Detecting Selfish Carrier-Sense Behavior in WiFi Networks by Passive Monitoring

Utpal Paul, Samir R. Das
Computer Science Department
Stony Brook University
Stony Brook, NY 11794-4400, USA
{upaul, samir}@cs.sunysb.edu

Ritesh Maheshwari
Akamai Technologies
8 Cambridge Center
Cambridge, MA 02142, USA
rmaheshw@akamai.com

Abstract

With the advent of programmability in radios, it is becoming easier for wireless network nodes to cheat to obtain an unfair share of the bandwidth. In this work we study the widely used 802.11 protocol and present a solution to detect selfish carrier-sensing behavior where a node raises the CCA (clear channel assessment) threshold for carrier-sensing, or simply does not sense carrier (possibly randomly to avoid detection). Our approach is based on detecting any asymmetry in carrier-sense behavior between node pairs and finding multiple such witnesses to raise confidence. The approach is completely passive. It requires deploying multiple sniffers across the network to capture wireless traffic traces. These traces are then analyzed by using a machine learning approach to infer carrier-sense relationships between network nodes. Evaluations using a real testbed as well as ns2 simulation studies demonstrate excellent detection ability. The metric of selfishness used to estimate selfish behavior matches closely with actual degree of selfishness observed.

Keywords: 802.11 protocol, Hidden Markov Model, MAC layer misbehavior.

1 Introduction

With the advent of programmable radios, different MAC protocol parameters can be manipulated in various ways to gain an unfair share of the available wireless bandwidth. The case in point is widely used 802.11 networks. Several radio interfaces and corresponding device drivers are available that let the user choose the clear channel assessment (CCA) threshold and/or the backoff window size [17]. Reseting any one of these appropriately can deliver an unfair bandwidth advantage to a selfish node [17] or even launch a denial of service attack. For example, a higher CCA threshold can effectively disable carrier sensing. Thus, the selfish node gains more transmission opportunities. This can also cause collisions and thereby force the other transmitters in the vicinity to 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 with disabled carrier-sensing. The authors in [17] demonstrate with extensive experiments how a selfish node with higher CCA threshold can gain a significant throughput benefit for different Transport layer protocol. They also show that this kind of selfish node is able to gain at least 5Mbps throughput gain from its fair share [17]. Similarly, setting the backoff window smaller provides an unfair advantage by backing off for a shorter interval on average.

The situation is expected only to get worse with use of more sophisticated radios (e.g., software radios), where any part of the protocol can be easily manipulated or entirely new protocols are used to gain bandwidth advantage. This behooves the research community to develop techniques so that such selfish behavior can be detected. While there are several efforts to develop spectrum etiquette rules for use by unlicensed devices [23] (for accessing fallow spectrum such as DTV whitespace, e.g.), there are nearly not enough effort to 'police' the spectrum to ensure that all radio devices follow a prescribed protocol or rule.

A general solution of 'policing' problem is outside the scope of this paper. We specifically focus on 802.11 networks, where available commodity hardware and software easily allow for selfish behavior as mentioned before. Our goal here is to detect such selfish behaviors via passive monitoring. The approach is equally applicable to WLANs and mesh networks, regardless of the topology or architecture and can detect misbehavior on the part of any network node, be it a client or access point (AP).

Looking at the literature, we find that there are several robust approaches that can detect manipulation of the backoff mechanism (see, e.g., DOMINO [21] and the discussions in

the related work section (Section 2)). However, detecting manipulation of the CCA threshold or completely disabling carrier sensing is a much harder problem. In our knowledge, the latter has been been addressed in only one recent paper [17] that provides a limited solution. Thus, in this work we only address the carrier sensing issue.

Our general approach is as follows. A distributed set of 'sniffers' collect traffic traces from the live network. These sniffers do not transmit any packets making the method completely unobtrusive. The traffic traces are then merged [25, 16, 8] and analyzed using a machine learning technique to determine *asymmetric carrier-sensing behavio*r between network nodes. This means that between a given pair of nodes, one is sensing the other node's carrier, but not vice versa. *Our general assumption is that significant asymmetry in favor of a specific node that persists long-term – when witnessed by multiple other nodes – points to selfish behavior.* This is because such asymmetry may be very unusual due to normal wireless channel/interface effects. Our technique is general and can detect selfish behavior on the part of multiple nodes in the network. While the technique is off-line, it can run on the background periodically to detect selfish behavior.

Since the approach is completely passive, it is dependent on the sufficiency of the available network traffic for analyzing carrier-sense behavior. The challenge in this case is to make accurate identification even (i) in presence of little traffic, and traffic of unknown and arbitrary nature, or (ii) for selfish node implementations that exhibit probabilistic behaviors to avoid detection. There are indeed many other issues related to the location of the sniffers and fidelity of the merged traces that will impact the accuracy of the technique to a varying degree. However, these are independent issues and have been discussed in related literature. We will not have chance to address these issues in the current paper.

The rest of the paper is organized as follows. We will discuss related work in Section II and the broad approach in Section III. The details of the HMM-based formulation will be covered in Section IV. Sections V has the experimental and simulation-based evaluations. We will conclude in Section VI.

2 Related Work

2.1 Detecting MAC-Layer Misbehavior in 802.11

Much of the work in literature only attempts to detect the manipulation of the backoff behavior in 802.11. We summarize them in the following.

Kyasanur and Vaidya [14] propose a mechanism where the receiver directly specify the backoff value to the sender to restrict the sender from being selfish. Cagalj *et al.* [5] develop a distributed protocol using a game theoretic approach that leads the selfish nodes in the network to a Pareto-optimal Nash equillibrium. Konoroski [13] also proposes a scheme to detect the deviation from the ideal backoff mechanism. Radosavac [19] uses a technique based on Sequential Probability Ratio Test (SPRT) to identify the same kind of misbehavior. All these above mentioned studies only deal with the selfishness of a node doing only by backoff manipulation. Raya *et al.* [21] propose and implement DOMINO which can detect certain misbehaviors from greedy stations. DOMINO can detect nodes that try to get larger share of the bandwidth by sending 'scrambled frames,' or using a smaller DIFS period before sending DATA packets, or using oversized NAV to have the medium idle for a longer period of time. It also can detect backoff manipulation by the greedy nodes. But DOMINO cannot detect any misbehavior by a node regarding the carrier sensing, that is, it cannot say whether a node is overhearing the carrier intentionally.

The above approaches detect manipulation of the backoff behavior. They can be complementary to our work. Detecting manipulation of the carrier-sense behavior, however, is a considerably harder problem. The reason for this is that it is quite possible that a node may simply fail to detect ongoing transmissions in the neighborhood due to normal wireless channel effects. For example, the received power may simply be below the normal CCA threshold. Thus, identifying abnormal behavior may be hard. In our knowledge, only one paper [17] has addressed this issue. The authors here make the 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 technique makes a strong assumption 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.[1] Also, depending on how the radio transceiver is designed, packet reception success may not be dependent of the CCA threshold. For example, in a software radio implementation, the selfish node may simply turn off carrier sensing all together or senses carrier probabilistically to avoid detection. In addition the proposed technique is not passive. It requires transmission of probes by the APs. This may interfere with normal network traffic.

[1] There may indeed be a latency issue that can slow down the selfish node if such changes are frequent. For example, a register write to the interface card must happen, or a call to the firmware using an API must be made, etc. But we do not consider this to be an fundamental issue. With the increasing efficiency of the radio hardware the latency may not be any serious issue. Also, there is no reason for any latency if the MAC protocol is implemented in a software radio platform.

2.2 Use of Distributed Sniffers

Previous studies have also used distributed sniffers to conduct a range of measurements over live networks to learn various properties such as congestion [10], protocol behavior in a hotspot setting [22, 8, 16], etc. The DAIR system also uses such an approach for troubleshooting [1] and security [2]. More details on similar related works appear in Section II-B of [11]. In this paper, we employ a technique similar to [25] to merge individual traces into a unified trace. However, unlike all the previous studies, our focus is on identifying selfish carrier sense behavior in the network using the merged trace.

3 Overall Approach

3.1 Problem Statement

If one observes live network traffic for long enough time, many instances will arise where each node pair have packets to transmit at their interface queues at the same time. In 802.11, if the packet already arrived at the queue when the interface was busy (transmitting another packet), it first undergoes a random backoff, freezing the backoff counter whenever the carrier is sensed busy during the backoff interval. At the end of the backoff the packet is transmitted. If the packet arrived at an empty queue instead, it first senses carrier. If the carrier is idle, the packet is transmitted immediately. Otherwise, it waits until the carrier is sensed idle. It then undergoes a random backoff before transmitting the packet. We ignore discussing various inter-frame spacings (DIFS, etc), as they have little impact on our study.

Freezing the backoff counter in the first case, or waiting until the carrier is busy in second case is called *deferral*. If we consider node pairs at a time, the deferral behavior of each node in the pair being considered with respect to the other can be inferred via a learning technique. We will describe this technique in the next section.

Our general goal is to understand the asymmetry in the deferral behavior. If Y defers for X's transmission and X does not defer for Y's, then the link between X and Y is asymmetric. While link asymmetry is possible in wireless networks due to interface heterogeneity, it is simply unlikely if a node X will demonstrate 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 node pairs only at a time. Due to the additive nature of the received power, a given node may defer due to transmissions from a set of other nodes. This is because a single transmission may not generate enough power to cause deferral, however, multiple such concurrent transmissions may still cause deferral (physical interference). However, pairwise consideration can still be quite powerful in practice. Also, in reality more than two concurrent packet transmissions may actually be rare even when there are many simultaneous active flows in the network. For example, using a major trace collected during the SIGCOMM 2004 conference, the authors in [16] showed that only 0.45% of packets actually overlapped in transmission. Thus, learning more elaborate higher order relationships may not be very useful in identifying selfish nodes. *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.*

Because of the inherent nature of wireless environment (e..g, fading) a probabilistic measure is suitable. Thus, our goal is to estimate, via passive monitoring, the probability $P_{\text{def}}(X, Y)$ that node X defers to node Y's transmission and 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., $P_{\text{def}}(X, Y) << P_{\text{def}}(Y, X)$. If this is also witnessed by more nodes such as Z, i.e., there exists several $Z \neq Y$ such that $P_{\text{def}}(X, Z) << P_{\text{def}}(Z, X)$ we have more confidence that X is behaving in a selfish manner.

3.2 Discussions

Our technique depends on the conjecture that if one observes the live network traffic for a long enough period, enough of such instances will arise when simultaneous transmissions are attempted in the network for each node pair. Analysis of the packet trace at these instances can estimate $P_{\text{def}}(X, Y)$ and $P_{\text{def}}(Y, X)$. Our goal is to (i) identify such instances, and (ii) infer the deferral behaviors during such instances. There are several challenges here. First, creating a complete and accurate trace is difficult. But incomplete trace may suffice in circumstances when it is statistically similar to the complete trace. Second, the selfish behavior becomes harder to detect if there is relatively low load in the network. High load on the part of network nodes makes discovering selfish behavior easier. This is true both for the selfish and the 'witness' nodes. Regardless, it is important to evaluate the performance of any detection technique with varying load on both the selfish and other network nodes. Third, heuristics are used to detect selfish behavior. But straightforward heuristics may have limited power. More details about these challenges appear in [11].

3.3 Approach

Thus, to determine deferral behaviors among network nodes, one needs a rigorous statistical modeling approach, instead of relying on heuristic-based trace analysis. Our basic approach is as follows. We model sender node pairs in the network (say, X, Y) via a Markov chain based on the MAC layer operation of 802.11. 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) [18]. 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 Interactions Between Senders

A hidden Markov model (HMM) [18] consists of a system modeled as a Markov chain with unknown parameters, where the states of the Markov chain are not directly visible, but some observation symbols influenced by the states are visible. There are standard methods [18, 9, 3] to learn the unknown parameters (such as the state transition probabilities of the Markov chain) using the observed sequence of observation symbols. HMMs have been used in various machine learning fields such as pattern, speech and handwriting recognition. We will be using the HMM approach for inferring the deferral behavior between pairs of senders in an 802.11 network.

4.1 Markov Chain

The 802.11 MAC protocol can be modeled as a Markov chain for each sender [4, 11]. An 802.11 sender, say X, resides in one of the following four states - 'idle,' 'backoff,' 'defer,' and 'transmit.' These four states capture the essence of the 802.11 MAC protocol. Figure 1 shows the Markov chain modeling the 802.11 MAC protocol for each sender. We are intentionally ignoring interframe spacings (e.g., DIFS) to keep the chain simple. Let us call the 4 states id, bk, de, and tx, respectively for brevity. The high level description of this chain can be found in [11]. Here, the state transition probabilities between bk and de depend on the state 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 probability from id to de and tx, and transition probability from tx to de and bk.

Since the transmissions from other nodes impact the state transitions for a given node, a combined Markov model needs to be considered to get a complete picture of the network behavior. Here, each state is a tuple consisting of states of individual nodes. Such a Markov chain would lead to a state space explosion with exponential number of states, and would thus be intractable. Since our focus in this work is on determining the pairwise 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 2-tuple consisting of the states of X and Y. For example, the state where X transmits and Y defers would be $\langle tx, de\rangle$. There could be 16 possible states in theory. However, 5 of them are not legal (e.g., $\langle de, de\rangle$, $\langle de, bk\rangle$ etc.[2]), leaving 11 possible states. See Figure 2 for the combined Markov chain.

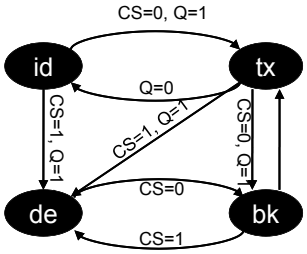

Figure 1. State transition diagram for a single sender. CS = 0 (CS=1) means that the carrier is sensed idle (busy). Q = 0 (Q =1) means that the interface packet queue is empty (non-empty).

In this Markov chain, the state transition probability between certain states depends on deferral probabilities between X and Y. For example, from state $\langle bk, bk\rangle$ to state $\langle tx, de\rangle$ or $\langle tx, bk\rangle$ would depend on deferral probability of Y with respect to X. To see this, assume that Y carrier senses X perfectly. Then when X moves from bk to tx state (i.e., starts transmitting as soon as the backoff interval is over), Y must also move from bk from de 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 bk state.

Note again that this combined Markov chain is specified for a node pair only, as we are interested in pair-wise relationships. This chain can be repeated for all pairs to determine the deferral behavior between all node pairs. When considering a particular pair, we filter out the packets of just the two senders for analysis, and ignore the other packets. This may cause an active node to appear idle for certain pe-

[2]Note that this Markov chain assumes only two nodes X and Y interact. Thus, for example, the state $\langle de, de\rangle$ is not possible as both nodes cannot defer at the same time.

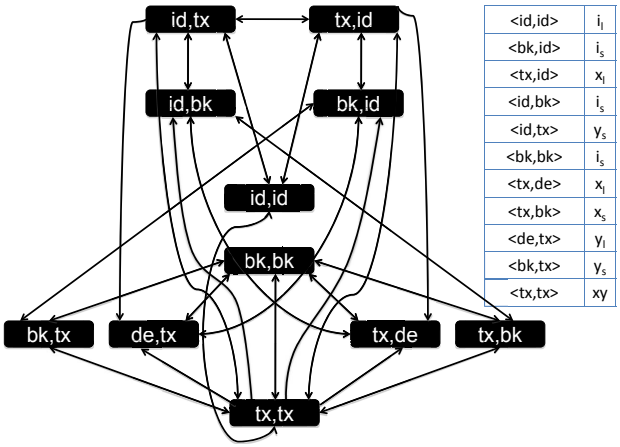

Figure 2. Markov model of the combined MAC Layer behavior of two nodes (sender side only). Note that some arrows are bidirectional.

riods of time if the node defers for a third node's transmission. While this may result in our method missing out on an opportunity to interpret the interaction between the particular pair as interfering or non-interfering, it is important to note that this does not create any incorrect interpretation. Recent studies [16] show that the number of instances of 3 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

As we do not know the deferral behavior yet, the state transition probabilities of the combined Markov chain are unknown. Also, the states of this Markov chain are not directly visible in the packet trace. We thus need to map each state in this Markov chain to an observation symbol obtained from the trace that can be used to learn the state transition probabilities. 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.

Each state in the Markov chain can be mapped to one of the four symbols above. This mapping is not unique as more than one state can map to the same observation symbol. For example, both states $\langle id, id\rangle$ and $\langle bk, bk\rangle$ map to the symbol i. Similarly, both $\langle bk, tx\rangle$ and $\langle de, tx\rangle$ 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 comes from a random process and can last for integral number of slots (20 μs in 802.11b). Also, the maximum backoff interval is bounded (31 slots in the first backoff stage[3]). While not impossible, it is very unlikely that a defer or idle period will be within this bounded interval and also last for exactly an integral number of slots.

This strategy to distinguish between backoff and idle/defer periods requires highly accurate clocks (within few microseconds). Without a specialized technique, the experimentally observed accuracy is not sufficient. 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 when airtime of an 802.11 frame is less than 31 slots (620 μs for 802.11b[4]). This also introduces errors for very small idle times. With these sources of error, the results in the next section provide only a lower bound on the accuracy obtainable by the base technique. In our future work, we will explore possibilities of using accurate timing information to remove these inaccuracies. Moreover, the parameters of this heuristic is not fundamental to our technique. We can use this technique by changing the paramaters for 802.11a/g.

With the above weaker heuristic, each observation symbol can be of two types. The symbol i can be either i_s or i_l, corresponding to short ($\leq$ 31 slots) and long ($>$ 31 slots) respectively. According to the heuristic, i_s is most likely output by $\langle bk, bk\rangle$ state, while i_l is most likely output by $\langle id, id\rangle$ state, for example. Similarly, the symbols x and y can be either x_s and x_l, and y_s and y_l, respectively. Figure 2 shows the observation symbols for each state.

With the help of the heuristic, we can distinguish between backoff and idle/defer periods. But we still cannot differentiate between idle and defer. For this reason, both the states $\langle tx, id\rangle$ and $\langle tx, de\rangle$ map to the same observation symbol x_l. This implies that the transition from state

[3] 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).

[4] This means TCP packets with payload less than 400 bytes in 802.11b to give the reader an idea.

$\langle tx, id\rangle$ to state $\langle tx, de\rangle$ will not be visible in the merged trace as there is no change in the observation symbol. Thus any transition from state $\langle tx, id\rangle$ to any other state, for example, state $\langle id, bk\rangle$ via state $\langle tx, de\rangle$ will not be correctly interpreted. To overcome this problem, we force transition links from state $\langle tx, id\rangle$ to states which have incoming transition from state $\langle tx, de\rangle$. We refer to these links as virtual links. Similarly, we also add virtual links from state $\langle id, tx\rangle$ symmetrically. After we calculate the transition probabilities of the model using the technique described in the following subsection, we remove such virtual links and distribute the probability on each such virtual link to the corresponding sequence of valid transition links.

Each packet in the merged packet trace consists of a timestamp for when the packet was received at the sniffer, the id of the sender, size of the packet, and the rate at which it was transmitted. This information is parsed to obtain the sequence of above observation symbols from the trace. 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 deferral between the senders.

4.3 Formal Specification and Learning

We now provide the complete formal specification of the HMM using standard notations [18]. The HMM consists of the following:

- Set S of N states, where $N = 11$. S is given by: $S = \{S_i\} = \{\langle id, id\rangle, \langle bk, id\rangle, \langle tx, id\rangle, \langle id, bk\rangle, \langle id, tx\rangle, \langle bk, bk\rangle, \langle tx, de\rangle, \langle tx, bk\rangle, \langle de, tx\rangle, \langle bk, tx\rangle, \langle tx, tx\rangle\}$.

- 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 absent in Figure 2.

- 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. But they are not unique. The mapping from states to symbols are shown in a table within Figure 2.

- 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 defines the HMM, $\lambda = (A, B, \pi)$. The packet trace provides the observation sequence $O = O_1, O_2, \cdots O_T$, 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 O, we wish 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 [9]. We use the well-known *Baum-Welch method*, which is a type of EM algorithm, based on the forward-backward algorithm developed by Baum *et al.* [3]. 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 using the Baum-Welch method, we do not readjust the parameters B and π in the model λ. We initialize the state transition probabilities such that equal probability is assigned to all the outgoing valid transitions from each state. This ensures that there is no initial bias in the model towards interfering or non-interfering pair of nodes. This aids in quick convergence of the method. We also need to use the scaling technique in the procedure [15]. This is needed as we deal with very long sequences of observations and continued multiplications of certain small fractions create problems with numeric accuracies.

4.4 Detecting Asymmetric Behavior

Let $\Pi = [\Pi_i]$ be the stationary (steady state) distribution of the states. Once the transition probabilities $A = [a_{ij}]$ are learnt, $\Pi = [\Pi_i]$ can be determined as $\Pi = \lim_{n\to\infty} \pi A^n$. The convergence is guaranteed as A is a stochastic matrix. Now, asymmetric behavior can be detected the following fashion.

If we represent Π_i's as $P(id, id)$, $P(bk, id)$ etc, the probability that X has a packet to transmit and it defers while Y transmits is given by

$$P_{\text{def}}(X, Y) = \frac{P(de, tx)}{P(de, tx) + P(bk, tx) + P(tx, tx)}.$$

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

$$P_{\text{def}}(Y, X) = \frac{P(tx, de)}{P(tx, de) + P(tx, bk) + P(tx, tx)}.$$

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) is getting 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 $\eta(X, Y) = -\eta(Y, X)$.

4.5 Selecting Witnesses

In general, each network node X must be evaluated for selfish behavior. By default, every other node Y acts as 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 $P(de, tx)$ or $P(tx, de)$ 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 regarding which node can serve as a effective witness. Ideally, we would only rely on witnesses that are within the carrier sense 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 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.

5 Evaluation

We have performed two sets of evaluations: (i) a set of microbenchmarking experiments to understand the effectiveness of the approach and (ii) a set of ns2 simulations to study larger networks and complex selfish behaviors.

5.1 Experiments

The experiments essentially achieve careful microbenchmarking, where only two network links are used but wireless channel quality, traffic load and selfish behaviors are varied over a wide range. The transmitter of one link is 'selfish'; the other transmitter is regular and acts as the sole 'witness.' Each link is monitored by a 'sniffer' node in close proximity of the transmitter. 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 the trace merging. In a normal deployment these beacons will be supplied by existing APs. We use 802.11a

in preference to more widely used 802.11b/g to reduce external interference. We have verified that no other 802.11a transmissions exist in our testbed location in the channel we used for our experiments (channel 52).

Each network node is essentially a single-board computer (SBC), meant for embedded use, with an 802.11a/b/g interface. We use Soekris 4826 boards [24] and Atheros 5213 chipset-based 802.11a/b/g mini-PCI cards manufactured by Winstron connected to a 5 dBi rubber-duck antenna. The boards run pebble Linux with `madwifi` device driver for the 802.11 interface. The 802.11 interfaces in the network nodes are set up in 'ad hoc' mode and the sniffer nodes in 'monitor' mode. The lowest possible PHY layer rate (6 Mbps) and a large packet size (1470 bytes) is chosen for the experiments. This is because, at higher rates or 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. The radiotap headers in the captured packets are used to record SNR.

The selfish transmitter achieves selfishness by not sensing carrier before transmitting. To achieve this we have used the antenna switching technique described in [6]. The 802.11 interface uses two antenna connectors for diversity. We have only one antenna connected to one connector, keeping the other connector unconnected. Using driver-level commands, any one of the connectors can be selected as the receiving/transmit antenna. Selecting the unconnected antenna as the receiving antenna effectively disables carrier sense.[5] 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 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, (i) the average SNR from the selfish to the witness transmitter is measured, and (ii) iperf is used to transmit UDP packets at different offered loads on their respective links for 60 sec. We use offered loads of 6 and 4 Mbps for iperf, denoting high and low loads, respectively. We experiment with both loads on the selfish node, while the witness node has only high load.

Figure 3 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.

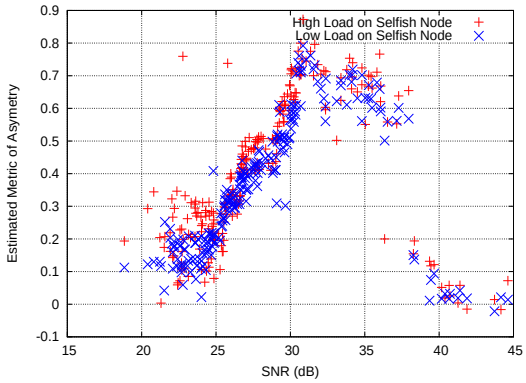

Figure 3. Experimental results with varying load on the selfish node.

Note that the above two node micro-benchmarking is sufficient to derive an insight into what would happen in a multiple node network. Essentially, nodes still need to be evaluated in a pair wise 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 interface only. But several other issues remain to be evaluated – (i) how to effectively combine the metric of asymmetry, (ii) how suitable are the witness nodes. We will explore these issues via a packet level simulation using the ns2 simulator.

5.2 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 de-

[5] Note that selfishness can also be achieved by resetting the CCA threshold as in [17]. However, in our hardware we have found that the antenna switching technique is more foolproof than using an increased CCA threshold.

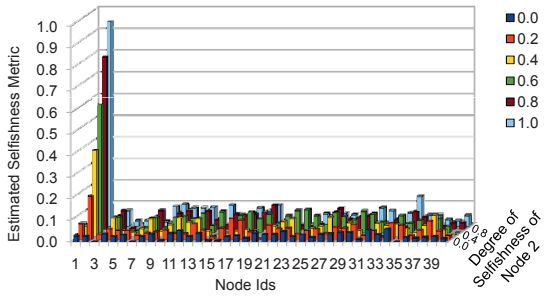

Figure 4. Simulation results for a 40 node network. Node 2 is the only selfish node. The estimated selfishness metric using heuristic H_2 is shown for each node for each of the 6 sets of simulations that are run with different degree of selfishness of node 2.

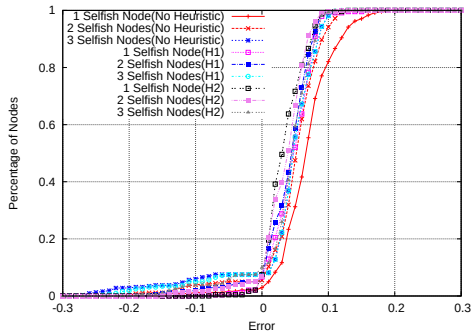

Figure 5. CDF of 'estimation error' for the selfishness metric. Three different scenarios are presented where number of selfish nodes are varied (1, 2 or 3) and witness nodes are identified in three different ways.

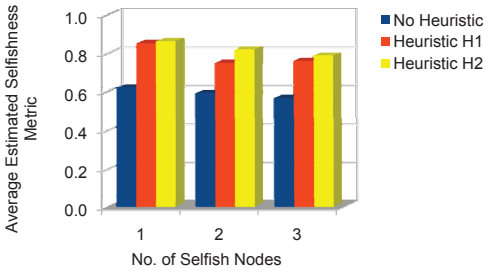

Figure 6. Simulation results for the sparse network.

ployment typical of dense WiFi client distribution in indoor office environments, assuming that there is one node in 300 sq. feet on average. The default ns2 wireless channel model is extended to include *shadowing* [20]. This introduces randomness in the transmission range of a node instead of making it a perfect disk. Shadowing parameters are taken from [12] 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 1-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 sec. 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 4.5: (i) using all possible witness nodes, (ii) using witness nodes based on heuristic H_1, and (iii) based on heuristic H_2.

Figure 4 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 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 Figure 5. 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 1500

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. Figure 6 shows the average estimated selfishness metric measured in three ways as before only for the selfish node(s). Note that as expected (i) estimation becomes better when we identify witness nodes using the heuristics in compared to using all the nodes as witness; (ii) H_2 is generally a better heuristic, and (iii) estimation becomes worse with a larger number of selfish nodes.

6 Conclusions

We have investigated a novel machine learning-based approach 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. The power of this technique is that it is purely passive and does not require any access to the network nodes. It can be used as a third-party solution for detecting MAC-layer misbehavior in 802.11 networks. Though it works offline, but it can be used periodically every few minutes (for example) to detect selfish nodes. Evaluations show excellent detection ability in presence of varying load and degree of selfishness.

There are indeed some limitations of the technique as presented here. So far, we only assumed pairwise interference and 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. Finally, the proposed technique along with an established technique for selfish backoff manipulation [19, 21] can form a complete solution to detect selfish behaviors in 802.11 networks.

References

[1] P. Bahl, et al. DAIR: A framework for troubleshooting enterprise wireless networks using desktop infrastructure. In *ACM HotNets-IV*, 2005.

[2] P. Bahl, et al. Enhancing the security of corporate Wi-Fi networks using DAIR. In *ACM MobiSys*, 2006.

[3] 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. Amer. Math. Soc.*, 73:360–363, 1967.

[4] G. Bianchi. Performance analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE JSAC*, 18(3):535–547, 2000.

[5] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On selfish behavior in csma/ca networks. In *Proc. IEEE Infocom Conference*, 2005.

[6] K. Chebrolu, B. Raman, and S. Sen. Long-distance 802.11b links: Performance measurements and experience. In *ACM MobiCom*, pages 74–85, 2006.

[7] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benkö, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. *Proc. ACM SIGCOMM*, 2007.

[8] 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.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[10] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding congestion in IEEE 802.11b wireless networks. In *ACM IMC*, 2005.

[11] A. Kashyap, U. Paul, and S. R. Das. Deconstructing Interference Relations in WiFi Networks. In *Proc. IEEE SECON*, 2010.

[12] A. Kashyap, S. R. Das, and S. Ganguly. Measurement-based approaches for accurate simulation of 802.11-based wireless networks. In *Proc. ACM MSWIM*, 2008.

[13] J. Konorski. Multiple access in ad hoc wireless lans with noncooperative stations. *LNCS*, 2345:1141–1146, 2002.

[14] P. Kyasanur and N. Vaidya. Detection and handling of mac layer misbehavior in wireless networks. In *Proc. IEEE DSN*, 2003.

[15] 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 Syst. Tech. J.*, 62(4):1035–1074, 1983.

[16] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the MAC-level behavior of wireless networks in the wild. In *Proc. ACM SIGCOMM*, 2006.

[17] K. Pelechrinis, G. Yan, S. Eidenbenz, and S. V. Krishnamurthy. Detecting selfish exploitation of carrier sensing in 802.11 networks. In *Proc. IEEE Infocom Conference*, 2009.

[18] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[19] S. Radosavac, J. S. Baras, , and I. Koutsopoulos. A framework for mac protocol misbehavior detection in wireless networks. In *Proceedings of the 4th ACM workshop on Wireless security*, 2005.

[20] Theodore S. Rappaport. Wireless Communications: Principles and Practice. *Prentice Hall*, 2002.

[21] M. Raya, J.-P. Hubaux, and I. Aad. Domino: A system to detect greedy behavior in IEEE 802.11 hotsopts. In *Proc. ACM Mobisys*, 2004.

[22] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan. Measurement-based characterization of 802.11 in a hotspot setting. In *ACM E-WIND*, 2005.

[23] D. P. Satapathy and J. M. Peha. Performance of unlicensed devices with a spectrum etiquette. In *Proc. IEEE GLOBECOM*, 1997.

[24] Soekris Engineering. http://www.soekris.com.

[25] J. Yeo, M. Youssef, and A. Agrawala. A framework for wireless LAN monitoring and its applications. In *Proc. ACM WiSe*, 2004.