show that PIE improves the performance of Centaur under high mobility and varying traffic properties (variable packet sizes and data rates).

Table 4 shows the Centaur's performance when operating on conflict information from PIE and bandwidth tests respectively, in one static and one mobile scenario. The UBT conflict graph is generated using 6 Mbps and a fixed packet size of 1400 bytes for static client locations. Due to the overhead of recomputing bandwidth tests, we use the static conflict graph for the mobility scenario too. One can clearly see that exploiting real time conflict information in scheduling is not only increasing the overall network throughput but also the fairness index across clients. More interestingly, the inaccuracies in the conflict graph generated using bandwidth tests almost negate the benefits of centralized scheduling under mobility. We performed similar experiments with auto-rate and observe that Centaur with PIE 's conflict graph provides 32% overall system throughput gain as compared to using the conflict graph generated using bandwidth tests under static scenarios (6Mbps, 1400 bytes).

Scenario	Mechanism	System Tput(Mbps)	Jain's Fairness Index
Static(UDP)	DCF	11.2	0.64
	Centaur (UBT)	12.6	0.88
	Centaur (PIE)	**13.0**	**0.84**
Static(TCP)	DCF	9.5	0.60
	Centaur (UBT)	12.2	0.85
	Centaur (PIE)	**12.4**	**0.89**
Mobile(UDP)	DCF	10.1	0.61
	Centaur (UBT)	10.4	0.71
	Centaur (PIE)	**12.4**	**0.95**

Table 4: Performance of centralized scheduling (Centaur) using PIE 's conflict graph. UBT and PIE lead to equivalent performance under static settings. The introduction of mobility confirms PIE's superiority to provide real time information. Note that UBT has very high measurement overheads compared to PIE .

TCP performance: We also analyze TCP performance for different conflict graphs. We observe system throughputs (fairness) of 9.5 Mbps (0.60), 12.2 Mbps (0.85) and 12.4 Mbps (0.89) for DCF, Centaur(UBT) and Centaur(PIE) respectively. As expected UBT and PIE perform close to each other and outperform DCF. However, as noted earlier, the measurement overhead of UBT is much higher than PIE making it impractical for real time mechanisms like Centaur.

5.4.4 Application IV: Wireless troubleshooting

Beyond PIE's ability to enable real time performance optimization in enterprise WLANs, its real time nature allows it to serve as a diagnosis tool that could be used proactively by a network operator to avoid performance problems. We test this property by running PIE in two production 802.11b/g WLANs (W_1 and W_2), co-located with our two testbeds.

These WLANs differ from each other in many significant ways as follows. $WLAN_1$ spans 5 floors of a building and uses 9 APs manufactured by vendor A. The network administrator was responsible for conducting RF site surveys, identifying locations to place the APs, and manually assigning the channel of operation of each AP to minimize interference. Exactly 3 APs were placed on channels 1, 6, and 11 in $WLAN_1$ to minimize the level of inter-AP interference. In contrast, $WLAN_2$ occupies a single floor of a different building, uses 21 APs manufactured by a different vendor, B, and features a controller in charge of dynamic channel assignment. The number of APs on each channel, thus, varies over time. In $WLAN_2$ the vendor was responsible for conducting the RF site surveys and making AP placement decisions.

WLAN	HT-Links ($LIR < 0.7$)	Anomaly-Link pairs ($Ratio < 0.2$)
WLAN1	31 / 386	231 / 1087
WLAN2	53 / 464	305 / 1391

Table 5: Performance issues observed in two production WLANs. The extent of hidden terminal interference ranges from 8% to 11% but can be significant for a small number of links. Rate anomaly affects approximately 20% of the links in both networks.

We select testbed nodes closest to the production APs to provide transmission reports to the PIE controller, sniffing the transmissions on the operational network. We use those reports to measure the carrier sense and interference relationships between different links in the production WLAN. PIE reveals two performance issues:

1) Hidden terminals: Performance degradation beyond a certain level due to interference can significantly impact client performance. We set LIR_{thresh} equal to 0.7 to identify those links that suffer more than 30% reduction in their LIR under interference and classify them as hidden terminals.

2) Rate anomaly: Rate anomaly is a well documented problem [23] in wireless environments. If a transmitter of a link operating at a high data rate (say 54 Mbps), carrier senses the transmitter of another link operating at a low rate (say 6Mbps), then the link operating at higher rate will experience significant slowdown in throughput (by a factor of 1/10 in this case). We classify a given link pair as a case of rate anomaly, when the ratio of their transmission rates is less than 0.2.

Both these issues are observed in both production networks. Table 5 shows the extent of hidden interference and rate anomaly in the two WLANs. The extent of hidden interference is rather limited (8% for WLAN1 and 11% for WLAN2). For comparison, Jigsaw [8] also reports that 5% of their links observe an LIR of less than 0.8. While limited on average, however, we do still observe, across both WLANs that hidden interference can lead to up to 70% LIR degradation for as many as 4% and 3% of the links in WLAN 1 and 2 respectively.

In terms of rate anomaly issues, we observe that for about 20% carrier sensing link pairs, the transmission rates differ by more than 80%. This could be one of the reasons for sudden performance slowdown experienced by perfectly good quality links in WLANs.

6 Conclusions

We presented a detailed evaluation of a passive, real time interference estimation mechanism (PIE). We showed that PIE is accurate in estimating link interference and can also adapt to changing interference patterns in real time. This enables PIE to be especially effective in realistic wireless environments, where client mobility, variable transmission rates, and bursty traffic result in changing interference scenarios, thereby limiting the usefulness of static bandwidth test mechanisms. Further, we showed that PIE is completely passive, does not require client support, and does not cause any network downtime, making it attractive for use in real WLAN settings. We have integrated PIE with interference mitigation mechanisms like centralized scheduling, transmit power control, and channel assignment and showed that PIE can enable these mechanisms to function efficiently and dynamically by providing an accurate conflict graph in real time. We also used PIE to monitor two production WLANs and demonstrated that PIE can diagnose certain performance issues in real systems.

Acknowledgments: We would like to thank the anonymous reviewers and our shepherd Hari Balakrishnan, whose comments helped bring the paper into its final form. V. Shrivastava, S. Rayanchu, and S. Banerjee have been supported in part by US NSF through awards CNS-1040648, CNS-0916955, CNS-0855201, CNS-0747177, and CNS-1059306.

References

[1] Madwifi atheros drivers. http://madwifi-project.org.
[2] D. Aguayo et al. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, 2004.
[3] N. Ahmed and S. Keshav. Smarta: A self-managing architecture for thin access points. In *CoNEXT*, 2006.
[4] N. Ahmed et al. Online estimation of RF interference. In *CoNext*, 2008.
[5] N. Ahmed et al. Measuring multi-parameter conflict graphs for 802.11 networks. In *MC2R*, 2009.
[6] K. Cai et al. Non-intrusive, dynamic interference detection for 802.11 networks. In *IMC*, 2009.
[7] R. Chandra, J. Padhye, A. Wolman, and B. Zill. A location-based management system for enterprise wireless LANs. In *NSDI*, 2007.
[8] Y.-C. Cheng et al. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM*, 2006.
[9] Y.-C. Cheng et al. Automating cross-layer diagnosis of enterprise wireless networks. In *SIGCOMM*, 2007.
[10] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye. Feasibility study of mesh networks for all-wireless offices. In *MobiSys*, 2006.
[11] J. Geier. Assigning 802.11b access point channels. *Wi-Fi Planet '04*.
[12] IEEE. Wireless LAN Medium Access Control (MAC) and physical layer (PHY) spec, IEEE 802.11 standard. *IEEE Standard 802.11*, 1999.
[13] R. Mahajan et al. Analyzing the MAC-level behavior of wireless networks in the wild. *SIGCOMM 2006*.
[14] A. Mishra et al. A client-driven approach for channel management in wireless LANs. In *INFOCOM*, 2006.
[15] D. Niculescu. Interference map for 802.11 networks. In *IMC*, 2007.
[16] J. Padhye et al. Estimation of link interference in static multi-hop wireless networks. In *IMC*, 2005.
[17] K. Ramachandran et al. Symphony: synchronous two-phase rate and power control in 802.11 WLANs. In *Mobisys*, 2008.
[18] M. Rodrig et al. CRAWDAD data set uw/sigcomm2004.
[19] M. Rodrig et al. Measurement-based characterization of 802.11 in a hotspot setting. In *SIGCOMM E-WIND*, 2005.
[20] E. Rozner, Y. Mehta, A. Akella, and L. Qiu. Traffic-aware channel assignment in enterprise wireless LANs. In *ICNP*, 2007.
[21] A. Sheth et al. MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs. In *MobiSys*, 2006.
[22] V. Shrivastava et al. CENTAUR: realizing the full potential of centralized WLANs through a hybrid data path. In *MobiCom*, 2009.
[23] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate WLANs. In *USENIX*, 2004.
[24] M. Vutukuru, K. Jamieson, and H. Balakrishnan. Harnessing exposed terminals in wireless networks. In *NSDI*, 2008.
[25] White-paper from Meru Networks. Microcell deployments: Making a bad problem worse for pervasive wireless LAN deployments. http://www.merunetworks.com/pdf/whitepapers/.
[26] J. Yeo, M. Youssef, and A. Agrawala. A framework for wireless LAN monitoring and its applications. In *WiSe*, 2004.