Passive Measurement of Interference in WiFi Networks with Application in Misbehavior Detection

Utpal Paul, Anand Kashyap, Ritesh Maheshwari, and Samir R. Das

Abstract—We present a tool to estimate the interference between nodes and links in a live wireless network by *passive monitoring* of wireless traffic. This tool does not require any controlled experiments, injection of probe traffic in the network, or even access to the network nodes. Our approach requires deploying multiple sniffers across the network to capture wireless traffic traces. These traces are then analyzed using a machine learning approach to infer the carrier-sense relationship between network nodes. This coupled with an estimation of collision probabilities helps us to deduce the interference relationships. We also demonstrate an important application of this tool-detection of selfish carrier-sense behavior. This is based on identifying any asymmetry in carrier-sense behavior between node pairs and finding multiple witnesses to raise confidence. We evaluate the effectiveness of the tool for both the applications using extensive experiments and simulation. Experimental and simulation results demonstrate that the proposed approach of estimating interference relations is significantly more accurate than simpler heuristics and quite competitive with active measurements. We also validate the approach in a real Wireless LAN environment. Evaluations using a real testbed as well as ns2 simulation studies demonstrate excellent detection ability of the selfish behavior. On the other hand, the metric of selfishness used to estimate selfish behavior matches closely with actual degree of selfishness observed.

Index Terms—802.11 protocol, hidden Markov model, MAC layer misbehavior, interference

1 Introduction

Poor WiFi performance is often attributed to wireless interference in highly loaded networking scenarios [1], [2]. While a lot of research has been conducted in understanding wireless interference in a theoretical context, real network deployments are yet to gain from it. In this work,[1] we present a technique to model and understand the wireless interference between network nodes and links in realistic WiFi network deployments. The goal is to do this in the most unobtrusive fashion possible: 1) *Without installing any monitoring software on the network nodes*. This is motivated by practicality as many APs are often closed devices, and clients may not be always be privy to new software; 2) *Using a completely passive technique*. This is important as active measurements impact (and are impacted by) network traffic.

To achieve these goals, our approach uses a distributed set of "sniffers" that capture and record wireless frame traces. We then analyze the trace to understand the interference relations. While this is true that this approach requires additional hardware for measurement, this can be viewed as a form of third-party solution. Such independent third-party solutions for wireless monitoring are not uncommon in industry [5], [6]. The research community has also provided similar approaches. See, for example, DAIR [7], [8], Jigsaw [9], and Wit [10]. While these approaches provide many monitoring solutions, they still do not provide fundamental understanding of interference relations between network nodes and links.

Aside from understanding interference relationships, there are other applications of the technique we develop. Certain types of selfish behaviors can be detected via this approach—an example we will demonstrate. A selfish node can gain unfair share of the available bandwidth by manipulating different MAC protocol parameters, such as the clear channel assessment (CCA) threshold, or the backoff window size. This can deliver an unfair bandwidth advantage to a selfish node [11] and can be used to even launch a denial of service attack. A node, for example, can be selfish by raising the CCA threshold. This can effectively disable its carrier sensing and creates more transmission opportunities for the selfish node. This can also cause collisions, and thereby force the other transmitters in the vicinity to perform backoff. While the selfish node itself may also undergo a collision, the backoff period will be shorter as it will not freeze its backoff counter when carrier sensing is disabled. We can detect the selfish carrier-sense behavior using the pairwise interference relationships discovered by the proposed technique. In our knowledge, this problem has been explored only in one paper [11], that provides a limited solution using a nonpassive technique.

1. An earlier version of this work was published in two conferences in two parts [3], [4].

- *U. Paul and S.R. Das are with the Computer Science Department, Stony Brook University, Stony Brook, NY 11794-4400. E-mail: {upaul, samir}@cs.sunysb.edu.*
- *A. Kashyap is with Symantec Research Labs, Mountain View, 65 Rio Robles E Unit 2211, San Jose, CA 95134. E-mail: kashyap.anand@gmail.com.*
- *R. Maheswari is with Akamai Technologies, 18 Day Street, Apartment 308, Somerville, MA 02144. E-mail: riteshm@gmail.com.*

Manuscript received 21 Nov. 2010; revised 10 Nov. 2011; accepted 18 Nov. 2011; published online 8 Dec. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-11-0533. Digital Object Identifier no. 10.1109/TMC.2011.259.

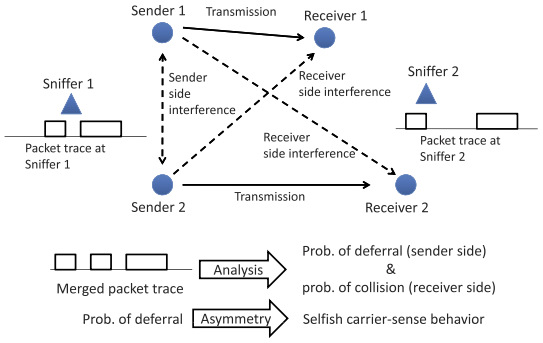

Fig. 1. Overview of the approach.

1.1 Approach

A set of "sniffers" are deployed to collect traffic traces from a live network. The traffic traces are then merged using existing merging techniques for distributed sniffer traces [9], [10], [12].[2] Then, we use a machine learning-based approach to analyze the merged traces to infer sender-side interference relationships. It also determines the receiver-side interferences. See Fig. 1. More specifically, the approach determines for each link (or node), which other links (or nodes) it interferes with, as well as the extent or degree of interference.

For detecting selfish behavior, we use the sender-side interference relation to identify *asymmetric behavior* between network nodes. This means that between a given pair of nodes, while one node can sense the transmission of the other node, the converse is not true. *The main idea of our approach is that significant asymmetry in favour of a specific node when witnessed persistently by multiple other nodes is indicative of selfish behavior.* This is because such asymmetry may be very unusual due to normal wireless channel effects.

Our approach can be used as a "toolbox" with two important applications: understanding the interference properties, and detecting selfish behavior in an arbitrary WiFi network, regardless of the topology or architecture. System managers can use this tool to perform capacity planning and appropriate radio resource management, such as assignment of channels, transmit power levels or directions when using directional antennas. In addition, this tool can act as a "police" to detect the malicious user activity and can provide a significant insight about WiFi interference behavior in large installations, potentially influencing future standards design.

Because of its passive nature, our approach is dependent on the sufficiency of the available network traffic. The most important challenge is to make accurate estimation of interference for traffic of unknown and arbitrary nature, especially in presence of low load in the network. Also, accurate identification is very challenging when a selfish node exhibits probabilistic behavior to avoid detection.

We discuss related work in Section 2 and the broad approach in Section 3. The details of the HMM formulation are covered in Section 4. Section 5 contains the experimental evaluations for interference relation. Section 6 defines the metric to identify selfish nodes and Section 7 presents the experimental evaluations for selfish carrier-sensing detection. We will conclude in Section 8.

2. These techniques also infer and add the packets that are missing from the merged trace.

2 Related Work

2.1 Analyzing Interference

Interference in an 802.11 wireless network can be readily measured by putting saturated traffic on two links simultaneously and measuring the aggregate throughput. The decrease in throughput due to interference from the other transmission indicates the amount of interference. This approach ordinarily needs $O(n^4)$ measurements for an n node network. However, [13] outlines a method to do this with only $O(n^2)$ measurements. More sophisticated approaches do not perform direct measurements as above, but uses certain modeling steps to reduce the number of measurements to $O(n)$. The idea here is to 1) measure Received Signal Strength (RSS) on each link using broadcast beacons, 2) perform a profiling study describing the deferral and packet capture behavior of the radio interface, 3) develop a suitable MAC-layer model. Together the above can estimate interference between active links and link capacities in presence of interfering traffic. There are different variations of this basic approach presented in [14], [15], [16] which need active measurement. While the requirement of a quiet, interference-free environment to do RSS measurements makes these methods unrealistic in live networks, the method presented in [14] can model interference by doing measurement even in the presence of external interference. However, the profiling needs to be done a priori.

In addition to the above, there are various sundry works on evaluating interference characteristics in an 802.11 network. For example, in [17], Jamieson et al. investigate the impact of carrier sensing. In [18], Chang et al. develop a model for the physical layer capture. In [19], Das et al. show that pairwise interference modeling is often not accurate and multiple interferers must be accounted for. In [20], Magistretti et al. present an inference tool to infer the activity share among a set of conflicting links. In [3], we present our approach of indentifying interference relations, but with limited evaluation.

2.2 Detecting MAC-Layer Misbehavior in 802.11

Most of the existing MAC-layer misbehavior detection techniques only attempt to detect one type of selfish behavior: backoff manipulation in 802.11. They use different methods, such as game theoretic approach [21], Sequential Probability Ratio Test (SPRT) [22], nonparametric cumulative sum (CUSUM) test [23], coordination from the receiver [24] to identify backoff manipulation or to restrict the sender from being selfish. DOMINO [25] can detect other misbehaviors in addition to backoff manipulation, e.g., sending "scrambled frames," using smaller DIFS and using oversized NAV. None of these techniques can detect selfish carrier-sense behavior and thus can be complementary to the approach described in this paper.

Manipulation of the carrier-sense behavior is harder to detect. This is because normal fluctuations of wireless channel must be distinguished from manipulated carrier sensing. In our knowledge, only one paper [11] has addressed this issue before our work in [4]. The technique proposed in [11] relies on a strong assumption that the selfish node that has increased its CCA threshold is unlikely to correctly recognize low power transmissions from the AP as legitimate packets. Thus, by sending low power probes,

the AP can potentially detect such nodes. This assumption implies that packet reception with power lower than CCA threshold is not possible, as such packets are treated as noise. However, the attacker can avoid detection by simply changing the CCA threshold only when it transmits a packet and reverting back to the normal threshold right after the transmission.[3] Also, depending on how the radio transceiver is designed, packet reception success may not be dependent on the CCA threshold. Also, this technique is not passive.

2.3 Use of Distributed Sniffers

Techniques based on using distributed sniffers can be found in a number of measurement studies for the purpose of learning various properties of live network such as congestion [1], protocol behavior in a hotspot setting [2], [9], [10], etc. The DAIR system also uses such an approach for troubleshooting [7] and security [8]. More details on similar related works appear in [3, Section 2.2]. In this paper, we employ a technique similar to [12] to merge individual traces into a unified trace. However, unlike all the previous studies, our focus is on learning the interference relations and detecting selfish carrier-sense behavior in the network.

3 Overall Approach

3.1 Problem Statement

In 802.11, interference can occur either at the "sender side" or at the "receiver side" (or both) [15]. Sender side interference pertains to deferral due to carrier sensing. In this case, one node freezes its backoff counter and waits when it senses the second node's transmission. In case of receiver side interference, overlapped packet transmission causes collisions at the receiver. This requires packet retransmission. In both cases, the sender additionally has to go through a backoff period, when the medium must be sensed idle.[4] The net effect of the interference is reduction of throughput capacity of the network.

Our general goal is to understand the deferral behavior that accounts for the sender side interference. To detect selfish carrier-sense behavior, we need to identify the asymmetry in the deferral behavior. The deferral behavior between two nodes, X and Y is said to be asymmetric if Y defers for X's transmission and X does not defer for Y's, or vice versa. Such asymmetry is possible in wireless networks due to interface heterogeneity. But it is simply unlikely that a node X demonstrates similar asymmetry with many such Y's in the same direction. Our strategy is to flag such nodes as potentially selfish, with degree of selfishness indicated by extent of asymmetries exhibited and the number of such Y's (called "witnesses").

For modeling convenience, we consider interference between node or link pairs only. Note that it will allow us to capture the "physical interference" [26] where a given link is interfered collectively by a set of other links, not by a single link alone. This is due to the additive nature of the received power. However, pairwise consideration can still be quite powerful in practice. Also, in reality the probability of having multiple concurrent packet transmission is very small even when there are many active flows in the network. For example, Mahajan et al. [10] analyzed a major trace collected during the SIGCOMM 2004 conference and found that only 0.45 percent of packets actually overlapped in transmission. This limits the usefulness of having a more elaborate higher order model for deconstructing interference relationship. On the other hand, pairwise relationship can be enough for our method of detecting selfish carrier-sense behavior. *We do note that this simplification is not fundamental to our basic technique. The technique can be extended, albeit with higher computational cost, to physical interference.*

In wireless networks, interference is better expressed in terms of probabilities because of the inherent fluctuation of the signal power due to fading effects and probabilistic dependency of error rates with signal to interference plus noise ratio (SINR). Prior measurement and modeling studies have elaborated on this aspect [13], [15]. Thus, *in this work, we estimate via passive monitoring the nonbinary, pairwise interference between any two network nodes or links, in terms of probability of interference*. For any link pair, the probability of interference is given by

$$p_d + (1 - p_d)p_c, \tag{1}$$

where p_d is the "probability of deferral" between the senders, and p_c is the "probability of collision" at the receivers if both senders transmit together.[5] See also Fig. 1. When considering node pairs only, probability of interference is just p_d, assuming symmetric interference between these two nodes.

If one of the nodes in a node pair shows selfish carrier-sense behavior, the sender-side interference (p_d) should be very asymmetric. Thus, our next goal is to quantify the asymmetry for each pair of nodes in the network. For a given pair of nodes, X and Y, we estimate the probability $P_{\text{def}}(X, Y)$ that node X defers to node Y's transmission. We do this estimation for all node pairs in either direction. As mentioned before, significant asymmetry in this probability indicates possible selfishness. Let us assume that there is asymmetry in favor of X, i.e., $P_{\text{def}}(X, Y) \ll P_{\text{def}}(Y, X)$. If this is also witnessed by more nodes such as Z, i.e., there exists several $Z \neq Y$ such that $P_{\text{def}}(X, Z) \ll P_{\text{def}}(Z, X)$ we have more confidence that X is behaving in a selfish manner.

3.2 Discussions

To estimate the interference relations between a given pair of nodes, our technique needs to have instances when simultaneous transmissions are attempted by the two nodes. The conjecture here is that if one observes the live network traffic for a long enough period, enough of such instances will be available for each node pair. Our goal is to 1) identify such instances, and 2) infer the deferral behaviors during such instances. There are several challenges here. First, creating a complete and accurate trace is itself a difficult problem. There are many approaches proposed in literature to create a complete trace. But for our technique, incomplete

3. There may indeed be a latency issue that can slow down the selfish node if such changes are frequent. But we do not consider this to be a fundamental issue.

4. We are assuming that the reader has an overall idea of the 802.11 MAC protocol. Specific details will be brought up as necessary.

5. This definition ignores ACKs for modeling and notational convenience as in [13], [15], and is not a limitation. We indeed use unicast traffic with ACKs for evaluation.

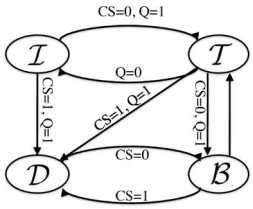

Fig. 2. State transition diagram for a single sender. $\mathrm{CS} = 0(\mathrm{CS} = 1)$ means that the carrier is sensed idle (busy). $\mathrm{Q} = 0$ $(\mathrm{Q} = 1)$ means that the interface packet queue is empty (nonempty).

trace may suffice as long as it is statistically similar to the complete trace. Second, unknown load of the nodes makes it harder to estimate the deferral behavior. In our approach, we utilize the strategy of analyzing interpacket times which can provide certain confidence. Third, heuristics can be used to infer the deferral behavior. But straightforward heuristics may have limited power. More details about these challenges appear in [3].

3.3 Approach

We need to come up with a rigorous statistical modeling approach to determine deferral behavior among network nodes. Our basic approach is as follows: we model the 802.11 MAC-layer operations of two sender nodes in the network (say, X, Y) via a Markov chain. The parameters of this chain (essentially the state transition probabilities) are estimated from the observed trace using an approach based on the Hidden Markov Model (HMM) [27]. These parameters in turn can estimate the deferral probabilities. We devote the entire next section describing the HMM-based approach.

4 Hidden Markov Model for Sender-Side Interactions

A hidden Markov model [27] represents a system as a Markov chain with unknown parameters. Here the states of the Markov chain are not directly visible, but some observation symbols influenced by the states are visible. The unknown parameters (such as the state transition probabilities of the Markov chain) can be learned using different standard methods [27], [28], [29] with the help of the observed sequence of observation symbols. Various machine learning applications such as pattern, speech, and handwriting recognition have used HMM technique. We will be using the HMM approach for modeling interactions between a pair of senders in an 802.11 network and inferring sender-side interference relations (deferral behavior) between them.

4.1 Markov Chain

Each sender in 802.11 MAC protocol can be modeled as a Markov chain [3], [30] as shown in Fig. 2. A sender node, say X, is found in one of the following four states—"idle," "backoff," "defer," and "transmit." The essence of the 802.11 MAC protocol lies in these four states. We intentionally ignore interframe spacings (e.g., DIFS) to keep the chain simple. In the rest of the paper, we call the four states $\mathcal{I}$, $\mathcal{B}$, $\mathcal{D}$, and $\mathcal{T}$, respectively for the sake of brevity. The high level description of this chain can be found in [3].

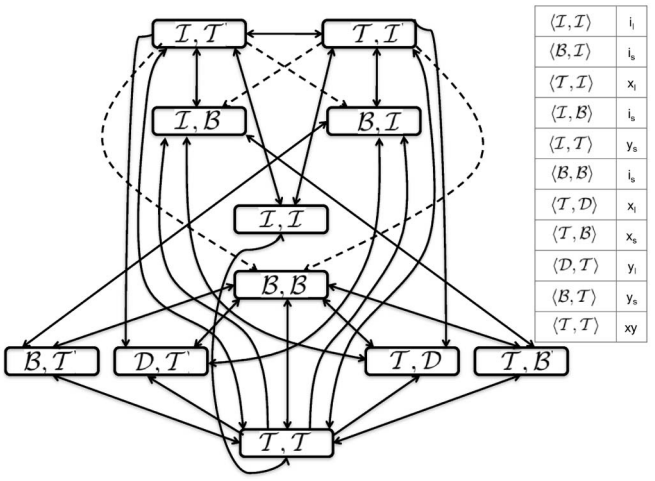

Fig. 3. Markov model of the combined MAC Layer behavior of two nodes (sender side only). Note that some arrows are bidirectional.

Note that the state transition probability between $\mathcal{B}$ and $\mathcal{D}$ of the corresponding sender node is influenced by the states of other nodes (i.e., transmitting or not) in the network, and the deferral probabilities between the sender and these nodes. Similar argument applies for the transition probabilities from $\mathcal{I}$ to $\mathcal{D}$ and $\mathcal{T}$, and transition probabilities from $\mathcal{T}$ to $\mathcal{D}$ and $\mathcal{B}$.

Since the state transitions of the Markov chain for a given sender is impacted by the transmissions from other nodes, a Markov model of a single sender is not enough to get the complete picture of the network behavior. Instead, a combined Markov model needs to be considered. Here, each state is a tuple consisting of states of individual nodes. Such a Markov chain would be intractable as it would lead to a state space explosion with exponential number of states. Since we focus mainly on determining the pairwise interference relationships, and our technique to detect selfish behavior needs only pair-wise deferral behavior, we can restrict ourselves to the consideration of a combined Markov chain for only a pair of nodes, say X and Y. Each state in this Markov chain is a two-tuple consisting of the states of X and Y. For example, the state where X transmits and Y defers would be $\langle \mathcal{T}, \mathcal{D} \rangle$. Out of 16 possible states in theory, five states are not legal (e.g., $\langle \mathcal{D}, \mathcal{D} \rangle$, $\langle \mathcal{D}, \mathcal{B} \rangle$ etc.[6]), leaving 11 ;possible states. See Fig. 3 for the two-node combined Markov chain. (Only the solid lines indicate valid transitions. The dotted transition lines will be discussed later.)

The state transition probabilities between certain states in this Markov chain are determined by the deferral probabilities between X and Y. For example, transition probabilities from state $\langle \mathcal{B}, \mathcal{B} \rangle$ to state $\langle \mathcal{T}, \mathcal{D} \rangle$ or $\langle \mathcal{T}, \mathcal{B} \rangle$ would depend on deferral probability of Y with respect to X. Let us explain this using an example. Assume that Y carrier senses X (or Y can sense X's transmission) perfectly. Then when X moves from $\mathcal{B}$ to $\mathcal{T}$ state (i.e., starts transmitting as soon as the backoff interval is over), Y must also move from $\mathcal{B}$ to $\mathcal{D}$ as it defers to X's transmission by freezing its backoff countdown timer. If instead Y never carrier senses X, it will remain in the $\mathcal{B}$ state. The deferral probability of X and Y

6. Note that this Markov chain assumes only two nodes X and Y interact. Thus, for example, the state $\langle \mathcal{D}, \mathcal{D} \rangle$ is not possible as both nodes cannot defer at the same time.

depends on the number of instances when either of the nodes moves to $\mathcal{D}$ state.

Note again that this combined Markov chain is specified for a node pair only, as we are interested in pairwise interference. This process can be repeated for all pairs to determine the all-pair sender-side interference. We filter out the packets of just the two senders under consideration for analysis, and ignore the other packets. This may misinterpret an active node, deferring for a third node's transmission, as idle, and we may miss an opportunity to interpret the interaction between the particular pair as interfering or noninterfering. But, it is important to note that this does not create any incorrect interpretation. Recent studies [10] show that the number of instances of three or more nodes simultaneously being active is much less than that of only a pair of nodes being active. Thus, we should get enough instances of just a pair of nodes being active in a long trace. An alternate but computationally expensive method could try to identify portions of the trace where only the senders in a node pair being considered are active.

4.2 Observation Symbols

The state transition probabilities of the combined Markov chain depend on the deferral behavior between the two nodes under consideration. Thus, if we can learn the unknown state transition probabilities, this will in turn provide us the deferral relations. But the states of this Markov chain are not directly visible in the packet trace. Instead a set of observation symbols are visible. There are four possible observation symbols in the trace depending on whether X or Y transmits:

- i: neither X, nor Y transmitting.
- x: X transmitting.
- y: Y transmitting.
- xy: both X and Y transmitting.

We thus need to map each of the 11 states in this Markov chain to one of the four observation symbols. This mapping obviously is not unique as more than one state can map to the same observation symbol. For example, both states $\langle \mathcal{I}, \mathcal{I} \rangle$ and $\langle \mathcal{B}, \mathcal{B} \rangle$ map to the symbol i. Similarly, both $\langle \mathcal{B}, \mathcal{T} \rangle$ and $\langle \mathcal{D}, \mathcal{T} \rangle$ map to symbol y. The difficulty here is that backoff cannot be distinguished from defer or idle periods. This ambiguity can be reduced by using a heuristic that exploits the time duration of various observation symbols. This is elaborated below.

A backoff interval in 802.11 lasts for an integral number of slots ($20\ \mu$s in 802.11b) chosen randomly from a window of 0 to 31 slots (for first backoff stage[7]). This knowledge can be used to distinguish between backoff and idle/defer periods. The conjecture here is that defer and idle periods are very unlikely (though not impossible) to be within this bounded interval and also last for an integral number of slots like backoff period. But this strategy requires the clock accuracy within few microsecond, which demands specialized technique.

We thus use a weaker heuristic in this work that does not require strong clock accuracy. We assume that defer/idle periods are always longer than 31 slots and backoffs are always equal or shorter. This, however, introduces errors for very short idle time and small 802.11 frames with airtime less than 31 slots ($620\ \mu$s for 802.11b[8]). These sources of error make the results presented in the next sections as only a lower bound on the accuracy obtainable by the base technique. We keep this as our future work to remove the timing inaccuracy by using more sophisticated technique.

Based on the above weaker heuristic, each observation symbol (except xy) can be classified into two types. The symbol i can be either i_s or i_l, corresponding to short (≤ 31 slots) and long (> 31 slots), respectively. According to the heuristic, i_s is most likely output by $\langle \mathcal{B}, \mathcal{B} \rangle$ state, while i_l is most likely output by $\langle \mathcal{I}, \mathcal{I} \rangle$ state, for example. Similarly, the symbols x and y can be either x_s and x_l, and y_s and y_l, respectively to differentiate among the activities (defer/idle or backoff) of the non-transmitting node during that period. Fig. 3 shows the observation symbols for each state.

The heuristic described above helps us to distinguish between backoff and idle/defer periods. However, we still cannot differentiate between idle and defer. For this reason, both the states $\langle \mathcal{T}, \mathcal{I} \rangle$ and $\langle \mathcal{T}, \mathcal{D} \rangle$ map to the same observation symbol x_l. This implies that the transition from state $\langle \mathcal{T}, \mathcal{I} \rangle$ to state $\langle \mathcal{T}, \mathcal{D} \rangle$ will not be visible in the merged trace as there is no change in the observation symbol. Thus any transition from state $\langle \mathcal{T}, \mathcal{I} \rangle$ to any other state, for example, state $\langle \mathcal{I}, \mathcal{B} \rangle$ via state $\langle \mathcal{T}, \mathcal{D} \rangle$ will not be correctly interpreted. To overcome this problem, we force transition links from state $\langle \mathcal{T}, \mathcal{I} \rangle$ to states which have incoming transition from state $\langle \mathcal{T}, \mathcal{D} \rangle$. We refer to these links as virtual links. Similarly, we also add virtual links from state $\langle \mathcal{I}, \mathcal{T} \rangle$ symmetrically. Fig. 3 shows the virtual links in the model in dotted lines. After we calculate the transition probabilities of the model using the technique described in the following section, we remove such virtual links and distribute the probability on each such virtual link to the corresponding sequence of valid transition links. See the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2011.259, for a detailed elaboration of this technique.

Each packet in the merged packet trace is timestamped with the arrival time at the sniffer along with other information including the id of the sender, size of the packet, and the rate at which it was transmitted. We parse this information in the trace to obtain the sequence of observation symbols for the two senders under consideration. Based on this sequence, we use the following technique to learn the state transition probabilities of the Markov chain, that in turn will provide the probability of interference between the senders.

4.3 Formal Specification and Learning

We now formally describe the HMM using standard notations [27]. The HMM consists of the following:

- Set S of N states, where $N = 11$. S is given by:

$$S = \{S_i\} = \{\langle \mathcal{I}, \mathcal{I} \rangle, \langle \mathcal{B}, \mathcal{I} \rangle, \langle \mathcal{T}, \mathcal{I} \rangle, \langle \mathcal{I}, \mathcal{B} \rangle, \langle \mathcal{I}, \mathcal{T} \rangle, \langle \mathcal{B}, \mathcal{B} \rangle, \langle \mathcal{T}, \mathcal{D} \rangle, \langle \mathcal{T}, \mathcal{B} \rangle, \langle \mathcal{D}, \mathcal{T} \rangle, \langle \mathcal{B}, \mathcal{T} \rangle, \langle \mathcal{T}, \mathcal{T} \rangle\}.$$

7. As a simplification, we develop the model only for the first backoff stage here. This implicitly assumes that retransmissions are rare (which has been true in our experiments). The general approach can be extended to handle multiple backoff stages by observing the number of retransmissions in the trace.

8. This means TCP packets with payload less than 400 bytes in 802.11b.

TABLE 1
Summary of Evaluation Scenarios Used in the Paper to Infer Interference Relations

Sec. #	Scenario	Trace type	Trace source	Nature of traffic	Interference evaluated
5.1	Micro-benchmarking	Experimental	Collected by authors	UDP broadcast (diff. rates)	Sender-side
5.2	NS2 simulations	Simulation	Collected by authors	Short TCP transfers	Sender-side
5.3	Departmental WLAN	Experimental	Collected by authors	Long HTTP downloads	Sender- & receiver-side
5.4	SIGCOMM 2004 trace	Experimental	Externally obtained	Real traffic mix	Sender- & receiver-side

- Set V of M observation symbols, where $M = 7$. V is given by: $V = \{i_s, i_l, x_s, x_l, y_s, y_l, xy\}$.
- Matrix A of state transition probabilities, indicated by $A = [a_{ij}]$, where a_{ij} is the transition probability from state S_i to S_j. This matrix is unknown at the outset and will be determined. Note that some state transitions are invalid and such a_{ij} is set to 0. Such transitions are not shown in Fig. 3.
- Matrix B of observation symbol probabilities, indicated by $B = [b_{jk}]$, where b_{jk} is the probability that the observation symbol is v_k for state S_j. In our case, observation symbols are deterministic for each state. However, they are not unique. The mapping from states to symbols are shown in a table within Fig. 3.
- Vector π of the initial state distribution, indicated by $\pi = [\pi_i]$, where π_i is the probability of initial state being S_i. We use $\pi_i = 1/N$ for all $i, 1 \leq i \leq N$.

The above specification defines the HMM, $\lambda = (A, B, \pi)$. The packet trace provides the observation sequence $O = O_1, O_2, \ldots, O_T$, where each observation $O_t \in V$, and T is the number of observations in the sequence.

Given the above HMM λ and the observation sequence $\mathcal{O}$, our goal is to learn the model parameters $\lambda = (A, B, \pi)$ that maximize $P(O|\lambda)$. This is a difficult problem, and there is no optimal algorithm for it. We can, however, use the expectation-modification (EM) algorithm, which is an iterative method to determine λ, such that $P(O|\lambda)$ is locally maximized. The EM algorithm alternates between an expectation (E) step, which computes the model parameters most likely to produce the observation, and a modification (M) step, which computes the maximum likelihood of model parameters across multiple E steps [28]. We use the well-known *Baum-Welch method*, which is a type of EM algorithm, based on the forward-backward algorithm developed by Baum and Eagon [29]. The method ensures that in every estimation step, we find a model which is more likely to produce the observation. Thus, if we estimate the parameters of the model λ to get $\overline{\lambda}$, then $P(O|\lambda) \geq P(O|\overline{\lambda})$.

While initializing the state transition probabilities in Baum-Welch method, we assign equal probability to all the outgoing valid transitions from each state. This ensures that there is no initial bias in the model toward interfering or noninterfering pair of nodes. This also aids in quick convergence of the method. We deal with the problems of numeric inaccuracies because of continued multiplications of certain small fractions by using the scaling technique in the procedure [31].

Let $\Pi = [\Pi_i]$ be the stationary (steady state) distribution of the states. After learning the transition probabilities $A = [a_{ij}]$, $\Pi = [\Pi_i]$ can be determined as $\Pi = \lim_{n\to\infty} \pi A^n$. The convergence is guaranteed as A is a stochastic matrix.

4.4 Interference Relations

4.4.1 Learning Sender Side Interference

Transitions into any state with a defer component (i.e., states such as $\langle \mathcal{D}, * \rangle$ and $\langle *, \mathcal{D} \rangle$) indicate interference. Similarly, transitions into any state of the set $\{\langle \mathcal{B}, \mathcal{T} \rangle, \langle \mathcal{T}, \mathcal{B} \rangle, \langle \mathcal{T}, \mathcal{T} \rangle\}$ indicate absence of interference. Thus the sender side interference can be interpreted as the total probability of transition into the interfering states. If we represent Π_i's as $P(\mathcal{I}, \mathcal{I})$, $P(\mathcal{B}, \mathcal{I})$, etc, the deferral probability, p_d, is given by

$$\frac{P(\mathcal{D}, \mathcal{T}) + P(\mathcal{T}, \mathcal{D})}{P(\mathcal{D}, \mathcal{T}) + P(\mathcal{T}, \mathcal{D}) + P(\mathcal{B}, \mathcal{T}) + P(\mathcal{T}, \mathcal{B}) + P(\mathcal{T}, \mathcal{T})}.$$

The above expression essentially captures the probability of being in the interfering states when one of the two nodes is transmitting. Here, we are assuming a symmetric link between a node pair. In reality, links may be asymmetric, and the above expression can be easily modified to consider asymmetric deferral probabilities. This is discussed in Section 6.1.

4.4.2 Learning Receiver Side Interference

The receiver-side interference causes collisions that can be detected relatively easily by tracking retransmissions in the trace.[9] One can identify retransmitted packets by observing the set "retransmit bit" in the frame header. A retransmitted frame, say R, can be correlated back to the original frame, say P, that has not been received correctly as both these frames carry the same sequence number. Any frame S from a different sender overlapping with P is a potential cause of collision. If P does not overlap with any other frame, the packet loss is due to wireless channel errors rather than collisions [10], [32]. Because of the probabilistic nature of packet capture, sufficient statistics need to be built up to determine receiver-side interference. This is because frames like S and P—even when overlapping—may not always result in a collision. Thus, the receiver-side interference between two links, or in other words, the probability of collision p_c can be determined as the ratio of the collision count and the overlapped-frame count.

5 Evaluating Interference Relations

We will now evaluate the effectiveness of our approach to infer interference relations by a series of evaluations. We will use a mix of different scenarios starting from careful micro-benchmarking to using large and congested wireless network traces. For the benefit of the reader, we summarize the various scenarios we will use in Table 1.

9. For unicast transmissions only. However, unicasts are much more frequent relative to broadcasts in a real network packet trace.

5.1 Microbenchmark for Sender-Side Interference

We first describe a set of microbenchmarking experiments. Here two senders transmitting broadcast traffic are used to specifically evaluate the sender-side interference using carefully controlled load. We evaluate for a range of interference scenario by positioning the senders at different locations. We also compare our microbenchmarking experiments to infer sender-side interference with two other possible methods described below.

5.1.1 Comparison Points

1) Profile-based method (PROFILE). This technique is specifically based on [14], [15] and needs active measurements. It creates a profile for each device in the network with specific interface card used. Profiling is done by collecting a large number of measurements using a pair of devices to create the correlation between the received signal strength and the probability of deferral. This needs to be repeated for all different cards used in a network. Later the profile can be used to estimate the probability of deferral between two nodes by measuring the average RSS values between them and doing a lookup on the profile. As this technique is expected to be quite accurate, we use this as a benchmark.

2) Moving window based method ($WINDOW(t)$). This is a simple heuristic that may need extensive parameter tuning. In this technique, a moving time window of size t seconds over the combined packet trace is maintained. For each window position, we analyze only the packets inside the window and infer whether the nodes considered interfere or not (see below). Finally, we count the number of window instances where the nodes interfere, and obtain the probability of deferral as a fraction.

Specifically, we use the following approach:

- Only consider windows that have packets from both nodes. (We do not want to consider windows that have mostly one node transmitting and the other silent.)
- Determine the saturation throughput T_{sat}. This is tricky and will depend on the transport protocol and packet sizes used.
- The aggregated throughput T_{obs} of the two nodes in the window being considered is calculated. If $T_{obs} > T_{sat} - \delta_1$, then the window is considered saturated, otherwise the window is considered unsaturated.
- A saturated time window is marked noninterfering if $T_{obs} > T_{sat} + \delta_2$.
- The parameters δ_1 and δ_2 are needed to ride out measurement noises and are tuned.
- Probability of deferral is the fraction of saturated time windows that are marked interfering.

5.1.2 Microbenchmarking with Two Nodes

Our microbenchmark experiment consists of a setup with two senders and two sniffers.

Each sniffer is colocated with a sender to guarantee that all frames are captured. Both the senders and sniffers have 802.11 radios. All the cards used have Atheros chipsets, and the popular MadWiFi driver is used. We also use a "beacon" node, whose sole responsibility is to transmit 802.11 beacons at regular intervals to provide a common time base needed for merging the traces. In a normal deployment, these beacons will be supplied by existing APs.

For the experiments, we configure all the four radios in the same channel. The choice of channel is immaterial. We also set the sender radios in "ad hoc" mode and the sniffer nodes in "monitor" mode. All experiments are done for 802.11b using the PHY-layer data rate of 11 Mbps. A large packet size (1,470 bytes) is chosen for the experiments. This is because, with smaller packets, the sniffers cannot capture all packets in our low-cost embedded hardware, likely due to inefficiencies in interrupt processing. Tcpdump is used for packet capture in the sniffers.

We create a range of interference scenarios by positioning one sender-sniffer pair fixed at one location, and moving the other to various locations in the building. For each scenario, we perform the following measurements. First, we measure the actual probability of deferral between the nodes. To do that, we follow the method in [13] briefly described below. We let each sender, configured with saturated UDP traffic, broadcast in isolation for a minute, and measure their throughputs in isolation. We then let them broadcast together with saturated traffic, and measure their throughputs again. The ratio of the sum of throughputs when the senders broadcast together to the sum of throughputs when the senders broadcast in isolation is defined as BIR, or the broadcast interference ratio [13]. Note $0.5 \leq BIR \leq 1$. The "measured" probability of deferral is estimated as $1/BIR - 1$.

We also collected the RSS measurements at each sender for each scenario when the other sender broadcasts in isolation. This is used to estimate the probability of deferral using the $PROFILE$ method described above. The profiling of each interface card have been independently done using a method similar to [15].

Next, we conduct a series of experiments with varying traffic load in the following fashion for each scenario to evaluate HMM and $WINDOW(t)$ methods. The senders are configured to broadcast UDP packets simultaneously for one minute with 10 different load levels ranging from 0.1 to 6 Mbps. The PHY-layer bit rate is chosen to be 11 Mbps; thus, 6 Mbps for each node means saturated load. Meanwhile, each sniffer captures all the packets it hears in that duration. The packet trace from each sniffer is merged using the techniques described earlier, and this combined trace is used to estimate the probability of deferral using the HMM and the $WINDOW(t)$ methods. The later is repeated for three different window sizes ($t = 0.01\text{ s}, 0.1\text{ s}, 1\text{ s}$).

We make such measurements for 11 different locations of the senders, creating 11 different scenarios. The distribution of the measured probability of deferral at different locations is presented in Fig. 4a. For each scenario, 10 different values of offered load are used between 0.1 and 6 Mbps, thus creating 110 measurements for HMM and the $WINDOW(t)$ methods, and 11 measurements (one for each scenario only) for the $PROFILE$ method. The distribution (CDF) of errors ("estimated"—"measured" probability of deferral) is plotted for all three methods in Fig. 4b. Note that the HMM approach is quite competitive with the $PROFILE$ method. In fact, it is slightly better overall for the particular distribution of deferral probabilities. The

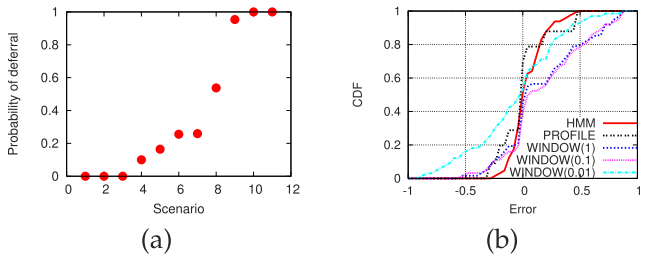

Fig. 4. Combined performance results for 11 chosen scenarios for two node experiments. (a) Measured probability of deferral for different scenarios; (b) CDF of error in estimating probability of deferral.

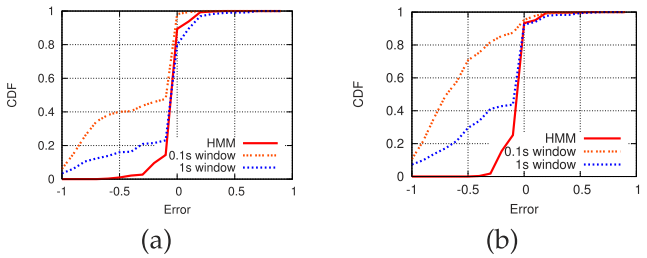

Fig. 5. NS2 simulation results showing CDF of error in deferral probability estimates for the (a) sparse and (b) dense networks.

reason for this is that the $PROFILE$ method uses profiles for interface card models, rather than from the specific cards used in the experiments [15], even though it uses RSS measurements on the actual network with the actual cards used. Variations between individual cards can lead to modeling errors.

The root mean square error (RMSE) values are 0.165 and 0.208 for HMM and $PROFILE$, respectively. The RMSE values for $WINDOW(t)$ methods is 0.385, 0.408, and 0.402 for $t = 0.01\text{ s}, 0.1\text{ s}$, and 1 s, respectively. We have noted before, however, that the $PROFILE$ method is impractical for analyzing live network traffic and it also requires access to the network nodes.

Overall, HMM is quite competitive with $PROFILE$, but requires only passive measurements. The experience with the window-based method is quite variable. It is also quite sensitive to choice of window size.

5.2 Simulation-Based Evaluation

Simulations let us create arbitrary topologies and interference conditions easily. However, the physical layer (including interface behavior for carrier sense and packet capture) implementation is often idealized or unrealistic in simulations. To address this issue, we use an extended version of the ns2 simulator that includes realistic measurement-based models [33]. These models were validated against experimental results showing excellent accuracy [33].

For the sake of completeness, we note that the enhancements in ns2 in [33] are done specifically in the following physical layer components—1) radio propagation model, 2) deferral or carrier sense model, and 3) packet reception model. For (1), models are derived from real measurements in a testbed. For (2) and (3), measurement-based profiles of a testbed are created where every value of RSS is mapped to a deferral probability and every value of SNR is mapped to receive probability, respectively. These profiles make the interference relations between links nonbinary.

We consider two scenarios, where 20 nodes are uniformly and randomly distributed in a 200×200 m area and a 100×100 m area. These two scenarios produce different topologies: sparse and dense. We generate traffic by creating one-hop TCP flows on randomly chosen feasible links. Both the interarrival time and duration of flows are chosen from an exponential distribution. For the results presented here (See Fig. 5), the simulations were run for 180 s, the average duration of each flow was 5 s, and the average interarrival time between flows was varied from 2.5 to 1 s, such that the average load in the network varies from 2 to 5 flows.

In Fig. 5, we show the CDF of the estimation error (as before) for the probability of deferral between node pairs. CDFs for both HMM and $WINDOW(t)$ methods are presented for the sparse and dense network. The $PROFILE$ approach is not shown here as it would be perfectly accurate in the simulator (as the simulator's deferral model itself uses the same profile model). From the plots note that HMM performs significantly better than the window-based method. Average RMSE value for the HMM method is about 0.1, while the average RMSE value for the better of the two window-based methods is about 0.4. Note again the accuracy of the window-based method is quite sensitive to window sizes.

5.3 Complete Evaluation on WLAN

Here, we provide a complete evaluation—both sender and receiver sides. These experiments are done on an active WLAN with seven APs spread over two floors of the Computer Science department building of Stony Brook University. Seven laptops are used as clients. Each client fetches a large file via HTTP download using a unicast link for about 20 mins. This simulates real network traffic that is sniffed using nine sniffers (Soekris single board computers with 802.11 miniPCI cards with Atheros chipset and with external USB flash memory to store packet traces). The sniffers are deployed based on convenience, i.e., near a power outlet and in the rooms that we have regular access to. However, an attempt was made to keep them as close to the APs as possible.

Sixteen client laptop pairs are considered for evaluation. All of these pairs associate with two different APs. Unlike the microbenchmarking experiments, the default autorate control with 802.11b is used. Also, the 802.11 frames are now unicast with ACK. RTS/CTS is disabled. For each pair, the probability of interference between the pair of download links (AP to client) is "estimated" using (1). First, the probability of deferral (p_d) is estimated using the HMM-based method using the merged sniffed traffic traces from all sniffers. Second, the probability of collisions (p_c) is estimated by observing the retransmissions for overlapped packets as described in Section 4.4.2. However, in all cases, retransmissions were rare, typically less than 1 percent of frames were retransmitted. This is consistent with prior experimental observations [10]. Thus, p_c could be safely ignored with p_d alone determining the probability of interference.

For validation, p_d is "measured" via the BIR method described in the previous section. For these measurements, simultaneous saturated UDP traffics on the downlinks are used for about 2 mins. The validation results are shown in

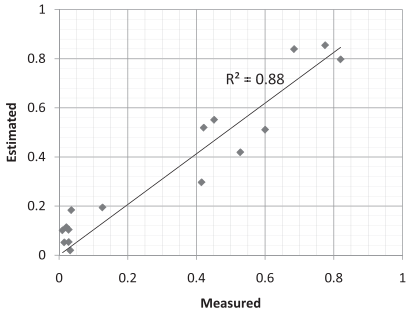

Fig. 6. Estimated and measured probabilities of deferral for the 16 test cases with the departmental WLAN.

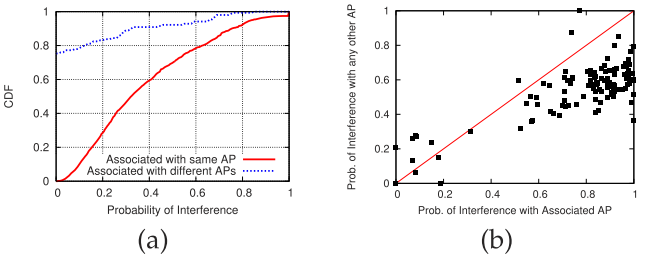

Fig. 7. Interference analysis of the SIGCOMM 2004 trace: (a) CDF of probability of interference between clients associated with the same AP and different APs; (b) comparison of interference between the associated AP and another AP.

Fig. 6 as a scatterplot. Note the high degree of predictability of the estimation in this real-life experiment. The straight line is the least square fit with the condition that the line passes through 0. Note that it is very close to the $y = x$ line. The R^2 value for this line is 0.88 showing a good fit.

A careful reader will notice a slight bias at the low end of the deferral probabilities. The HMM method consistently overestimates deferral probability, when the probability is very small. We have also observed this in our microbenchmarking though it does not show up in the CDF plots. The reason for this is the heuristic we used in our modeling (Section 4.2) that defer/idle periods are always assumed longer than 31 slots. When there is little interference, often idle periods could be shorter than backoffs. If they are misclassified as backoffs, the possibility of misclassifying some idle states as defer increases. As discussed in Section 4.2, a stronger heuristic using more accurate clocks could address this issue.

5.4 Using Large-Scale Wireless Traces

Encouraged by the strong validation results in the departmental WLAN trace analysis, we use the wireless network trace collected at the SIGCOMM 2004 conference [10] for demonstrating powerful capabilities of our tool. The trace was obtained from the CRAWDAD archive [34]. The SIGCOMM 2004 conference was four days long and was attended by more than 500 attendees. During busy periods, several simultaneously active flows were not uncommon [10]. The WLAN under consideration in this trace had five APs—three on channel 1, one on channel 8 and the other one on channel 11. Five sniffers were used each with three wireless interfaces. Two of them listened on channel 1 and 11, respectively, and the third one listened either on channel 8 or 6 [10]. We consider only channel 1 in this work.

First, we analyze the probability of interference between client-to-AP links where the clients are associated with the same AP. For this analysis, we pick random pairs of clients associated with the same AP and find a 20 mins long period when they are both simultaneously active. In Fig. 7a, we plot the CDF of the probability of interference for 1,990 such randomly chosen link pairs. This shows that the interference is well-distributed over the entire range showing roughly similar probabilities of (mostly) interfering clients and (mostly) noninterfering clients. This indicates that a significant number of "hidden" clients associate with the same AP. However, these hidden clients almost never collide. Collision probability is found to be minuscule (less than 0.4 percent). Thus, probability of interference is again controlled by the deferral probability alone.

Next, we do a similar analysis but for pairs of clients that associate with different APs. This study is exhaustive instead of a random sampling as the number of such pairs is relatively small (154). In Fig. 7a, note that almost 75 percent of such client pairs do not interfere at all and about 5 percent interfere strongly. The rest are in between. This indicates that the association control works quite well. This point is further elaborated in Fig. 7b where we show a comparison of the deferral probability of 120 randomly selected clients with its associated AP and with another random AP. In a good deployment we would normally expect the latter to be small and the former to be much higher than the latter. However, we see that while the interference with the associated AP is higher about 90 percent of the cases (indicating a good association control), the other AP often presents significant interference. This can indicate, for example, a poor channel assignment.

6 Detecting Selfish Behavior

In this section, we demonstrate how the interference relationship can be used to detect selfish carrier-sense behavior and define a metric to quantize the selfishness of a node. We also define the characteristic of an effective witness and introduce two simple heuristics to identify effective witnesses.

6.1 Detecting Asymmetric Behavior

To detect selfish carrier-sense behavior, we need to identify asymmetric behavior. This can be detected using the following fashion. The probability that X has a packet to transmit and it defers while Y transmits is given by

$$P_{\text{def}}(X,Y) = \frac{P(\mathcal{D},\mathcal{T})}{P(\mathcal{D},\mathcal{T}) + P(\mathcal{B},\mathcal{T}) + P(\mathcal{T},\mathcal{T})}.$$

The opposite probability (i.e., Y has a packet to transmit and it defers while X transmits) is likewise

$$P_{\text{def}}(Y,X) = \frac{P(\mathcal{T},\mathcal{D})}{P(\mathcal{T},\mathcal{D}) + P(\mathcal{T},\mathcal{B}) + P(\mathcal{T},\mathcal{T})}.$$

The difference between $P_{\text{def}}(X,Y)$ and $P_{\text{def}}(Y,X)$ characterizes asymmetry. Larger the difference, higher is the asymmetry. Due to the nature of our approach, the asymmetry is tested between a node pair at a time. A positive (negative) difference indicates that Y (X) gets a

bandwidth advantage due to asymmetric carrier sensing. In our evaluation, we have used the difference with a simple normalization as the "metric of asymmetry," $\eta(X,Y)$, except when the two probabilities are both close to zero. Thus, when both $P_{\text{def}}(X,Y)$ and $P_{\text{def}}(Y,X) < \epsilon$ (ϵ was chosen to 0.01 in the evaluations), the metric of asymmetry, $\eta(X,Y)$, is given by

$$P_{\text{def}}(Y,X) - P_{\text{def}}(X,Y),$$

else it is given by

$$\frac{P_{\text{def}}(Y,X) - P_{\text{def}}(X,Y)}{\max(P_{\text{def}}(Y,X), P_{\text{def}}(X,Y))}.$$

Note that $\eta(X,Y) = -\eta(Y,X)$.

6.2 Selecting Witnesses

In general, each network node X must be evaluated for selfish behavior. By default, every other node Y acts as a witness and the above metric of asymmetry is evaluated for the pair (X,Y). Thus, for each network node X, we take the *average of the metric of asymmetry $\eta(X,Y)$ over all the witnesses Y that provide a positive value*. The negative values are discounted as they will be accounted when Y is evaluated with X as the witness. We call this average the "selfishness metric." We will evaluate this metric later in our simulations.

However, if X and Y are not within carrier sense range of each other (i.e., they never hear each other), Y cannot serve as an effective witness. This is because $P(\mathcal{D},\mathcal{T})$ or $P(\mathcal{T},\mathcal{D})$ would evaluate to zero. (In practice, due to measurement noise, they evaluate to a very small value close to zero.) Thus, the metric of asymmetry is zero. While this is correct, this does present a problem. Assume that X is indeed selfish in a 4 node network and witness Y_1 detects a very large (i.e., $\eta(X,Y_1)$ is close to 1) metric of asymmetry. However, witnesses Y_2 and Y_3 do not hear X at all (and vice versa). They offer the metric ($\eta(X,Y_2)$ and $\eta(X,Y_3)$) as close to 0. Here, witness Y_1 is an effective witness while witness Y_2 and Y_3 are ineffective witnesses. Without any further information, if we aggregate these measures using an average, we obtain a low confidence in X's selfishness (about 0.3 in this example), even when we have one perfect witness and the other witnesses are clearly ineffective. On the other hand, relying on a single witness (e.g., Y_1) that points to a severe asymmetry may not be right as this may simply be due to random wireless channel/interface effects and not due to a systematic selfish behavior. Thus, this can raise false alarms.

This problem cannot be addressed without some additional knowledge of the network topology regarding which node can serve as an effective witness. Ideally, we should only rely on witnesses that are within the carrier sensing range from a potential selfish node. The more such nodes, the better.

To address this issue, we use two simple heuristics named as H_1 and H_2. For heuristic H_1, we assume that the sniffer locations are known, as well as some bounds on the carrier sense range (R_C) and transmit range (R_T) for the network nodes. Then the sniffers that are separated by at least $R_C + 2R_T$ distance, must sniff nodes that cannot hear each other. Thus in other words, for a node X sniffed by a sniffer S_X[10] and another node Y sniffed by a sniffer S_Y, node Y will not be an effective witness of node X if S_X and S_Y are separated by at least $R_C + 2R_T$ distance. This simple heuristic eliminates many nodes that should not serve as witness to each other. Note that this may not remove all ineffective witnesses, and if the bounds are incorrect, this technique may even remove some effective witnesses. But this technique is practical and easy to use, and at minimum eliminates a large number of far-away witnesses that cannot be effective by being outside the carrier-sense range.

For heuristic H_2, we do not even need to assume anything. In H_2, Y is an effective witness of X, if they are both sniffed by a common sniffer. H_2 will surely remove all the ineffective witnesses, and may also remove some effective witnesses.

For any given heuristic, for each network node X we take the average of the metric of asymmetry $\eta(X,Y)$ over all the nodes Y that are selected as effective witnesses by the heuristic and that provide a positive value for η. Then we calculate the "selfishness metric" by a simple averaging.

7 Evaluating Selfish Carrier-Sense Detection

In this section, we evaluate our technique to detect selfish carrier-sense behavior. We have performed two sets of evaluations: 1) a set of microbenchmarking experiments to understand the effectiveness of the approach and 2) a set of ns2 simulations to study larger networks and complex selfish behaviors.

7.1 Experiments

The experiments essentially achieve careful microbenchmarking using similar setup described in Section 5.1.2. Only two network links are used but wireless channel quality, traffic load, and selfish behaviors are varied over a wide range. One transmitter is configured as "selfish"; the other transmitter is regular and acts as the sole "witness." A sniffer node, located in close proximity of each transmitter, monitors the traffic on corresponding link. In this experiment we use 802.11a and channel 52 with 6 Mbps PHY layer rate and a large packet size (1,470 bytes). We use Soekris boards as the transmitters and laptops running linux as sniffers.

A node achieves selfishness by not sensing carrier before transmitting. To make a node selfish, we have used the antenna switching technique described in [35]. There are two antenna connectors on 802.11 interface for diversity where either of them can be selected for receiving/transmitting using driver-level command. We have connected one antenna to one connector, kept the other connector unconnected. Selecting the unconnected antenna as the receiving antenna effectively disables carrier sense.[11] The impact of the selfish behavior can be varied by simply varying the distance between the selfish and witness nodes. A close distance means the witness node is

10. We say a node is sniffed by a sniffer, when the packets transmitted from the node can be heard by the sniffer.

11. Note that selfishness can also be achieved by resetting the CCA threshold as in [11]. However, in our hardware we have found that the antenna switching technique is more foolproof than using an increased CCA threshold.

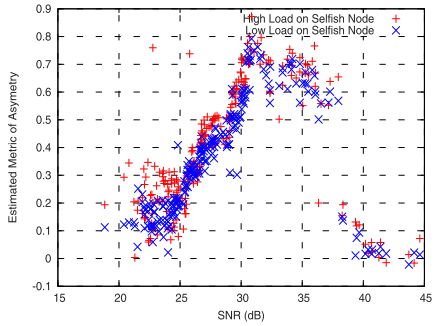

Fig. 8. Experimental results with varying load on the selfish node.

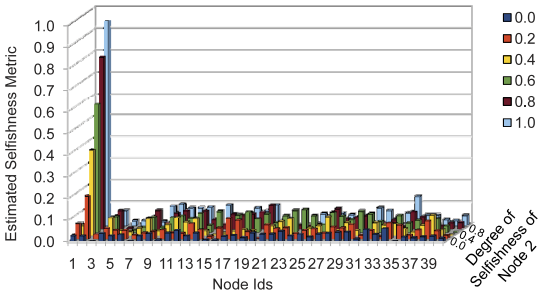

Fig. 9. Simulation results for a 40 node network. Node 2 is the only selfish node. The estimated selfishness metric using heuristic H_2 is shown for each node for each of the 6 sets of simulations that are run with different degree of selfishness of node 2.

impacted significantly due the selfish behavior as the RSS at the witness node is high. A large distance means that RSS is low and often the witness node cannot hear the selfish node due to channel fading, and thus the selfishness causes little impact.

The benchmarking experiments are performed by increasing the distance between the two transmitters (selfish and witness) from a very small value at steps of 3 ft in 28 discrete steps. For each position, 1) the average SNR from the selfish to the witness transmitter is measured, and 2) UDP packets are transmitted at different offered loads on their respective links for 60 s. We use offered loads of 6 and 4 Mbps, denoting high and low loads, respectively. We experiment with both loads on the selfish node, while the witness node has only high load.

Fig. 8 plots the estimated metric of asymmetry η for the <selfish, witness> node pair for each of the experiments. The plots are color-coded based on the load. The asymmetry is clearly higher with higher SNR. Note that with lower load on the selfish node the asymmetry tends to be somewhat lower as expected. Also, note significantly lower asymmetry when the SNR is very high (i.e., nodes are very close). This is an artifact of our experimental technique. The selfish node starts picking up some signal at close ranges even when the antenna is disconnected, and thus it stops being selfish. So, much lower asymmetry is detected for very high SNRs.

Note that the above two node microbenchmarking is sufficient to derive an insight into what would happen in a multiple node network. Essentially, nodes still need to be evaluated in a pairwise fashion. For each potential selfish node, we need to evaluate the metric of asymmetry with each possible witness node independently. Note again (as discussed in Section 3), we are currently considering pairwise interference only. But several other issues remain to be evaluated—1) how to effectively combine the metric of asymmetry for a selfish node as provided by multiple witness nodes into a single measure, defined as "selfishness metric" in Section 6.2, 2) how suitable are the witness nodes. We will explore these issues via a packet-level simulation using the ns2 simulator.

7.2 Simulations

Ns2 simulations let us implement various degrees of selfishness, where the selfish node senses carrier with only a certain probability. We use the term *degree of selfishness* (P_s) to indicate that the selfish node senses carrier with probability equal to $1 - P_s$. Ns2 simulations also make it easier to investigate larger networks, where there are many nodes, possibly with more than one selfish node with varying traffic and degrees of selfishness.

In our simulated scenario, there are 40 network nodes distributed randomly in a square region. We chose a deployment typical of dense WiFi client distribution in indoor office environments, assuming that there is one node in 300 sq ft on average. The default ns2 wireless channel model is extended to include *shadowing* [36] effects. This introduces randomness in the transmission range of a node instead of making it a perfect disk. Shadowing parameters are taken from [33] where a set of measurements was done to model such parameters in an indoor environment. A set of feasible network links are chosen randomly and one-hop UDP flows are generated with randomly chosen loads (between 0.5-1 Mbps). Each flow is active (and then inactive) only for a random interval of time. Both intervals are chosen from an exponential distribution with a mean of 5 s. Note that the exact traffic parameters are not important for our work. All that is important is that *enough traffic is recorded so that for each pair of nodes that are potentially within the carrier sense range there are concurrent packet transmission attempts*. This ensures that any possible selfish node will find enough witnesses.

We deploy a set of 10 sniffers at random locations. Among the 40 network nodes, 1, 2, or 3 nodes are selfish. The degree of selfishness is varied. For each pair of nodes, we evaluate the metric of asymmetry by using the procedure in Section 4. For each network node X, we measure the selfishness metric in three ways as discussed in Section 6.2: 1) using all possible witness nodes (also called "no heuristic" case), 2) using witness nodes based on heuristic H_1, and 3) based on heuristic H_2.

Fig. 9 plots the selfishness metric of each node in the scenario with one selfish node with varying degree of selfishness where the witness nodes are selected using heuristic H_2. Note that the metric has a very visible peak only for the selfish node. The values of metric for the selfish nodes are roughly similar to the degree of selfishness.

Because of space limitation we do not present the similar plots for the scenarios with 2 and 3 selfish nodes using different heuristics. We instead show the overall statistics that summarizes how good our detection is. For each scenario and for each type of witness node identification technique, we evaluate for each node the "estimation error" as the algebraic difference between the *computed*

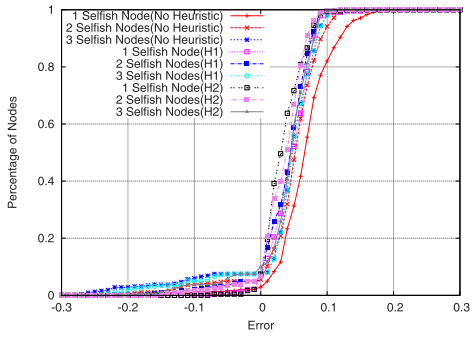

Fig. 10. CDF of "estimation error" for the selfishness metric. Three different scenarios are presented where number of selfish nodes are varied (1, 2, or 3) and witness nodes are identified in three different ways.

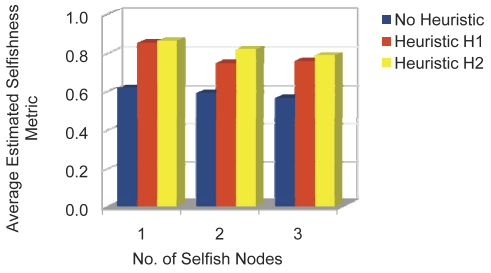

Fig. 11. Simulation results for the sparse network.

selfishness metric and the *actual degree of selfishness* of that node. All nodes (selfish and regular) are included. The estimation error is plotted as a CDF in Fig. 10. Nine plots are shown for three techniques used to identify the witness nodes and for three different numbers of selfish nodes. The CDF shows that the estimation error is very small in general and heuristic H_2 performs somewhat better than the other two techniques in general.

In this scenario, the heuristics do not perform much better than the no heuristic case, because the no heuristic case itself performs very well. The reason for this is the high density of the network. To demonstrate the power of the heuristics we consider a sparser network with 40 nodes distributed randomly in squared region with one node in 1,500 sq. feet on average. Different scenarios are created by varying the number of selfish nodes (1, 2, or 3) with degree of selfishness $= 1$. Because of the sparsity of the network we now have to deploy more sniffers to capture all network traffic. So, this time we deploy 40 sniffers randomly as before. Fig. 11 shows the average estimated selfishness metric measured in three ways as before only for the selfish node(s). Note that as expected 1) estimation becomes better when we identify witness nodes using the heuristics in comparison to using all the nodes as witnesses; 2) H_2 is generally a better heuristic, and 3) estimation becomes worse with a larger number of selfish nodes. The reason for H_2 performing better is that it only considers effective witnesses, while H_1 may include ineffective witnesses as well. The reason for the third observation is that selfish nodes cannot be used to correctly identify other similarly selfish nodes.

8 Conclusions

We have investigated a novel machine learning-based approach to estimate interference and to detect selfish carrier-sense behavior in an 802.11 network. The technique uses a merged packet trace collected via distributed sniffing. It then recreates the MAC layer interactions on the sender-side between network nodes via a machine learning approach using the Hidden Markov Model. This coupled with an estimation of collision probability on the receiver-side is helpful in inferring the probability of interference in the network links. Significant asymmetry in the sender-side interaction in favor of a particular node witnessed by multiple other nodes indicates selfishness. The power of this technique is that it is purely passive and does not require any access to the network nodes. Although our technique works offline, it can be used periodically every few minutes (for example). Moreover, interference relationship can be used for efficient network design and capacity allocation. It can be used as a third-party solution for detecting MAC-layer misbehavior in 802.11 networks. Evaluations show the effectiveness of the tool for both the applications.

There are indeed some limitations of the technique as presented here. So far, we have estimated deferral behavior assuming only pairwise interference and have ignored physical interference (see discussions in Section 3.1) arguing that the improvement in accuracy will be relatively minor. Also, 802.11 retransmissions were ignored in the modeling to reduce complexity. These are not fundamental limitations and can be accommodated with higher computational cost, but are likely unnecessary. So long as enough of the common baseline case that we modeled indeed show up in the traffic trace, we will have a very good estimation accuracy. Our future work will include more evaluations to demonstrate this aspect. We will also study the impact of inaccuracy in trace gathering.

References

[1] A.P. Jardosh, K.N. Ramachandran, K.C. Almeroth, and E.M. Belding-Royer, "Understanding Congestion in IEEE 802.11b Wireless Networks," *Proc. ACM SIGCOMM*, 2005.

[2] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan, "Measurement-Based Characterization of 802.11 in a Hotspot Setting," *Proc. ACM SIGCOMM*, 2005.

[3] A. Kashyap, U. Paul, and S.R. Das, "Deconstructing Interference Relations in WiFi Networks," *Proc. IEEE Seventh Comm. Soc. Conf. Sensor Mesh and Ad Hoc Comm. and Networks (SECON)*, 2010.

[4] U. Paul, S.R. Das, and R. Maheshwari, "Detecting Selfish Carrier-Sense Behavior in Wifi Networks by Passive Monitoring," *Proc. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN)*, 2010.

[5] "AirMagnet WiFi Analyzer," http://www.airmagnet.com/products/wifi_analyzer, 2012.

[6] "AirPatrol's Wireless Threat Management Solutions," http://www.airpatrolcorp.com, 2012.

[7] P. Bahl et al., "DAIR: A Framework for Troubleshooting Enterprise Wireless Networks Using Desktop Infrastructure," *Proc. ACM HotNets-IV*, 2005.

[8] P. Bahl et al., "Enhancing the Security of Corporate Wi-Fi Networks Using DAIR," *Proc. ACM/USENIX Mobile Systems, Applications, and Services (MobiSys)*, 2006.

[9] Y.-C. Cheng, J. Bellardo, P. Benkö, A.C. Snoeren, G.M. Voelker, and S. Savage, "Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis," *Proc. ACM SIGCOMM*, 2006.

[10] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analyzing the MAC-Level Behavior of Wireless Networks in the Wild," *Proc. ACM SIGCOMM*, 2006.

[11] K. Pelechrinis, G. Yan, S. Eidenbenz, and S.V. Krishnamurthy, "Detecting Selfish Exploitation of Carrier Sensing in 802.11 Networks," *Proc. IEEE INFOCOM,* 2009.

[12] J. Yeo, M. Youssef, and A. Agrawala, "A Framework for Wireless Lan Monitoring and its Applications," *Proc. Third ACM Workshop Wireless Security (WiSe),* 2004.

[13] J. Padhye, S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill, "Estimation of Link Interference in Static Multi-Hop Wireless Networks," *Proc. Internet Measurement Conf. (IMC),* 2005.

[14] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-Based Models of Delivery and Interference in Static Wireless Networks," *Proc. ACM SIGCOMM,* 2006.

[15] A. Kashyap, S. Ganguly, and S.R. Das, "A Measurement-Based Approach to Modeling Link Capacity in 802.11-Based Wireless Networks," *Proc. ACM MobiCom,* 2007.

[16] L. Qiu, Y. Zhang, F. Wang, M.K. Han, and R. Mahajan, "A General Model of Wireless Interference," *Proc. ACM MobiCom,* 2007.

[17] K. Jamieson, B. Hull, A.K. Miu, and H. Balakrishnan, "Understanding the Real-World Performance of Carrier Sense," *Proc. ACM SIGCOMM Workshop Experimental Approaches to Wireless Network Design and Analysis (E-WIND),* Aug. 2005.

[18] H. Chang, V. Misra, and D. Rubenstein, "A General Model and Analysis of Physical Layer Capture in 802.11 Networks," *Proc. IEEE INFOCOM,* 2006.

[19] S. Das, D. Koutsonikolas, Y. Hu, and D. Peroulis, "Characterizing Multi-Way Interference in Wireless Mesh Networks," *Proc. First Int'l Workshop Wireless Network Testbeds, Experimental Evaluation and Characterization (WINTECH),* 2005.

[20] E. Magistretti, O. Gurewitz, and E. Knightly, "Inferring and Mitigating a Link's Hindering Transmissions in Managed 802.11 Wireless Networks," *Proc. ACM MobiCom,* 2010.

[21] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On Selfish Behavior in CSMA/CA Networks," *Proc. IEEE INFOCOM,* 2005.

[22] S. Radosavac, J.S. Baras, and I. Koutsopoulos, "A Framework for Mac Protocol Misbehavior Detection in Wireless," *Proc. ACM Workshop Wireless Security,* 2005.

[23] J. Tang, Y. Cheng, Y. Hao, and C. Zhou, "Real-Time Detection of Selfish Behavior in IEEE 802.11 Wireless Networks," *Proc. IEEE 72nd Vehicular Technology Conf. Fall (VTC-Fall),* 2010.

[24] P. Kyasanur and N. Vaidya, "Detection and Handling of Mac Layer Misbehavior in Wireless Networks," *Proc. IEEE Int'l Conf. Dependable Systems and Networks (DSN),* 2003.

[25] M. Raya, J.-P. Hubaux, and I. Aad, "Domino: A System to Detect Greedy Behavior in IEEE 802.11 Hotspots," *Proc. ACM Second Int'l Conf. Mobile Systems, Applications, and Services (MobiSys),* 2004.

[26] P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Trans. Information Theory,* vol. 46, no. 2, pp. 388-404, Mar. 2000.

[27] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Readings in Speech Recognition,* pp. 267-296, Morgan Kaufmann, 1990.

[28] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. Series B (Methodological),* vol. 39, no. 1, pp. 1-38, 1977.

[29] L.E. Baum and J.A. Eagon, "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology," *Bull. Am. Math. Soc.,* vol. 73, pp. 360-363, 1967.

[30] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 3, pp. 535-547, 2000.

[31] S.E. Levinson, L.R. Rabiner, and M.M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *Bell System Technical J.,* vol. 62, no. 4, pp. 1035-1074, 1983.

[32] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing Wireless Packet Losses in 802.11: Separating Collision from Weak Signal," *Proc. IEEE INFOCOM,* 2008.

[33] A. Kashyap, S.R. Das, and S. Ganguly, "Measurement-Based Approaches for Accurate Simulation of 802.11-Based Wireless Networks," *Proc. ACM 11th Int'l Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM),* 2008.

[34] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, J. Zahorjan, and E. Lazowska, "CRAWDAD Data Set uw/sigcomm2004," http://crawdad.cs.dartmouth.edu/uw/sigcomm2004, 2012.

[35] K. Chebrolu, B. Raman, and S. Sen, "Long-Distance 802.11b Links: Performance Measurements and Experience," *Proc. ACM MobiCom,* 2006.

[36] T.S. Rappaport, *Wireless Comm.: Principles and Practice.* IEEE Press, 1996.

Utpal Paul received the BSc and MSc degrees in computer science and engineering from the Bangladesh University of Engineering and Technology in 2004 and 2007, respectively. He is working toward the PhD degree in the Computer Science Department of Stony Brook University. His research interests include the areas of protocols, systems, and analysis of wireless networks with a focus on MAC layer issues in WiFi, mesh, and sensor networks.

Anand Kashyap received the BTech degree in computer science and engineering from IIT Kanpur and the PhD degree in computer science from Stony Brook University in 2002 and 2008, respectively. He is a researcher in the Core Research Group of Symantec Research Labs in Mountain View, California. Currently, he does research in computer networking, mobile computing, and the application of machine learning to these fields.

Ritesh Maheshwari received the BTech degree in computer science and engineering from IIT Kharagpur, India, and the PhD degree in computer science from Stony Brook University in 2009. Currently, he is working as a senior performance engineer at Akamai Technologies, Cambridge, Massachusetts. His research interests include the general area of computer networking.

Samir R. Das received the PhD degree in computer science from Georgia Tech in 1994. He is a professor in the Computer Science Department at Stony Brook University. His research interests include wireless networking, mobile computing, and performance evaluation. More information about his research activities can be obtained from http://www.cs.stonybrook.edu/~samir.