Detection of Selfish Wi-Fi Tethering in Multi-AP Networks

Jaehyuk Choi, *Member, IEEE,* Alexander W. Min, *Member, IEEE,* Kang G. Shin, *Fellow, IEEE,*

Abstract—Wi-Fi tethering, using a mobile device (e.g., a smartphone or a tablet) as a hotspot for other devices, has become a common practice. Despite the potential benefits of Wi-Fi tethering, the open source nature of mobile operating systems (e.g., Google Android) can be abused by a selfish device to manipulate channel-access parameters to gain an unfair advantage in throughput performance. This can cause serious performance problems within a well-planned Wi-Fi network due to an unauthorized selfish tethering device interfering with nearby well-planned APs. In this paper, we show that the selfish behavior of a tethering node that adjusts the clear channel assessment (CCA) threshold has strong adverse effects in a multi-AP network, while providing the selfish node a high throughput gain. To mitigate/eliminate this problem, we present a passive online detection architecture, called `CUBIA`, that diagnoses the network condition and detects selfish tethering nodes with high accuracy by exploiting the packet loss information of on-going transmissions. To best of our knowledge, this is the first to consider the problem of detecting a selfish tethering node in multi-AP networks.

Keywords—Network monitoring, Wi-Fi Tethering, Selfish Behavior

1 INTRODUCTION

With the rapidly-increasing deployment of mobile devices and skyrocketing demand for mobile data traffic [6], Wi-Fi has been the most widely-available and energy-efficient means to provide the Internet access to mobile users and devices [2]. While most of the today's mobile devices are equipped with Wi-Fi interfaces, supporting very high throughput performance even with small form-factor smartphones, e.g., up to 433 Mbps for 802.11ac,[1] the limited and short coverage of Wi-Fi networks restrict their usage. For example, a mobile user may not be able to find a Wi-Fi access point (AP) while traveling by a car on a highway. Or a mobile user in a coffee shop may find an AP with very limited bandwidth because many people share the same AP. To address these issues, Wi-Fi tethering has recently been gaining popularity as a means to provide Internet connectivity by sharing the Internet connection of mobile devices with another Wi-Fi-enabled devices through Wi-Fi [4].

While Wi-Fi tethering allows mobile users to go online almost anywhere, even on-the-move, the ease of setting up a tethered Wi-Fi hotspot and the open source nature of mobile Operating Systems (OSes) can pose serious performance problems to well-planned Wi-Fi networks, such as enterprise and campus networks [7]. Since the tethered Wi-Fi hotspot can potentially open up the network with an arbitrary channel number without restriction from the network administrator, it may interfere with nearby well-planned APs and cause unpredictable performance degradation, e.g., availability and throughput. Even worse, the open and customizable nature of mobile OSes, such as Google Android [35], can be further abused by misbehaving devices/users to gain an unfair advantage in throughput performance by manipulating the tethering function. For example, by rooting or jail-breaking mobile OSes, the channel access functions of Wi-Fi protocol, i.e., IEEE 802.11, can be manipulated "selfishly," at the cost of other nearby well-behaving Wi-Fi devices' performance. Therefore, it is essential to design an efficient mechanism to detect unauthorized and misconfigured Wi-Fi tethering.

In this paper, we observe the network throughput behavior in a multi-AP environment in the presence of a selfish tethering node, and study the impact of selfish tethering on other users' throughput performance. Our evaluation results reveal that the symptoms of selfish tethering resembles that of the well-known hidden node problem—both result in a high frame transmission error rate. However, the main difference is that the frame losses at legitimate nodes caused by the selfish tethering cannot be easily resolved by the RTS/CTS mechanism because the selfish nodes may not recognize the RTS/CTS frames due to their manipulated CCA thresholds, or RTS/CTS frames can be easily ignored by the selfish nodes. Therefore, it is imperative to design a detection mechanism that can identify the root cause of the throughput degradation and pinpoint the selfish node in the network.

Based on our observations, we present an online Wi-Fi tethering detection mechanism, called `CUBIA`, that can accurately detect selfish Wi-Fi tethering nodes in a multi-AP environment. `CUBIA` runs at each AP and passively monitors the channel-access success/failure behavior of ongoing data traffic to detect any abnormal changes caused by selfish tethering. In particular, it monitors any noticeable increase in frame loss rate to trigger the

1. The new 802.11ac standard supports up to several Gbps of throughput depending on transmission configurations, e.g., the number of antennae and channel bandwidth.

second-phase detection mechanism, where it attempts to identify the root cause of the frame loss, e.g., selfish tethering or hidden node problem. For this, we formulate the selfish tethering detection problem as a hypothesis testing problem and employs a modified CUSUM algorithm. We evaluate `CUBIA` with an in-depth simulation in various wireless environments, and our evaluation results show high detection accuracy. To best of our knowledge, this is the first to consider the problem of detecting selfish misconfigured Wi-Fi tethering nodes in a multi-AP network.

1.1 Contributions

This paper makes several contributions as follows.

- We observe that concurrent transmission mechanisms, e.g., PHY capture or MIM, can be abused by selfish nodes, and study the impact of the selfish Wi-Fi tethering behavior on the throughput performance of other nearby well-behaving nodes.
- We propose a selfish misbehavior detection mechanism, called `CUBIA`, that can accurately identify the cause of frame losses and detect selfish behavior under strict detection latency requirements.
- We design a two-step detection process using an online change detection algorithm, i.e., CUSUM, based on passive monitoring of frame loss rates at multiple APs. We also study the impact of detection thresholds on detection latency/accuracy performance.
- We perform extensive simulation-based evaluation for various network conditions. Our evaluation results show that `CUBIA` can promptly detect selfish behavior with high accuracy in realistic wireless environments.

1.2 Related Work

Selfish and malicious misbehavior in CSMA networks has been extensively studied under various scenarios in different communication layers. Majority of previous work has concentrated on detecting or punishing MAC-layer misbehavior [13], [25], [24], [27]. In [13], Kyasanur and Vaidya studied the MAC-layer misbehavior of selecting always small backoff values rather than random selection, and showed that such selfish misbehavior can seriously degrade the network performance. Raya *et al.* [25] investigated multiple misbehavior policies in 802.11 MAC protocol including backoff manipulations, and presented a detection framework, called `DOMINO`, which considers all possible strategies jointly and improves the detection accuracy. Several statistic-based frameworks for detecting misbehavior also have been introduced [24], [27]. Their common detection approach is to measure the inter-arrival time of target stations in terms of the number of backoff slots and verify whether the backoff time of the stations follows a legitimate pattern or not. Recently, the authors in [26] proposed a non-parametric CUSUM test to detect real-time selfish misbehavior in 802.11 networks. The problem of selfish misbehavior also has been addressed in several different layers and perspectives, including routing layers [3], [17], multi-channel protocols [32], and game theory-based behaviors [16], [1].

The problem of misbehavior using CCA threshold is relatively new and unexplored in the literature. In [30], the authors have identified the effect of an 802.11 node's selfish behavior by increasing CCA threshold. They have shown that the selfish behavior can achieve higher throughput than other well-behaving nodes based on a game-theoretical model. Recently, the authors in [23] addressed the selfish carrier sense problem in a Wireless LAN (WLAN). The authors have performed experimentation on a real-world testbed and shown that such selfish behaviors can cause extremely unfair allocations of the wireless medium. However, none of these studies considered the problem of the selfish problem exploiting Wi-Fi tethering environments.

1.3 Organization

The remainder of paper is organized as follows. Section 2 describes the system model and adversary model. Section 3 illustrates the impact of selfish manipulation of the CCA thresholds on throughput performance in multi-AP environments. Section 4 identifies the unique features of selfish tethering and proposes `CUBIA` that accurately detects the selfish behavior. Section 5 presents the evaluation results and Section 6 concludes the paper.

2 SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we present the network, channel and capture models, and describe selfish misbehavior in Wi-Fi tethering.

2.1 Network Model

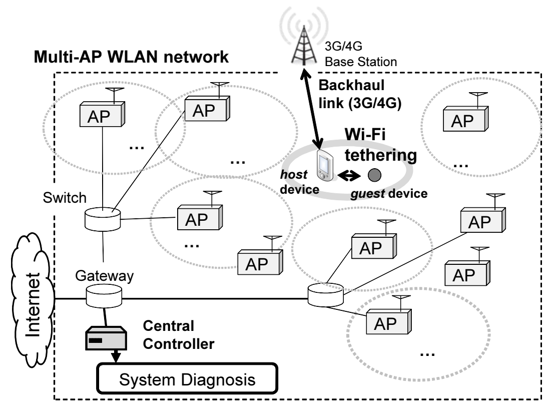


Fig. 1. Illustration of the problem: selfish misconfigured Wi-Fi tethering sets up the network in a centrally managed multi-AP network

We consider a scenario in which an unauthorized Wi-Fi tethering system sets up a Wi-Fi hotspot within a centrally managed Wi-Fi network, where the network consists of a central *controller* and N 802.11 access points (APs), $A = \{AP_1, AP_2, \cdots, AP_N\}$, as illustrated in Fig. 1. The tethering consists of an Internet-connected (e.g., through 3G/4G cellular connection) mobile phone and a tethered Wi-Fi-enabled device, contending for channel access with nearby Wi-Fi systems. Let B_{cel} denote the capacity of the cellular backhaul link. We will henceforth refer to the mobile device and the tethered Wi-Fi device as *host* and *guest* nodes, respectively; the *host* node shares its 3G/4G Internet connection with the *guest* nodes via its Wi-Fi interface.

We assume that the multi-AP network is monitored and managed by the controller, as shown in Fig. 1. Each AP_i ($AP_i \in A$) monitors the channel access activity and reports the information to the controller periodically. We also assume that frequency planning for each AP in the network has been done *a priori* through a proper channel allocation algorithm (e.g., [19], [20], [22]), so as to minimize inter-AP inference on each AP.

2.2 Channel Model

For a given link, let s and r denote the transmitter node and its corresponding receiver node of a link, respectively. The distance between the two nodes s and r is denoted by $d_{s,r}$. We assume that the channel gain $G_{s,r}$ between the transmitter s and the receiver r is determined based on the log-distance path-loss model described in [28]:

$$G_{s,r} = \frac{1}{{d_{s,r}}^{\alpha}}, \tag{1}$$

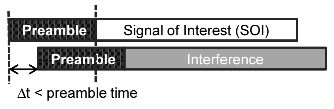
where α is the path-loss exponent (normally, ranging from 2 to 5). Let P_s and P_r denote the transmission power of node s and the received power at r, respectively. We assume that all the transmitters transmit frames at the same nominal power P_0. Then, P_r is expressed as $P_r = G_{s,r}P_0$. The received signal to interference ratio (SIR) SIR_r at the receiver is expressed as

$$SIR_r = \frac{G_{s,r}P_0}{\sum_{k \neq i} G_{k,r}P_0}. \tag{2}$$

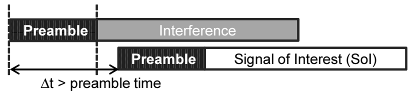
Let γ_m denote the minimum SIR requirement (i.e., SIR threshold) for successful reception at a receiver node under modulation scheme m, where we consider multi-rate MAC as in 802.11a/b/g/n.

2.3 Capture Model

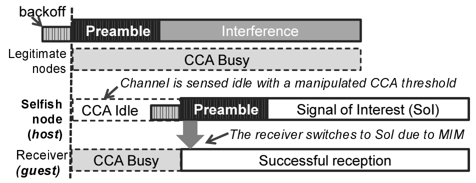
Even when multiple independent transmissions occur simultaneously to a receiver, the receiver can successfully capture the signal of interest (SoI) if the received SIR is higher than a certain SIR threshold, which is known as the *capture effect* [14].



(a) PHY capture effect



(b) MIM (Message-In-Message)

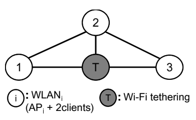


(c) Adversary Model

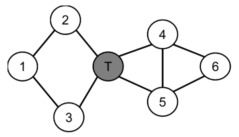
Fig. 2. (a) PHY capture effect allows a receiver to successfully capture the signal of interest (SoI) if its Tx power is sufficiently higher than the sum of interferences. (b) MIM (Message-In-Message) allows a receiver to disengage from an ongoing packet reception, and engage in a new, stronger packet. (c) Selfish behavior with CCA manipulation exploits the benefits of MIM if the selfish pair (e.g., tethering link) has a short link distance.

There are two well-known concurrent transmission technologies: PHY capture effect [15] and message-in-message (MIM) [14], [18]. Fig. 2 illustrates the main difference between PHY capture and MIM. The PHY capture effect is the property of 802.11 radios, where an SoI can be decoded successfully even when the interference arrives at the same time, as long as the overlap *within* the preamble detection stage and the SoI is stronger than a specific threshold, which we call the *capture threshold*. MIM is an enhanced PHY-layer capability that enables a receiver to decode an SoI even if the SoI arrives after the preamble time of the interference. Specifically, MIM allows a receiver to disengage from the current on-going frame reception and re-engage in a new, stronger frame. However, MIM requires a higher SIR than the PHY capture, which we call this SIR value the *MIM threshold*, β_m, for modulation scheme m. These capture and MIM thresholds can vary [29], [14], particularly depending on the modulation scheme [14].

Throughput the paper, we assume that a receiver can decode a frame successfully with probability greater than 0.9 when the received SINR is consistently above the SIR threshold for a given modulation scheme. We use the experimental results in [14] to configure the SIR threshold γ_m and MIM threshold β_m for modulation scheme m (e.g., m=BPSK, QPSK, 16QAM, and 64QAM), which satisfy the requirement of achieving the 90 % frame reception ratio.



(a) Topology-1



(b) Topology-2

Fig. 3. AP Interference graph of two simulated topologies.

2.4 Adversary Model

We assume that an adversary user manipulates the CCA threshold of the *host* node's Wi-Fi interface in the tethering while the guest device is legitimate. We assume that the *host* node selects the maximum possible CCA threshold without losing the tethering connectivity, which is likely a little lower than the observed strength of received signals transmitted by the guest node. In addition, we assume that the adversary user minimizes the link distance as much as possible so as to exploit the benefits of physical-layer concurrent transmission technologies, i.e., PHY capture effect and MIM. Recall a property of Wi-Fi tethering: in general, a tethered hotspot is formed for communication between personally owned devices which are placed close-by while in use, and thus the link distance between the communicating devices is highly *controllable* and typically short (less than 10 m or 30 ft [4]).

As shown in Fig. 2(c), the selfish tethering can thus gain exclusive channel access opportunities, while guaranteeing successful packet delivery. On the other hand, the legitimate Wi-Fi nodes in the vicinity will suffer from severe packet corruptions due to the transmissions from the selfish tethering.

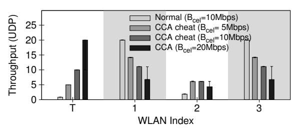
3 Selfish Configuration in Wi-Fi Tethering

To understand the impact of the selfish behavior of a Wi-Fi tethering system using CCA tuning, we performed extensive simulations with different transport-layer protocols (i.e., TCP and UDP, in multi-AP environments) using the ns-2 simulator [33].

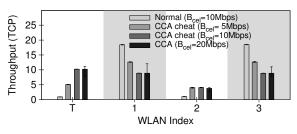
Multi-AP Topology: First, we study the impact when a selfish tethering launches within a well-planned multi-AP network with two representative topologies. The topologies used in the simulations are shown in Fig. 3.[2]

We compare the performance for two cases using UDP and TCP protocols: (i) *Normal* scenario in which the *host* node uses the default CCA threshold, and (ii) *Selfish* scenario in which the *host* node manipulates the MAC

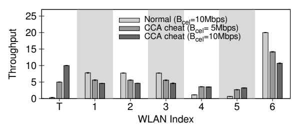
2. We sampled the topologies from a popular Wi-Fi database *Wigle*. For example, the topology shown in Fig. 3(a) corresponds to the well-known FIM (Flow-In-the-Middle) topology [8], which can be easily found in real-world AP deployments [5]. An edge in the figure represents the neighboring interference, meaning that an AP is in the carrier sensing range of its connected APs. The vertex denoted by 'T' in the figure represents the tethering system launched within the network.



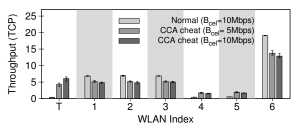
(a) Topology-1, UDP



(b) Topology-1, TCP



(c) Topology-2, UDP



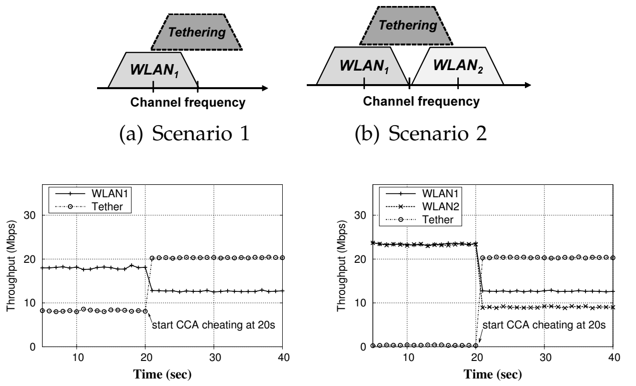
(d) Topology-2, TCP

Fig. 4. Impact of selfish carrier sense on throughput of transport-layer protocols over various cellular backhaul link capacities B_{cel} for tethering in two multi-AP topologies.

protocol by increasing the CCA threshold as high as possible without losing the connectivity to the guest. We used the following settings in our simulations. Each WLAN has an AP and two associated client nodes (i.e., 2 AP–clients flows per WLAN), and all nodes operate on the same channel. We considered typical IEEE 802.11a/g MAC/PHY parameters with the maximum speed of 54 Mbps, and set the capacity B_{cel} of the 3G/4G backhaul link of tethering to 5, 10, and 20 Mbps. Traffic is generated by a constant bit rate (CBR) traffic generator for the UDP protocol (1 KB packet size), and an FTP download application is used to create TCP flows (1.5 KB packet size). For UDP flows, we assume 10 Mbps for flows in infrastructure APs, and 20 Mbps for tethering flows.

Fig. 4 shows the throughput of the tethering link and APs. In *Normal* case, we can see a throughput imbalance among APs, and especially, the tethering flow is shown to experience starvation due to the FIM problem [8]. With the CCA cheating, on the other hand, we can see that the selfish tethering achieves a significant throughput gain at the cost of significant throughput degradation for other nearby well-behaving APs, for both UDP and TCP flows. Although we manipulate the CCA threshold only on the host side, i.e., the *guest* node is legitimate, the selfish tethering achieves a high throughput gain even with the TCP protocol, which is a bi-directional, transfer-based protocol. This is attributed to the closed-loop TCP-ACK mechanism, i.e., the more data packets (or ACKs) the legitimate guest successfully receives from the misconfigured host, the more outstanding uplink ACKs (or data packets) the guest can transmit. The figure shows that the throughput gain increases proportional to the cellular backhaul link bandwidth, B_{cel}, for both UDP and TCP flows.

One can expect that selfish CCA manipulation would increase the collision probability of the tethering link since the *host* node initiates packet transmission even in the present of other nodes' transmissions. Nevertheless,



(a) Scenario 1 (b) Scenario 2

(c) Throughput for Scenario 1 (d) Throughput for Scenario 2

Fig. 5. Impact of launching tethering on a partial-overlapped channel.

we observe a significant gain of selfish behavior using CCA manipulation, as seen from the above results. To understand why the selfish tethering link can achieve a significant gain with CCA manipulation, we investigate the property of the successful receptions at the tethering receiver node, i.e., *guest* node. Table II plots the collision probability, and the percentage of successful receptions at the receiver in the simulation with B_{cell}=10 Mbps corresponding to the results shown in Figs. 4(a) and 4(c), respectively. We observe that, despite the high collision ratios (71.5 % and 91.8 % for Topology 1 and 2, respectively), the tethered receiver can capture the signal of interest (SoI), thanks to PHY capture effect and MIM. A majority of the successful receptions are due to MIM, i.e., 63.6 % and 63.5 % for Topology 1 and 2, respectively. Note that the link distances between *host* and *guest* nodes are very short, making the received signal at the *guest* node sufficiently stronger than the sum of interferences.

TABLE 1
Statistics of receptions at the receiver (in case of UDP and B_{cell}=10 Mbps

Topology	Collision prob.	Capture effect	MIM
1	71.5	17.9	63.6
2	91.8	28.3	63.5

These results clearly demonstrate that the selfish behavior using CCA manipulation abuses the short link distance property of Wi-Fi tethering and thus makes it possible to exploit the benefits of MIM. Consequently, the selfish tethering can achieve an unfair throughput gain regardless of the network condition surrounding the tethering.

Impact of Tethering Channel: Next, we study the impact of the CCA manipulation on performance when tethering is launched on a channel partially overlapping with nearby APs. As mentioned above, the tethered Wi-Fi hotspot can potentially set up the network with an *arbitrary* channel number, which may cause serious interference to nearby well-planned APs. We used two simple scenarios in [31] and the channel model presented in [21] for our simulation. Fig. 5(a) illustrates a simple case that the channel selected by the tethering is partially overlapping with a nearby AP. Since two overlapping channels are sensed by each other by the CCA mechanism of 802.11, its effective spectrum usage is only 20 MHz [31]. Fig. 5(b) depicts the case when the tethering shares the spectrum with two adjacent orthogonal channels. Note that in this case, the tethering link may suffer from channel starvation [31]; the tethering link can transmit only when both $WLAN_1$ and $WLAN_2$ are idle, but the probability of the two outer channels being idle at the same time is very low because the channel activities on the two outer channels are asynchronous and may overlap randomly.

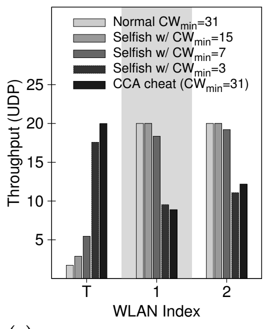
To evaluate the impact of the selfish behavior using CCA manipulation in these scenarios, we performed simulations with following setting. Each WLAN consists of two client nodes where the traffic on each flow is generated with 10 Mbps downlink CBR over UDP. The capacity B_{cel} for tethering is configured to 20 Mbps (with 20 Mbps CBR/UDP traffic). Initially, the tethering link is set to be legitimate. The CCA manipulation is activated at 20 s.

Figs. 5(c) and 5(d) plot the resulting throughput showing the effect of selfish behavior using CCA manipulation. When the tethering is legitimate, it achieves a fair share of shared medium with two flows in $WLAN_1$ in the first scenario. In the second case shown in Fig. 5(b), the tethering is almost starved due to the channel starvation problem [31]. After the CCA value is configured selfishly, despite the same unfair channel condition, the tethering link achieves a significant throughput gain at the cost of significant reduction in throughput of other flows.

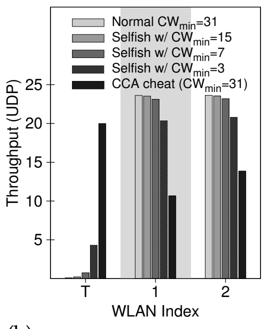
In the same scenario depicted in Fig. 5(b), we also compare the impact of CCA manipulation with different types of selfish behavior manipulating other MAC parameters, in particular manipulation of the backff mechanism using smaller values of CW_{min}.[3] Fig. 6 shows the simulation results with different values of CW_{min} =31, 15, 7, and 3. The figure indicates that the selfish behavior using CCA manipulation achieves throughput gain above those with smaller values of CW_{min} and even higher than very aggressive setting with $CW_{min} = 3$. The results imply that the manipulation of CCA threshold is a simple, yet more attractive approach that can be abused by adversaries in a tethering environment.

Impact of Cellular Backhaul Link Capacity: Finally, we evaluate the impact of the backhaul link capacity of selfish tethering. Fig. 7 shows the throughput gain of the selfish behavior as a function of backhaul link capacity for TCP and UDP downlink traffic in a network

3. The manipulation of backoff mechanism is widely adopted by selfish users [13].



(a)
2 client nodes per WLAN



(b)
5 client nodes per WLAN

Fig. 6. Throughput comparison with selfish configurations of the CW_{min} parameter.

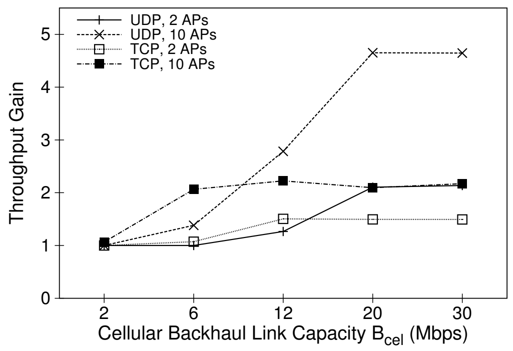


Fig. 7. Throughput gain of selfish carrier sensing over various cellular backhaul link capacities and AP densities.

with a high density of APs consisting of 10 APs and 30 client nodes. The figure indicates that the throughput gain is proportional to the backhaul link capacity of the maximum achievable goodput determined by the transport-layer protocol.[4] This is because the higher backhaul link capacity the tethering is connected to, the more outstanding packets the selfish node can transmit.

4 Detection Architecture in a Multi-AP Network

In this section, we first identify the key features of selfish carrier sensing behavior. We then propose an online detection architecture, called CUBIA, that diagnoses the network condition in a multi-AP network, consisting of (i) a *AP-level diagnosis* stage and (ii) a *system-level reaction* stage. In what follows, we will detail the design of this architecture.

4. Note that in our simulation when the backhaul apacity is larger than 20 Mbps, the maximum throughput is bounded by the Wi-Fi link capacity in this simulation.

4.1 Key Feature of Selfish Carrier Sense Behavior

Selfish carrier sensing on a tethered host node may result in excessive frame loss of nearby legitimate APs, and thus, successful detection of selfish misbehavior depend largely on a AP's ability of identifying the root cause of its frame losses. In 802.11 WLANs, there are four main causes of frame losses: (i) PHY-layer link quality degradation, (ii) MAC-layer collision, (iii) hidden nodes, and (iv) selfish nodes (e.g., manipulation of the CCA thresholds).

Fortunately, selfish tethering exhibits unique features that can help distinguishing the selfish carrier sensing of a tethering host from other causes of frame losses, especially from the hidden node problem. We made the following two observations:

O1) While the victim nodes of the selfish tethering experience symptoms similar to those of the *hidden node problem* (i.e., transmissions of the legitimate node are continuously corrupted by the selfish node), the victim nodes' communications cannot be fully protected by the Request-to-Send/Clear-to-Send (RTS/CTS) exchange mechanism, i.e., a virtual carrier sensing, because the selfish node may not recognize (due to its short communication range) or respect the RTS/CTS frames.

O2) When selfish tethering exists, the victim nodes may suffer from increased number of frame losses, hence throughput degradation, because the selfish tethering initiates a transmission even when the victim nodes have already been transmitting packets. In a multi-AP network, majority of nearby APs will likely to experience the similar severe packet errors simultaneously when the selfish tethering is present. In contrast, the impact of the hidden node problem is limited to the nodes in the hidden area.

4.2 CUBIA: A Proposed Selfish Diagnosis Algorithm

Based on the above observations, we propose a simple, yet efficient online detection algorithm at each AP, which can accurately distinguish the selfish behavior with CCA manipulation from other types of network problems, such as severe network congestion (i.e., collisions) and the hidden node problem.

4.2.1 Diagnosis Stage

The first diagnosis stage is to infer the asymmetric carrier sensing condition at each AP in the network. Based on above observations, we propose a simple interference inference algorithm, called CUBIA (*Two-Phase CUsum-Based Interference inference with Adaptive RTS/CTS*). CUBIA runs at each AP and it diagnoses the network condition by passively monitoring the ongoing traffic with its client nodes. Specifically, if CUBIA detects severe and persistent frame transmission failures, it employs the RTS/CTS exchange before each data transmission to rule out the possibility of the hidden node problem. For this, CUBIA employs the CUSUM (CUmulative SUM) algorithm [9] to quantify the duration of frame losses. CUSUM algorithm suits our needs because it is simple and light-weight, and it has been widely used for the detection of state changes. We will elaborate the detailed procedure of the detection mechanism using CUSUM algorithm.

CUBIA monitors the frame error rate (FER) for every m transmissions to detect any abrupt changes in network condition, which might indicate the possibility of selfish tethering nodes. Let p_k^i denote the frame error rate for the k-th measurement period at AP_i, given by $p_k^i = e_k/m$, where e_k denotes the number of transmission failures. Then, we calculate the average error probability, $E[p^i]$, by a moving average to reflect the network dynamics as:

$$E[p^i] = \lambda \cdot E[p^i] + (1 - \lambda) \cdot p_k^i. \tag{3}$$

We define the CUSUM change detection filter c_k^i of AP_i as:

$$c_k^i = \max(\ 0,\ c_{k-1}^i + p_k^i - v\ ), \tag{4}$$
$$c_0^i = 0,$$

where v is a drift parameter, which is a filter design parameter. v is configured differently according to the value of c_k^i as:

$$v = \begin{cases} E[p^i] + \mathcal{T}_{FER}, & \text{if } c_k^i < \theta_{AS} \\ \mathcal{T}_{FER}, & \text{if } \theta_{AS} \le c_k^i \le \theta_S, \end{cases} \tag{5}$$

where θ_{AS} and θ_S denote the *first alarm threshold* for asymmetric carrier sensing, and the *second alarm threshold* for inferring selfish carrier sensing, respectively.

CUBIA issues the first alarm to AP_i when $c_k^i > \theta_{AS}$. Then, CUBIA turns on the RTS/CTS exchange mechanism, and keeps track of the change detection with $v = \mathcal{T}_{FER}$ in Eq. (4). Note that the first alarm can be caused by a sudden severe MAC-layer congestion or a hidden node problem. However, in such a case, the magnitude of c_k^i tends to decrease with RTC/CTS frames, because the the collisions are filtered out [12] and the hidden terminal problem are mitigated with RTS/CTS exchange, and the frame error rate decreases accordingly.

On the other hand, if the first alarm is caused by the selfish carrier sensing problem, the transmission of a frame transmission following successful RTS/CTS exchanges will be interfered with, and hence, even with RTS/CTS, the magnitude of c_k^i would continue to increase, even beyond θ_S. Recall that under the normal condition, PHY-layer link quality is stable and has a certain upper bound $\mathcal{T}_{FER}$, namely *target* frame error rate, which is guaranteed by the underlying rate-adaptation scheme that adjusts the modulation schemes to meet the target FER.[5]

Consequently, AP_i sends a second alarm to the central controller (see Fig. 1) if $c_k^i > \theta_S$, which may trigger the *reaction* to solve the selfish carrier sensing problem.

5. For example, practical rate adaptations [10] adjust the modulation schemes to meet the target FER (frame error rate), and thus guarantee the average FER performance to be maintained around the target value.

4.2.2 Reaction Stage

Although the focus of this paper is to propose an AP-level detection mechanism, it is important to cope with the selfish misbehavior detected whin the network at the system level. Here, we discuss how the controller determines when to take follow-up actions to cope with the selfish misbehavior detected whin the network.

In typical managed Wi-Fi networks, the information obtained at APs is integrated on the controller. Then, the controller utilizes the information to improve the detection accuracy. As explained in observation **O2**, majority of nearby APs will likely to experience the similar severe packet errors simultaneously when the selfish tethering is present. Consequently, the victim APs send the second alarm to the controller at the same time. By exploiting the spatial and temporal correlation in those alarms, the controller identifies the root of the problem more effectively and accurately. For example, if a certain condition is satisfied, the controller can take various follow-up actions, which can be (i) localizing the rogue interfering node [11], and (ii) remedying victim APs (e.g., interference-aware channel reassignment), etc. However, the detailed follow-up actions are beyond the scope of this paper.

5 Conclusion and Future Work

In this paper, we present CUBIA, a novel Wi-Fi tethering misbehavior detection mechanism that can accurately detect selfish behavior, e.g., manipulating CCA threshold, via AP-level collaboration in multi-AP network environments. We show that the benefits of MIM can be fully exploited and abused further by a selfish tethering node via CCA manipulation combined with its short link-distance. We also observe that the consequence of the selfish tethering behavior resembles that of the hidden node problem, but selfish tethering nodes tend to ignore the RTS/CTS mechanism. CUBIA employs CUSUM algorithm for online detection of abnormal network behavior and inject RTS/CTS frames to avoid mis-diagnosing the hidden node problem as a selfish tethering. Our simulation-based evaluation results show that CUBIA accurately distinguishes the selfish tethering behavior from other types of misbehavior including the hidden node problem.

In future, we would like to extend our work to pinpoint and localize the selfish node based on the cooperation among APs. It would also be interesting to study effective system-level follow-up actions against selfish tethering misbehavior, such as jamming-resilient dynamic channel re-assignment.

References

[1] A. Akella, S. Seshan, R. Karp, and S. Shenker, "Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP," in *Proc. ACM SIGCOMM*, August 2002.

[2] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," *in Proc. ACM IMC*, November 2009.

[3] N. BenSalem, L. Buttyan, J.P. Hubaux, and M. Jakobsson, "A Charging and Rewarding Scheme for Packet Forwarding in Multihop Cellular Networks," in *Proc. ACM MobiHoc*, June 2003.

[4] J. Choi, and K. G. Shin, "Out-of-band sensing with ZigBee for dynamic channel assignment in on-the-move hotspots," *in Proc. IEEE ICNP*, October 2011.

[5] J. Choi, and K. G. Shin, "QoS provisioning for large-scale multi-AP WLANs," *Elsevier Ad-hoc Networks*, vol. 10, no. 2, pp. 174-185, March 2012.

[6] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017," February 2013.

[7] H. Dwivedi, C. Clark, and D. Thiel, *Mobile application security*, McGraw-Hill Education, 2010.

[8] M. Garetto, T. Salonidis, and E. W. Knightly, "Modeling per-flow throughput and capturing starvation in csma multi-hop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 864-877, August 2008.

[9] F. Gustafsson, Adaptive filtering and change detection. *John Wiley & Sons, Ltd*, 2000.

[10] T. Jensen, S. Kant, J. Wehinger, and B. Fleury, "Fast link adaptation for MIMO OFDM," *IEEE Trans. Veh. Tech.*, vol. 59, no. 8, pp. 3766-3778, October 2010.

[11] K. Joshi, S. Hong and S. Katti "PinPoint: Localizing interfering radios" *in Proc.* 10th USENIX NSDI, April 2013.

[12] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: Collision-aware rate adaptation for ieee 802.11 WLANs," *in Proc.* IEEE INFOCOM, April 2006.

[13] P. Kyasanur and N.H. Vaidya, "Selfish MAC layer misbehavior in wireless networks," *IEEE Trans. Mob. Comp.*, vol. 4, no. 5, pp. 502-516, September/October 2005.

[14] J. Lee, W. Kim, S. Lee, D. Jo, J. Ryu, T. T. Kwon, and Y. Choi, "An experimental study on the capture effect in 802.11a networks," *in Proc.* ACM WinTECH, August 2007.

[15] K. Leentvaar and J. Flint, "The capture effect in FM receivers," *IEEE Trans. on Comm.*, vol. 24, no.5, pp. 531-539, May 1976.

[16] A. MacKenzie and S. Wicker, "Game theory and the design of selfconfiguring, adaptive wireless networks," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 126-131, November 2001.

[17] S. Marti, T.J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," in *Proc. ACM MobiCom*, August 2000.

[18] J. Manweiler, N. Santhapuri, S. Sen, R. R. Choudhury, S. Nelakuditi, and K. Munagala, "Order matters: Transmission reordering in wireless networks," *IEEE/ACM Trans. on Net. (ToN)*, vol. 20, no. 2, pp. 353-366, April 2012.

[19] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for WLANs," *Mobile Computer Communications Review (MC2R)*, vol. 9, no. 3, pp. 19-31, July 2005.

[20] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. "A client-driven approach for channel management in wireless LANs," *in Proc.* IEEE INFOCOM, April 2006.

[21] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh, "Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage," *in Proc.* ACM/USENIX IMC, October 2005

[22] A. Mishra, V. Shrivastava, D. Agarwal, and S. Banerjee, "Distributed channel management in uncoordinated wireless environments," *in Proc.* ACM MOBICOM, September 2006.

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

[24] S. Radosavac, J. Baras, and I. Koutsopoulos, "A framework for MAC protocol misbehavior detection in wireless networks," *in Proc.* ACM WiSe, September 2005.

[25] M. Raya, I. Aad, J.-P. Hubaux, and A. E. Fawal, "DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots," *IEEE Trans. Mobile Comp.*, vol. 5, no. 12, pp. 1691-1705, December 2006.

[26] J. Tang, and Y. Cheng, "Selfish Misbehavior Detection in 802.11 Based Wireless Networks: An Adaptive Approach Based on Markov Decision Process," *in Proc.* IEEE INFOCOM, April 2013.

[27] A. Toledo, and X. Wang, "Robust Detection of Selfish Misbehavior in Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 1124-1134, August 2007.

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

[29] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz, "Unfairness and capture behaviour in 802.11 adhoc networks," *in Proc.* IEEE ICC, June 2000.

[30] E. Yang, J. Choi, and S. Lee, "On selfish behavior using asymmetric carrier sensing in IEEE 802.11 wireless networks," *in Proc.* IEEE LCN, October 2008.

[31] X. Zhang and K. G. Shin, "Adaptive subcarrier nulling: Enabling partial spectrum sharing in wireless lans," *in Proc.* IEEE ICNP, October 2011.

[32] Y. Zhang and L. Lazos, "Countering Selsh Misbehavior in Multichannel MAC Protocols," *in Proc.* IEEE INFOCOM, April 2013.

[33] The Network Simulator-ns2, www.isi.edu/nsnam/ns.

[34] Wiggle: Wireless geographic logging engine, http://www.wigle.net/.

[35] Google Android, http://www.android.com/.