*Article*

# Detection of Misconfigured Wi-Fi Tethering in Managed Networks

**Jaehyuk Choi** [1,†]

[1] Department of Software, Gachon University; jchoi@gachon.ac.kr
[*] Correspondence: jchoi@gachon.ac.kr; Tel.: +82-31-750-8657
[†] Current address: 1342 Seongnamdaero, Sujeong-gu, Seongnam-si, Gyeonggi-do, Korea.

**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 or misconfigured tethering device interfering with nearby well-planned access points (APs). In this paper, we demonstrate 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 this problem, we present a passive online detection scheme that identifies the network condition and detects selfish tethering nodes with high accuracy by exploiting the packet loss information of on-going transmissions. To the best of our knowledge, this is the first research to consider the problem of detecting a selfish tethering node in managed Wi-Fi networks.

**Keywords:** spectrum sharing; network monitoring; Wi-Fi tethering; sensing of misbehavior

---

## 1. Introduction

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 several Gbps of throughput for 802.11ac and 802.11ax [1,2], 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 [3].

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 [4]. 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, 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 jailbreaking 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. Thus, it is essential to design an efficient mechanism to detect unauthorized and misconfigured Wi-Fi tethering.

In this paper, we investigate the impact of misconfigured Wi-Fi tethering on other users' throughput performance in a multi-access point (AP) environment in the presence of a selfishly misconfigured tethering device. 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. Note that the dynamic CCA adjustment is considered as a potential means to manage channel access and in the coming 802.11ax standard (aka Wi-Fi 6) [2]. 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.

To this end, we present an online 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 ongoing data traffic to detect any abnormal changes caused by target selfish tethering. In particular, it monitors any noticeable increase in frame loss rate and identifies 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.

**Contributions.** This paper makes several contributions as follows.

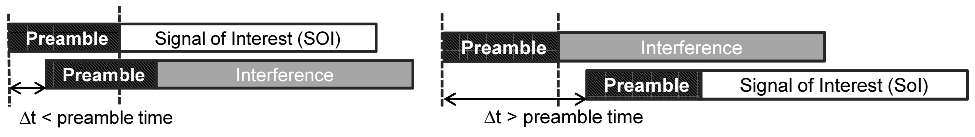
- We observe that concurrent transmission mechanisms, e.g., PHY capture or MIM, can be abused by selfish nodes, and study the impact of the selfish Wi-Fi tethering behavior on the throughput performance of other nearby well-behaving nodes.
- We propose a selfish misbehavior detection mechanism, called `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.

**Organization.** The remainder of paper is organized as follows. We summarize related research work in Section 2. Section 3 describes the system model and 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. Related Work and Preliminaries

### *2.1. Related Work*

Selfish and malicious misbehavior in wireless 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 [5–8]. In [5], 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.* [6] investigated multiple misbehavior policies in 802.11 MAC protocol including backoff manipulations,



(a) PHY capture effect (b) MIM (Message-In-Message)

**Figure 1.** (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.

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 [7,8]. 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. The authors in [9] 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 coexistence between LTE and Wi-Fi systems in unlicensed bands [10], routing layers [11,12], multi-channel protocols [13], and game theory-based behaviors [14,15].

The problem of misbehavior using the unfair coexistence between cellular and Wi-Fi system is relatively new and unexplored in the literature. In [10], the problem of detecting misbehaving of LTE/Wi-Fi has been addressed. In [16], the authors have identified the effect of an 802.11 node's selfish behavior by manipulating CCA threshold. They have shown that the selfish behavior can achieve higher throughput than other well-behaving nodes based on a game-theoretical model. The authors in [17] 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.

## *2.2. Channel Model and Assumptions*

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 [18]:

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

where $\alpha$ 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 $\gamma_m$ denote the minimum SIR requirement (i.e., SIR threshold) for successful reception at a receiver node (i.e., $SIR_r > \gamma_m$) under modulation scheme $m$, where we consider multiple data rates with various modulation and coding scheme (MCS) values as in 802.11n/ac.

Even when multiple independent transmissions occur simultaneously to a receiver $r$, the receiver can successfully capture *the signal of interest* (SoI) thanks to two well-known concurrent transmission technologies: PHY capture effect [19] and message-in-message (MIM) [20,21].

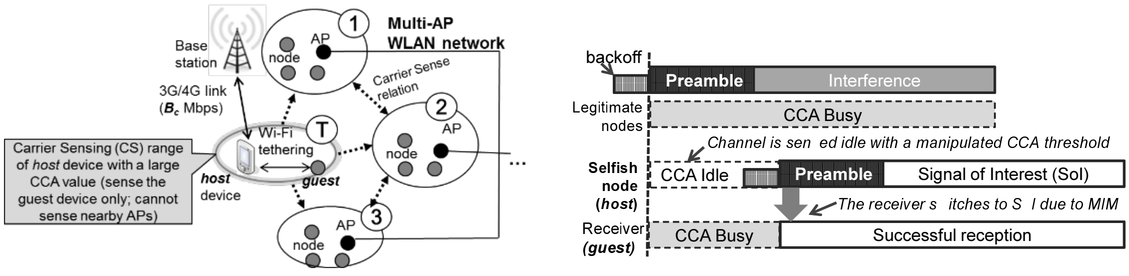
Fig. 1 illustrates the main difference between PHY capture and MIM. The PHY capture effect is the property of 802.11 radios, where an SoI can be decoded successfully even when the interference arrives at the same time, as long as the overlap *within* the preamble detection stage and the SoI is stronger 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*, $\beta_m$, for modulation scheme $m$.

Following the experimental results in [20], through the paper, we configure the SIR threshold $\gamma_m$ and MIM threshold $\beta_m$ for a given modulation scheme $m$ (e.g., $m$=BPSK, QPSK, 16QAM, and 64QAM), and assume that the receiver can decode a frame successfully when the received $SIR_r$ is consistently above $\beta_m$ even when multiple independent transmissions occur simultaneously.

## 3. Selfish Configuration in Wi-Fi Tethering

In this section, we first introduce the system model, including the adversary model and the network model. Then, we present the problem illustrating the impact of a selfish Wi-Fi tethering system using the CCA manipulation in a managed multi-AP environment.

### 3.1. System Model



(a) (b)

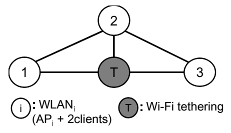
**Figure 2.** (a) Illustration of the problem: selfish misconfigured Wi-Fi tethering sets up the network in a centrally managed multi-AP network; (b) Adversary model: selfish behavior with CCA manipulation will not freeze its back-off counter even if other nodes are transmitting.

We consider a scenario where an unauthorized Wi-Fi tethering system creates a Wi-Fi hotspot within a managed multi-AP 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. 2.[1] The multi-AP network is monitored and managed by the controller. Each AP monitors the channel access activity and reports the information to the controller periodically.

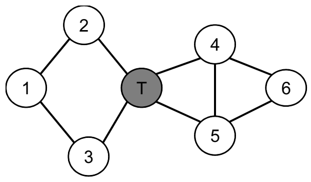
The tethering consists of an Internet-connected (e.g., through 4G or 5G cellular connection) mobile phone and a tethered Wi-Fi-enabled device contending for channel access with nearby Wi-Fi systems. We will henceforth refer to the mobile device and the tethered Wi-Fi device as *host* and *guest* nodes, respectively; the *host* node shares its cellular Internet connection with the *guest* nodes via its Wi-Fi interface.

We assume that an adversary user manipulates the CCA threshold of the *host* node's Wi-Fi interface in the tethering whereas the *guest* nodes (e.g., laptop, tablet, or etc.) are legitimate, i.e., *guest* nodes use the default CCA threshold. The CCA threshold controls the sensitivity of carrier sense, that

[1] Note that fully managed or cloud-managed Wi-Fi network solutions with central controllers, e.g., Aerohive, Aruba, Cisco Meraki and Extreme, have become a common practice since they provide fast, seamless and secure wireless connectivity across the organization.

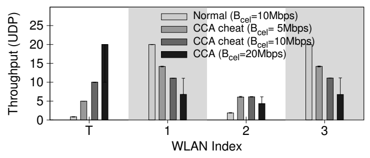


(a) Topology-1

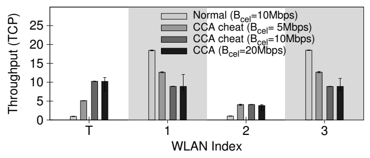


(b) Topology-2

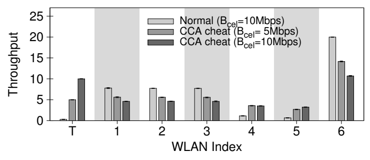
**Figure 3.** AP Interference graph of two simulated topologies.



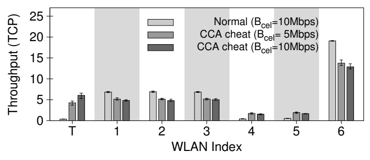
(a) Topology-1, UDP



(b) Topology-1, TCP



(c) Topology-2, UDP



(d) Topology-2, TCP

**Figure 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.

is, a node with a higher CCA threshold can access channel with a higher interference/noise level [22]. The channel is considered to be idle when the sensed energy is lower than the CCA threshold. 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 nodes.

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 [23]).

### 3.2. 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 a network simulator [24].

**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 protocol by increasing the CCA threshold as high as possible without losing

[2] We sampled the topologies from a popular Wi-Fi database *Wigle* [25]. For example, the topology shown in Fig. 3(a) corresponds to the well-known FIM (Flow-In-the-Middle) topology [26], which can be easily found in real-world AP deployments [27]. 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.

the connectivity to the guest. We used the following settings in our simulations. Each Wi-Fi has an AP and two associated client nodes (i.e., 2 AP–clients flows per Wi-Fi), and all nodes operate on the same channel. We considered typical IEEE 802.11n MAC/PHY parameters with the maximum speed of 54 Mbps, and set the capacity $B_{cel}$ of the cellular 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 [26]. 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.

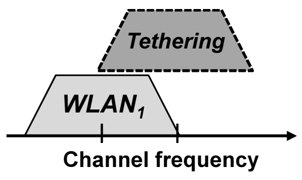
One can expect that selfish CCA manipulation would increase the collision probability of the tethering link since the *host* node initiates packet transmission even in the present of other nodes' transmissions. Nevertheless, we observe a significant gain of selfish behavior using CCA manipulation, as 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)

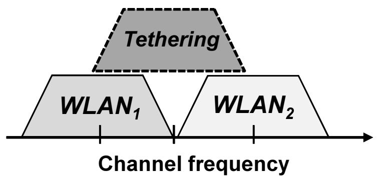
| **Topology** | Collision prob. | Capture effect | MIM |
|---|---|---|---|
| 1 | 71.5 | 17.9 | 63.6 |
| 2 | 91.8 | 28.3 | 63.5 |

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.

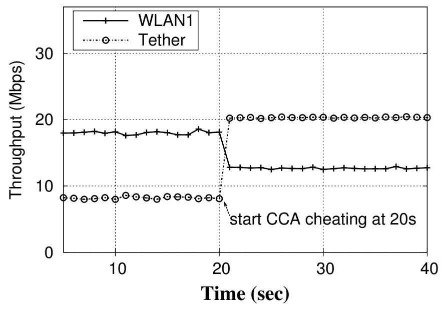
**Impact of Tethering Channel**: Next, we study the impact of the CCA manipulation on performance when tethering is launched on a channel partially overlapping with nearby APs. As mentioned above, the tethered Wi-Fi hotspot can potentially set up the network with an *arbitrary* channel number, which may cause serious interference to nearby well-planned APs. We used two simple scenarios in [28] and the channel model presented in [29] 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 [28]. 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



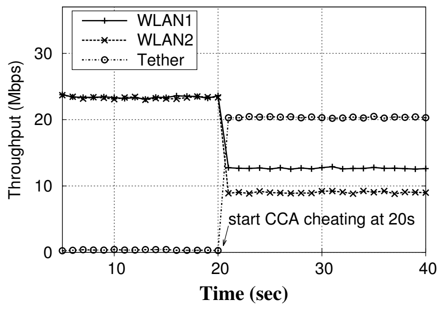
(a) Scenario 1



(b) Scenario 2



(c) Throughput for Scenario 1



(d) Throughput for Scenario 2

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

channel starvation [28]; 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.

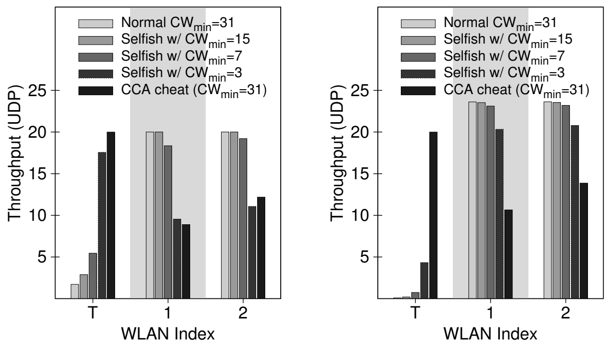
To evaluate the impact of the selfish behavior using CCA manipulation in these scenarios, we performed simulations with following setting. Each Wi-Fi consists of two client nodes where the traffic on each flow is generated with 10 Mbps downlink CBR over UDP. The capacity $B_{cel}$ for tethering is configured 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 [28]. 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 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

[3] The manipulation of backoff mechanism is widely adopted by selfish users [5].



(a) 2 client nodes per WLAN (b) 5 client nodes per WLAN

**Figure 6.** Throughput comparison with selfish configurations of the $CW_{min}$ parameter.

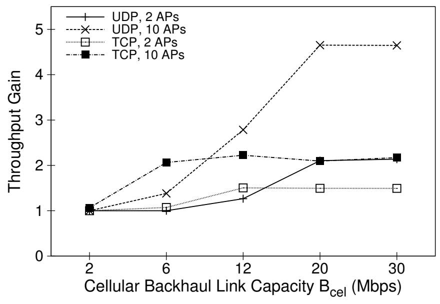


**Figure 7.** Throughput gain of selfish carrier sensing over various cellular backhaul link capacities and AP densities.

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. A Proposed Online Detection Algorithm

The symptoms of selfish tethering resembles those of the well-known hidden node problem—both result in excessive frame loss of nearby legitimate APs. Thus, successful detection of selfish misbehavior depends largely on a AP's ability to identify the root cause of its frame losses. Note that in 802.11 Wi-Fi, there are four main causes of frame losses: (i) PHY-layer link quality degradation, (ii) MAC-layer collision, (iii) hidden nodes, and (iv) selfish misbehavior (e.g., manipulation of the CCA thresholds).

Fortunately, selfish tethering exhibits unique features that can facilitate distinguishing the selfish carrier sensing of a tethering host from other causes of frame losses, especially from the hidden node problem. The main difference is that the frame losses at legitimate nodes caused by selfish tethering cannot be easily resolved by the RTS/CTS mechanism. This is because the selfish nodes may not recognize the RTS/CTS frames owing to their manipulated CCA thresholds (i.e., their short sensing ranges) or RTS/CTS frames can be intentionally ignored by the selfish nodes.

Based on this observation, we propose a simple, yet efficient online detection algorithm, called `CUBIA` (*CUsum-Based Interference inference with Adaptive RTS/CTS*) that can accurately distinguish selfish behavior with CCA manipulation from other types of network problems, such as severe network congestion (i.e., collisions) and the hidden node problem. `CUBIA` operates at each AP and diagnoses the network condition by passively monitoring the ongoing traffic with its client nodes. Specifically, if

[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.

CUBIA detects severe and persistent frame transmission failures, it employs the RTS/CTS exchange before each data transmission to eliminate the possibility of the hidden node problem. For this, CUBIA employs the CUSUM (CUmulative SUM) algorithm [30] to quantify the duration of the frame losses. CUSUM algorithm suits our needs because it is simple and light-weight and has been widely used for the detection of state changes. In the following, we describe the detailed procedure of the detection mechanism using the CUSUM algorithm.

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

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

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

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

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

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

where $\theta_{AS}$ and $\theta_S$ denote the *first alarm threshold* for asymmetric carrier sensing and the *detection (second) alarm threshold* for inferring selfish carrier sensing, respectively. CUBIA issues the *first* alarm to the AP when $c_k > \theta_{AS}$. Then, CUBIA adaptively activates the RTS/CTS exchange mechanism and continues to track the change detection with $v = \mathcal{T}_{FER}$ in Eqs. (4) and (5).

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 $c_k$ tends to decrease with RTC/CTS frames because the collisions are filtered out [31] and the hidden terminal problem are mitigated with RTS/CTS exchange. The frame error rate decreases accordingly.

Conversely, 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 $c_k$ would continue to increase, even beyond $\theta_S$. Recall that under a normal condition, the PHY-layer link quality is stable and has a certain upper bound $\mathcal{T}_{FER}$, namely, the *target* FER, which is guaranteed by the underlying rate-adaptation scheme that adjusts the modulation schemes to meet the target FER (e.g., $\mathcal{T}_{FER}$ = 0.05 is used in the evaluation).[5]

Consequently, the AP issues a *detection* alarm to the central controller in the network (see Fig. 2) if $c_k > \theta_S$, which may trigger a follow-up action at the controller level to solve the selfish carrier sensing problem within the managed network.

**Remark.** 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 briefly discuss how the controller determines when to take follow-up actions to cope with the selfish misbehavior detected within 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. When the selfish tethering is present, majority of nearby APs will likely to

[5] For example, practical rate adaptations [32] 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.

experience the similar severe packet errors simultaneously. Consequently, the victim APs send the detection 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 [33], 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. Performance Evaluation

We now evaluate the performance of the proposed detection algorithm via simulation. We have implemented the proposed `CUBIA` in ns simulator [24].

### 5.1. Simulation Setup

In the simulation, the multi-AP network is deployed in a $200 \times 200\ m^2$ area, where 5 APs with 10 client nodes are randomly generated; this represents a densely-populated configuration fully covering the entire area. The transmission range and carrier sensing range of legitimate nodes are set to 75 m, and 150 m, respectively. A selfish tethering pair is placed at the center of the area, whose link distance and carrier sensing are set to 1 m and 10 m, respectively. Table 2 lists the parameter values used in the simulation study.

**Table 2.** Parameters used in performance evaluation

| Parameter | Value |
|---|---|
| Transmission range | 75 |
| Carrier range | 150 |
| Data rate / ACK rate | 54 Mbps / 6 Mbps |
| CBR rate per AP-client pair (UDP) | 20 Mbps |
| payload size of UDP , TCP | 1000, 1500 bytes |
| $\mathcal{T}_{FER}$ (in Eq. (5)) | 0.05 |
| $\lambda$ (in Eq. (3)) | 0.1 |
| $m$ (number of transmissions for $p_k$ in Eq. (3)) | 10 |
| the maximum allowed latency of detection | 2 sec |

The performance is evaluated in terms of detection accuracy and time for TCP and UDP protocols. We consider a downlink scenario where each AP transmits frames to its client nodes.

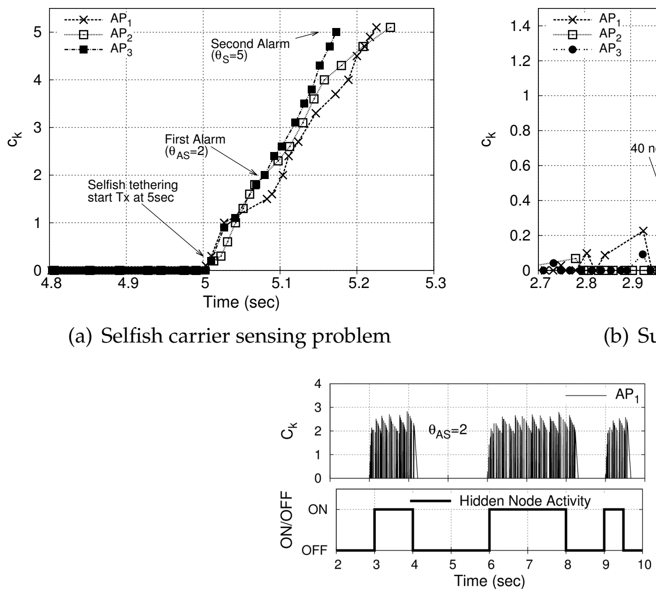
### 5.2. Detection Performance

#### 5.2.1. Accuracy of Frame Loss Differentiation

To demonstrate the efficacy of `CUBIA` in distinguishing the selfish carrier sensing problem, we consider three testing scenarios: (i) selfish carrier sensing, (ii) hidden node problem, and (iii) collisions.

Fig. 8 shows the temporal behavior of CUSUM change detection filter $c_k$ for the three testing scenarios. Fig. 8(a) plots the results of `CUBIA` over time for the case of selfish problem in the topology depicted in Fig. 5(b). We can observe that the detection filters of APs in the interference range continues to increase, where the selfish node starts transmissions at 5 s.

Figs. 8(b) and 8(c) show `CUBIA`'s ability of filtering the collisions and the hidden node problem, respectively. To simulate an abrupt change in the network state, initially, 10 AP–clients pairs are considered until additional 40 nodes are abruptly activated at 3 second. Fig. 8(b) demonstrates that `CUBIA` effectively filters out the MAC-layer collisions. To simulate the impact of the hidden node problem, we generated ON/OFF traffic on a hidden node, as illustrated in Fig. 8(c). In the figure, we can see that the detection filter of `CUBIA` promptly reacts to the hidden node, increasing above $\theta_{AS}$,



(a) Selfish carrier sensing problem

(b) Sudden collisions

(c) Hidden node problem

**Figure 8.** The dynamics of `CUBIA` toward three difference types of frame losses.

but the value of $c_k$ remains low and does not exceed $\theta_S$. This is because the effect of a hidden node is mitigated by adaptive RTS/CTS changes, thus imparting a negative drift to the detection filter.

Overall, our proposed scheme `CUBIA` accurately distinguishes the selfish misbehavior with CCA manipulation from collisions and the hidden node problem.

### 5.2.2. Detection Performance

We now evaluate the detection performance of `CUBIA`. It is important to meet the detectability requirements, such as the maximum allowed latency of detection. In this simulation study, we assume that the maximum allowed latency of detection is 2 seconds, that is, if the selfish misbehavior is not detected within 2 seconds, we consider this case as a mis-detection. In practice, the detection latency requirement can be adjusted based on specific needs/conditions of the network and protocol stack. For example, an extended exposure to selfish carrier sensing can cause the TCP congestion control and fast recovery algorithms to kick in, which make it very difficult to recover the throughput performance. There also exists a tradeoff between the detection sensitivity and the false-alarm rate, which is part of our future work. We run the simulation 150 times for each set of tests with various backhaul link capacities $B_{cel}$ of selfish tethering, and alarm thresholds.

Tables 3 and 4 show the detection results for UDP and TCP protocols, respectively. We can observe that our scheme can detect the selfish behavior with high backhaul capacity with very high accuracy. Note that $B_{cel}$ implies the intensity of selfish behavior because the larger the $B_{cel}$, the more outstanding packets the selfish node can transmit, causing severe interference, as observed in Section 3.2. However, the results imply that in many cases, the detection decision takes more than 2 seconds with low selfish intensity, i.e., small $B_{cel}$. This is because the impact of such a moderately selfish node on the network performance is not significant, i.e., the selfish node achieves only a small throughput gain over the legitimate nodes. As a result, the moderate selfish node is not immediately detectable by `CUBIA` within 2 seconds, since it takes more samples for the AP to accurately detect such selfish nodes. Fig. 9 shows the average throughput degradation ratio of well-behaving nodes due to the selfish node for various values of selfish intensity (i.e., $B_{cel}$). The figure implies that a higher selfish intensity incurs a severer interference on well-behaving nodes. The throughput degradation can be ignored when the backhaul link capacity $B_{cel}$ of the selfish node is small.

**Table 3.** Detection performance for the UDP protocol.

| $(\theta_{AS}, \theta_S)$ \ $B_{cel}$ | *20 Mbps* | *10 Mbps* | *5 Mbps* | *2 Mbps* |
|---|---|---|---|---|
| (2, 3) | 1.00 | 1.00 | 0.99 | 0.73 |
| (2, 4) | 1.00 | 1.00 | 0.99 | 0.74 |
| (2, 5) | 1.00 | 1.00 | 0.97 | 0.61 |
| (3, 4) | 1.00 | 0.94 | 0.05 | 0.00 |
| (3, 5) | 1.00 | 0.93 | 0.07 | 0.00 |
| (3, 6) | 0.99 | 0.92 | 0.07 | 0.00 |
| (4, 5) | 1.00 | 0.27 | 0.00 | 0.00 |
| (4, 6) | 1.00 | 0.33 | 0.00 | 0.00 |
| (4, 7) | 0.95 | 0.27 | 0.00 | 0.00 |

**Table 4.** Detection performance for the TCP protocol.

| $(\theta_{AS}, \theta_S)$ \ $B_{cel}$ | *20 Mbps* | *10 Mbps* | *5 Mbps* | *2 Mbps* |
|---|---|---|---|---|
| (2, 3) | 1.00 | 1.00 | 1.00 | 0.99 |
| (2, 4) | 1.00 | 1.00 | 1.00 | 0.47 |
| (2, 5) | 1.00 | 1.00 | 1.00 | 0.03 |
| (3, 4) | 1.00 | 1.00 | 0.99 | 0.80 |
| (3, 5) | 1.00 | 1.00 | 0.98 | 0.28 |
| (3, 6) | 1.00 | 1.00 | 0.93 | 0.03 |
| (4, 5) | 0.87 | 0.86 | 0.11 | 0.01 |
| (4, 6) | 0.84 | 0.84 | 0.07 | 0.00 |
| (4, 7) | 0.63 | 0.72 | 0.02 | 0.00 |

We next evaluate the impact of the backhaul link capacity $B_{cel}$ of selfish tethering on the detection performance for $B_{cel} = 2, 5, 10$, and 20 Mbps. Fig. 10 shows the cumulative distribution of detection time under various $B_{cel}$. The results indicate that more aggressive selfish behavior, i.e., higher $B_{cel}$, is detected more quickly by `CUBIA` for both TCP and UDP protocols. This indicates that `CUBIA` can quickly detect aggressive selfish behaviors, which is a very important design requirement for any good detection scheme since such an aggressive behavior can seriously degrade the performance of well-behaving nodes.

Finally, we study the impact of alarm thresholds, i.e., $\theta_{AS}$ and $\theta_S$, on the detection time. Fig. 11 depicts the distribution of detection time for 3 different values of the *first alarm threshold* $\theta_{AS}$. It is straightforward that the use of a smaller value of $\theta_{AS}$ may issue the first alarm too frequently and may incur an unnecessary overhead for RTS/CTS exchange, resulting in the performance degradation. Meanwhile, the results show that it takes less detection time with a smaller $\theta_{AS}$.

Similarly, the impact of the *second alarm threshold* $\theta_S$ can be seen in Fig. 12. The figure compares the detection time for different values of $\theta_S$ with the given $\theta_{AS} = 3$. We can see that the detection time increases proportional to $\theta_S$. However, there is a tradeoff between the false alarm ratio and the detection time according to the choice of $\theta_S$. For example, in case of the temporal behavior of the detection filter in Fig. 8(c), the use of a small value of $\theta_S$ can cause false alarms (if $\theta_S$ is set to be less than $\theta_{AS} + 1$, it would issue several false alarms.) although it can reduce the detection time unless the hidden node exists. Thus, it is recommended to use a value of $\theta_S$ larger than $\theta_{AS} + 1$, to avoid false alarms although it might take more time to detect.

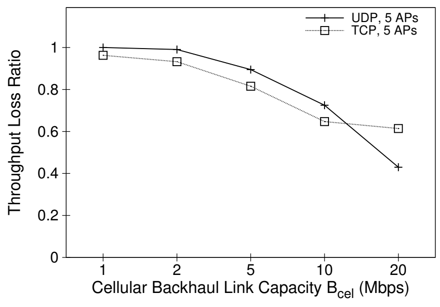


**Figure 9.** Impact of selfish intensity on the performance of well-behaving nodes.

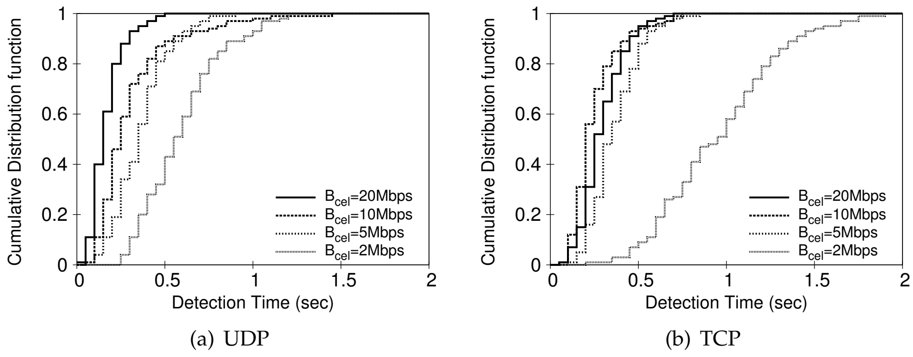


**Figure 10.** Impact of $B_{cel}$ on detection time.

## 6. 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.

**Funding:** This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2017R1D1A1B03034915.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

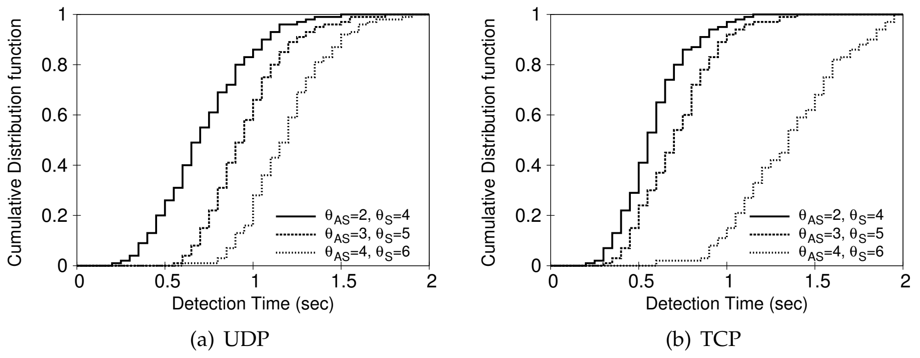


**Figure 11.** Impact of the first alarm threshold $\theta_{AS}$ on detection time for UDP and TCP protocols with $B_{cel}$ =20Mbps.

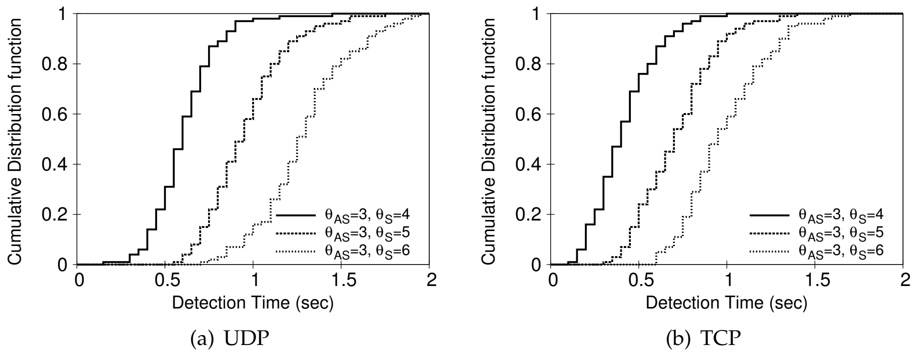


**Figure 12.** Impact of the second alarm threshold $\theta_S$ for the given $\theta_{AS}$=3 on detection time with $B_{cel}$ =20Mbps.

| | |
|---|---|
| AP | Access Point |
| CCA | Clear channel assessment |
| CUSUM | cumulative sum |
| FER | frame error rate |
| MIM | Message-In-Message |
| SIR | signal to interference ratio |
| SoI | the Signal of Interest |
| WLAN | Wireless LAN |
| $\gamma_m$ | *SIR threshold* for modulation scheme *m* |
| $\beta_m$ | *MIM threshold* for modulation scheme *m* |

## References

1. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016* (Revision of IEEE Std 802.11-2012) **2016**, pp.1-3534.
2. Khorov, E.; Kiryanov, A.; Lyakhov, A.; Bianchi, G. A Tutorial on IEEE 802.11ax High Efficiency WLANs. *IEEE Comm. Surv. & Tut.* **2019**, 21, pp. 197–216.
3. Hoffman, C.; Summerson, C. How to Tether Your Android Phone and Share Its Internet Connection with Other Devices. https://www.howtogeek.com/170302/the-htg-guide-to-tethering-your-android-phone/.
4. Doherty, J. Wireless and Mobile Device Security. *Jones & Bartlett Publishers* **2015**.
5. Kyasanur, P.; Vaidya,N.H. Selfish MAC layer misbehavior in wireless networks. *IEEE Trans. Mob. Comp.* **2015**, 4, pp. 502-516.

6. Raya, M.; Aad, I.; Hubaux, J.-P.; Fawal, A. E. DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots. *IEEE Trans. Mobile Comp.* **2006**, 5, pp. 1691-1705.
7. Radosavac, S.; Baras, J.; Koutsopoulos, I. A framework for MAC protocol misbehavior detection in wireless networks. *in Proc. ACM WiSe* **2005**, pp. 33–42.
8. Toledo, A.; Wang, X. Robust Detection of Selfish Misbehavior in Wireless Networks. *IEEE J. on Sel. Area. in Comm.* **2007**, 25, pp. 1124-1134.
9. Tang, J.; Cheng, Y. Selfish Misbehavior Detection in 802.11 Based Wireless Networks: An Adaptive Approach Based on Markov Decision Process. *in Proc.* IEEE INFOCOM **2013**, pp. 1357-1365.
10. Samy, I.; Lazos, L.;Xiao, Y.; Li, M.; Krunz, M. LTE misbehavior detection in Wi-Fi/LTE coexistence under the LAA-LTE standard. *in Proc. Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, **2018**, pp. 87–98.
11. BenSalem, N.; Buttyan, L.; Hubaux, J.P.; Jakobsson, M. A Charging and Rewarding Scheme for Packet Forwarding in Multihop Cellular Networks. *in Proc. ACM MobiHoc* **2003**, pp. 13–24.
12. Marti, S.; Giuli, T.J.; Lai, K.; Baker, M. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. *in Proc. ACM MobiCom* **2000**, pp. 255–265.
13. Zhang, Y.; Lazos, L. Countering selfish misbehavior in multi-channel MAC protocols. *in Proc. IEEE INFOCOM* **2013**, pp. 2787–2795.
14. MacKenzie, A.; Wicker, S. Game theory and the design of selfconfiguring, adaptive wireless networks. *IEEE Commun. Mag.*, **2001**, 39, pp. 126–131.
15. Akella, A.; Seshan, S.; Karp, R.; Shenker, S. Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP. *in Proc. ACM SIGCOMM* **2002**.
16. Yang, E.; Choi, J.; Lee, S. On selfish behavior using asymmetric carrier sensing in IEEE 802.11 wireless networks. *in Proc. IEEE LCN* **2008**, pp. 527–529.
17. Pelechrinis, K.; Yan, G.; Eidenbenz, S.; Krishnamurthy, S. V. Detecting Selfish Exploitation of Carrier Sensing in 802.11 Networks. *in Proc. IEEE INFOCOM* **2009**, pp. 657-665.
18. Tse, D.; Viswanath, P. *Fundamentals of Wireless Communications* **2005**, Cambridge University Press.
19. Leentvaar, K.; Flint, J. The capture effect in FM receivers. *IEEE Trans. on Comm.* **1976**, 24, pp. 531-539.
20. Lee, J.; Kim, W.; Lee, S.; Jo, D.; Ryu, J.; Kwon, T. T.; Choi, Y. An experimental study on the capture effect in 802.11a networks. *in Proc. ACM WinTECH* **2007**, pp. 19–26.
21. Manweiler, J.; Santhapuri, N.; Sen, S.; Choudhury, R.; Nelakuditi, S.; Munagala, K. Order matters: Transmission reordering in wireless networks. *IEEE/ACM Trans. on Net. (ToN)* **2012**, 20, pp. 353–366.
22. Wang, W.; Zhang, F.; Zhang, Q. Managing channel bonding with clear channel assessment in 802.11 networks. *in Proc. IEEE Int. Conf. on Comm. (ICC)* **2016**, pp. 1–6.
23. Choi, J.; Shin, K. G. Out-of-band sensing with ZigBee for dynamic channel assignment in on-the-move hotspots. *in Proc. IEEE ICNP* **2011**, pp. 216–225.
24. The Network Simulator-ns2, www.isi.edu/nsnam/ns.
25. Wiggle: Wireless geographic logging engine. http://www.wigle.net/.
26. Garetto, M.; Salonidis, T.; Knightly, E. W. Modeling per-flow throughput and capturing starvation in csma multi-hop wireless networks. *IEEE/ACM Trans. Netw.* **2008**, 16, pp. 864–877.
27. Choi, J.; Shin, K. G. QoS provisioning for large-scale multi-AP WLANs. *Elsevier Ad-hoc Networks* **2012**, 10, pp. 174–185.
28. Zhang, X.; Shin, K. G. Adaptive subcarrier nulling: Enabling partial spectrum sharing in wireless lans. *in Proc. IEEE ICNP* **2011**, pp. 311-320.
29. Mishra, A.; Rozner, E.; Banerjee, S.; Arbaugh, W. Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. *in Proc. ACM/USENIX IMC* **2005**, pp. 29.
30. Gustafsson, F. Adaptive filtering and change detection. *John Wiley & Sons, Ltd* **2000**.
31. Kim, J.; Kim, S.; Choi, S.; Qiao, D. CARA: Collision-aware rate adaptation for ieee 802.11 WLANs. *in Proc. IEEE INFOCOM* **2006**, pp. 1–11.
32. Jensen, T.; Kant, S.; Wehinger, J.; Fleury, B. Fast link adaptation for MIMO OFDM. *IEEE Trans. Veh. Tech.* **2000**, 59, pp. 3766–3778.
33. Joshi, K.; Hong, S.; Katti, S. PinPoint: Localizing interfering radios. *in Proc. 10th USENIX NSDI* **2013**, pp. 241–254.