# FIJI: Fighting Implicit Jamming in 802.11 WLANs

Ioannis Broustis[⋆], Konstantinos Pelechrinis[⋆], Dimitris Syrivelis[‡],
Srikanth V. Krishnamurthy[⋆], Leandros Tassiulas[‡]
[⋆]University of California, Riverside, [‡]University of Thessaly
{*broustis, kpele, krish*}*@cs.ucr.edu*, {*jsyr, leandros*}*@inf.uth.gr*

**Abstract.** The IEEE 802.11 protocol inherently provides the same long-term throughput to all the clients associated with a given access point (AP). In this paper, we first identify a clever, low-power jamming attack that can take advantage of this behavioral trait: *the placement of a low-power jammer in a way that it affects a single legitimate client can cause starvation to all the other clients*. In other words, the *total throughput* provided by the corresponding AP is drastically degraded. To fight against this attack, we design FIJI, a cross-layer anti-jamming system that detects such intelligent jammers and mitigates their impact on network performance. FIJI looks for anomalies in the AP load distribution to efficiently perform jammer detection. It then makes decisions with regards to *optimally* shaping the traffic such that: (a) the clients that are not explicitly jammed are shielded from experiencing starvation and, (b) the jammed clients receive the maximum possible throughput under the given conditions. We implement FIJI in real hardware; we evaluate its efficacy through experiments on a large-scale indoor testbed, under different traffic scenarios, network densities and jammer locations. Our measurements suggest that FIJI detects such jammers in real-time and alleviates their impact by allocating the available bandwidth in a fair and efficient way.

**Key words:** IEEE 802.11 WLANs, Fairness, Jamming, Measurement.

## 1 Introduction

The proliferation of IEEE 802.11 WLANs makes them an attractive target for malicious attackers with jamming devices [1, 2]. A jammer typically emits electromagnetic energy thereby causing: *(a)* prolonged packet collisions at collocated devices, and *(b)* packet transmission deferrals due to legitimate nodes detecting continuous medium activity. Hence, jamming attacks can lead to significant throughput degradation, especially when they intelligently exploit the properties of the MAC protocol in use.

In this paper, we first identify a clever jamming attack where the jammer can not only hurt its intended victim, but cause starvation to other clients that are associated with the same AP as the victim. We call this attack the *Implicit-Jamming* attack. We design and implement FIJI, a cross-layer anti-jamming system to effectively detect such jammers and mitigate the impact of their attack.

[⋆] This work was done partially with support from the US Army Research Office under the Multi-University Research Initiative (MURI) grants W911NF-07-1-0318 and the NSF NeTS:WN / Cyber trust grant 0721941.

**The implicit-jamming attack:** An inherent characteristic of the IEEE 802.11 MAC protocol is that under saturated traffic demands, an AP (access point) will provide the *same* long-term throughput to all of its affiliated clients [3]. If a client cannot receive high throughput from its AP for any reason (e.g. long-distance AP→client link or high levels of interference at the client side), the AP will spend a large amount of time serving this client at a low transmission bit-rate; this rate is determined by the rate adaptation algorithm in use. This will compel the AP to serve each of its other "healthier" clients (to which it can support higher transmission rates) for smaller periods. In other words, the AP does not distinguish between clients with low-SINR links and clients with high-SINR links; the long times taken to serve the former class of clients hurts the time available to serve the latter class of clients. This behavior is referred to as *the performance anomaly* of 802.11 [4] and is caused by the inherent design principles of the IEEE 802.11 MAC protocol (described in more detail in section 2).

The implicit jammer exploits this anomaly. To illustrate, consider the scenario depicted in Fig. 1. In this scenario: *(a)* all clients have high-SINR links with their AP in benign conditions, and *(b)* a low power jammer is placed next to a particular client (say client $C$) such that it does not *directly* affect any other client of the AP. The jammer causes high levels of interference at client $C$ and thus, most of the packets sent by the AP to $C$ are not successfully received. This in turn causes the AP to reduce the transmission rate used to serve $C$ (an inherent property of rate adaptation). As a result, the AP spends more time attempting to serve $C$, and this reduces the fraction of time that it provides to its other clients. Thus, the throughput of all the clients drops significantly due to the jamming of only client $C$. In other words, jamming a small subset of clients (even only a single client) implicitly affects all the clients that are affiliated with the same AP.

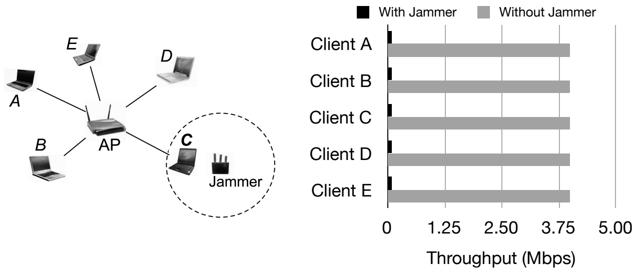

**Fig. 1. Implicit Jamming:** The jammer takes advantage of the 802.11 performance anomaly. Using very low transmission power, it simply attacks client $C$. This is sufficient to tremendously degrade the throughput of all clients.

**The impact of the implicit-jamming attack:** In order to demonstrate the potential impact of this attack on the performance of the network, we conduct a set of preliminary experiments on our wireless testbed (described later in section 4). In particular, we construct the scenario in Fig. 1, where an AP maintains ongoing sessions with 5 clients and transmits saturated unicast traffic to all of these clients. We place a jammer 7 ft.

away from one client ($C$). The jammer emits energy continuously at 0 dBm (1 mW), such that it causes interference to client $C$ only. Fig. 1 depicts our throughput measurements, with and without the jammer. We observe that in the absence of jamming each client receives 4.1 Mbits/sec, on average. When the jammer is enabled, however, the long-term throughput of *all* clients drops to 90 Kbits/sec.

**FIJI: An anti-jamming system to mitigate the implicit-jamming attack:** In order to alleviate the effects of this intelligent attack, we design and implement FIJI, a distributed software system that is executed locally at the APs. With FIJI, the AP is able to quickly detect an implicit jamming attack and identify the clients that are under the direct influence of the jammer(s). Furthermore, via a minimal set of online calibrating measurements that characterize the impact of the attack, the AP shapes the downlink traffic such that: *(a)* the jammed clients receive the maximum possible throughput given the circumstances, and *(b)* the rest of the clients are unaffected, i.e., shielded from the influence of the jammer(s). Some parts of FIJI are implemented on the *Click* software framework [5] and the rest are implemented on the driver/firmware of our wireless cards. Via extensive experiments, we observe that FIJI effectively mitigates the implicit-jamming attack on an 802.11a/g wireless testbed.

*Our work in perspective:* FIJI can be potentially applied in scenarios wherein jammers attack APs directly. However, in this work, we focus on addressing intelligent jammers that exploit the performance anomaly at the client side. Moreover, note that the impact of implicit jamming is exacerbated in downlink traffic scenarios; with uplink traffic, jammed clients will simply defer accessing the medium and will thereby allow the other clients to obtain higher levels of access.

The remainder of the paper is structured as follows. In section 2, we provide a brief background on the performance anomaly in 802.11 as well as jamming attacks, and discuss related studies. In section 3, we describe the implicit jamming detection and mitigation with FIJI, our anti-jamming system. We describe the implementation of FIJI and evaluate its effectiveness in section 4. Section 5 provides the scope of our study. We conclude in section 6.

## 2 Background and Previous Work

In this section, we first describe the so-called performance anomaly with IEEE 802.11 and efforts related to addressing the anomaly. We then discuss jamming attacks in brief as well as prior work related to anti-jamming.

### 2.1 Performance Anomaly in 802.11 WLANs

Heusse et al. [4] were the first to observe that the long term throughput of all the clients associated with an AP in a WLAN is limited by the client with the poorest link. This effect eventually provides the same long-term throughput to all clients. Although [4] considers uplink traffic, this “anomaly” arises with downlink traffic as well [6, 7]. With either uplink or downlink saturated traffic, 802.11 provides equal medium access probability to all links. Let us consider the downlink scenario. An AP→client link with low SINR will coerce the rate adaptation mechanism at the AP to use a low transmission rate for this client. Thus, when attempting to serve this client, the AP will spend large amounts of time. Given that the AP will access the channel with equal probability for low-SINR clients and high-SINR clients (higher bit rate, shorter transmission durations), the latter will be served for smaller proportions of time.

Let us assume that AP $\alpha$ is sending saturated unicast traffic to each of its $\kappa$ clients. The *theoretical* instantaneous transmission rate from AP $\alpha$ towards client $c_i$, where $i \in \{1, ..., \kappa\}$, is a step function of the SINR for this client [8]. In this work, we consider $f_{c_i}$ to be the instantaneous *deliverable* rate towards client $c_i$, which in practice may not always be equal to the transmission rate (especially at high rates). Each client $c_i$ of AP $\alpha$ will receive the ***same*** throughput $T_i$ in the long term; this throughput is given by:

$$T_i = M_\alpha \cdot \frac{B}{\sum_{i=1}^{\kappa} \frac{B}{f_{c_i}}} = M_\alpha \cdot \frac{1}{\sum_{i=1}^{\kappa} \frac{1}{f_{c_i}}} \quad . \tag{1}$$

In the above equation, $M_\alpha$ is the fraction of the time that AP $\alpha$ is able to access the medium, given the contention with its co-channel neighbor devices. We assume that AP $\alpha$ transmits data packets of the same length $B$ to all clients. From the above equation it is evident that if a client $c_i$ receives low throughput, *all* clients will also receive equally low throughput under saturated conditions. Note that this phenomenon has been taken into account during the design of previous performance improvement algorithms for WLANs; examples can be found in [3], [6], [7], [8]. All these studies take the anomaly as a given and try to improve the network performance through other intelligent strategies, such as AP load balancing and power control. In other words, such studies are inherently based on the fact that the 802.11 MAC protocol provides long-term fairness. Clearly, when this property of 802.11 is exploited by a malicious attacker, the performance of the schemes that are based on this property is also compromised. Hence, the existence of a mechanism that detects and mitigates such jammers becomes very vital.

**Studies on mitigating the performance anomaly in 802.11:** There have been numerous efforts on addressing the anomaly in 802.11. Most of them either require significant modifications on the 802.11 protocol functionality or they are very difficult to implement in practice.

***Packet aggregation:*** Razafindralambo et al., [9] propose *PAS*, a technique that involves packet aggregation with dynamic time intervals. With PAS, nodes transmit consecutive packets back-to-back, separated by a SIFS period [10]. As a result, high-rate clients are able to transmit/receive many packets during an allocated time interval. However, packet aggregation requires modifications on the 802.11 protocol, in order to allow back-to-back data frame transmissions.

***Contention window manipulation:*** Kim et al., [11] show that the anomaly can be addressed by tuning the 802.11 contention window size. They compute the minimum value of the window for the elimination of the anomaly. This technique, however, requires modification to the algorithm that selects the value of the contention window in 802.11. In contrast, our proposed scheme (described in the following section) does not require any changes to the 802.11 protocol semantics.

***Data traffic manipulation:*** Bellavista *et al.*, in [12] propose *MUM*, an application-level middleware for facilitating multimedia streaming services. MUM tries to detect the anomaly by monitoring the RSSI of received packets and estimating the goodness of links. It employs the Linux `tc/iptables` to implement a hierarchical token buffer scheduler [13] that "differentiates" data transmissions towards low-rate nodes. The RSSI, however, cannot accurately capture the levels of contention and interference [14]. In addition, [12] uses a limited set of 4 static rate classes for traffic differentiation; *this setting is not adequate in jamming scenarios, as we show in section 4*. Along the same lines, Dunn *et al.*, [15] propose a heuristic for allocating a packet size to every client, which is proportional to the transmission rate. *We show in section 4 that the use of this heuristic during an implicit-jamming attack leads to some undesirable effects that in turn lead to poorer throughput than what is possible with FIJI*. Similar approaches are followed in [16, 17] and [18]. Finally, Yang et al. [19] analytically model a WLAN with stations that support multiple transmission rates in order to demonstrate the performance anomaly. In contrast with these studies, our anti-jamming solution addresses the fact that the maximum transmission rate achieved by a single client can bound the total AP throughput. From the above discussion, as well as our measurements in section 4, it becomes evident that prior efforts on overcoming the performance anomaly problem in 802.11 cannot efficiently mitigate implicit jammers. We approach the 802.11 anomaly from the security point of view; in particular we examine a case where a malicious adversary can remotely exploit this feature as a vulnerability to cause complete starvation to the associated clients. *FIJI is effective against the implicit jamming attack, provides the best trade-offs between throughput and fairness and does not require any modifications on the 802.11 protocol.*

### 2.2 Jamming in Wireless Networks

Jammers are classified into two main categories based on their behaviors.

- ***Constant jammers:*** They emit electromagnetic energy all the time. This jamming technique is not usually adopted, since it depletes the battery of mobile jammers rather quickly. This category includes *deceptive* jammers [20], which transmit seemingly legitimate back-to-back data packets. With this, deceptive jammers can mislead other nodes and monitoring systems into believing that legitimate traffic is being sent over the medium.
- ***Intermittent jammers:*** They conserve battery life by emitting energy intermittently. As examples: (i) *Random* jammers alternate between random jamming and sleeping periods. (ii) *Reactive* jammers emit energy right after the detection of traffic on the medium, and remain inactive as long as the medium is idle. The implementation of reactive jammers is difficult; the detection and alleviation of such attacks is very challenging.

**Previously proposed anti-jamming techniques:** Prior work has focused on the impact of jamming on the performance of isolated wireless links. To the best of our knowledge, FIJI is the first system to examine the effects of implicit jamming on the *overall* performance of WLANs. Some previous studies employ frequency hopping techniques to avoid jammers [21, 22, 23]. We do not adopt such techniques in FIJI, since frequency hopping cannot overcome wide-band jammers [2], which are capable of jamming a plurality of the available bands simultaneously. Moreover, frequency hopping has limited effectiveness when multiple collocated jammers operate on different frequencies. FIJI, however, can be complementary to frequency hopping.

Gummadi *et al.* [21] show that even ultra-low power jammers can corrupt the reception of packets; towards coping with these jammers they propose a rapid frequency hopping strategy. Navda *et al.* [22] implement a proactive frequency hopping protocol with pseudo-random channel switching. They compute the optimal frequency hopping parameters, assuming that the jammer is aware of the frequency hopping procedure that is followed. Xu *et al.* [23] propose two anti jamming techniques: reactive channel surfing and spatial retreats. However, they do not consider 802.11 networks. In [20], efficient mechanisms for jammer detection at the PHY layer are developed. However, the authors do not propose any anti-jamming mechanisms. The work in [24] suggests that the proper adjustment of transmission power and error correction codes could alleviate jamming effects. However, it neither proposes an anti-jamming protocol nor performs evaluations of these strategies. Along the same lines, Lin and Noubir [25] present an analytical evaluation of the use of cryptographic interleavers with various coding schemes to improve the robustness of wireless LANs. In subsequent work, Noubir and Lin [26] investigate the power efficiency of a jammer. They show that in the absence of error-correction codes a jammer can conserve battery power by simply destroying only a portion of a legitimate packet. Finally, Noubir [27] proposes a combination of directional antennae and node-mobility in order to alleviate jammers.

*None of these efforts consider the implicit jamming attack; FIJI is the first system to address this attack.*

## 3 FIJI to Combat the Implicit Jamming Attack

In this section, we describe the design of our anti-jamming software system, FIJI. The goal of FIJI is twofold:

1. To detect the attack and restore the throughput on clients that are not explicitly jammed (we call these clients **"healthy"**).
2. To maintain connectivity and provide the highest possible throughput to clients that are explicitly jammed (we call these clients **"jammed"**).

FIJI involves the co-design of two individual modules, executed at the AP: a *detection* module and a *traffic shaping* module. We have implemented the two modules in the kernel space (we provide implementation details in section 4).

**Attack model:** In this work, we focus on low-power deceptive jammers. In particular, we assume that the jamming device has the following properties:

- It is placed next to legitimate clients. With this, the jammer is able to distort packets destined to the jammed client(s). In addition, the jammer is constantly transmitting packets back-to-back, thereby prohibiting the jammed clients from accessing the medium.
- It operates at very low power. As discussed earlier, the jammer simply needs to explicitly affect one of the clients of the AP. By transmitting at low power the jammer can conserve energy and make the detection of the attack a challenging task.

– It is able to operate on a wide band (covering all the available channels); this makes frequency hopping techniques inappropriate.

We describe the operation of the detection and the traffic shaping modules in what follows.

### 3.1 Detecting the implicit-jamming attack

The purpose of this module is to make the AP capable of detecting the jammed clients. *Previous jamming detection schemes assume that the jammed node is always the one that performs the detection. However with the implicit-jamming attack, the AP needs to detect the jammed client(s) in order to prevent the throughput starvation of the healthy clients.* As an example, in [20] the jammed node performs a consistency check between the instantaneous PDR (Packet Delivery Ratio), and the RSSI (Received Signal Strength Indicator) that it measures on its antenna. If the PDR is extremely low (i.e., almost zero), while the RSSI is much higher than the CCA threshold[2], the node is considered to be jammed. With the implicit jamming attack, however, the AP does not know the RSSI value that is observed by each of its clients. Thus, the approach in [20] does not allow the AP to detect the implicit jamming attack.

**Measuring the transmission delay per client:** FIJI relies on measuring the data unit transmission delay $d_{c_i} = B/f_{c_i}$ of every client $c_i$ at the AP. More specifically, the denominator of Eq. (1) is the aggregate transmission delay $D_\alpha$ incurred by AP $\alpha$ in order to serve all of its associated clients once; it is the sum of the individual $d_{c_i}$ values, $i \in \{1, ..., \kappa\}$, of the $\kappa$ clients that are associated with AP $\alpha$ [3]. In other words, if we assume saturated downlink traffic, $D_\alpha$ corresponds to the average time that AP $\alpha$ needs in order to send one data unit to every client. The value of $D_\alpha$ is the same for all clients, and the transmission delay $d_{c_i}$ of client $c_i$ contributes to the value of $D_\alpha$. Hence, a sudden, very large increment in $D_\alpha$ indicates that one or more of the $d_{c_i}$ values has suddenly increased; *this would imply that one or more clients are under attack*. Towards calculating $D_\alpha$, AP $\alpha$ needs to measure the $d_{c_i}$ value for every client $c_i$ (this includes possible retransmission delays and the rate-scaling overhead[3]). Measuring $d_{c_i}$ will directly reveal the jammed clients: the value of $d_{c_i^J}$ for a jammed client $c_i^J$ is likely to be much higher than the delays of the other clients. We adopt this detection strategy in FIJI.

### 3.2 Shaping the traffic at the AP to alleviate jammers

A trivial solution to the problem of mitigating the attack would be for the AP to simply stop serving the jammed clients. However, this would be unfair, since in many cases the jammed clients might still be able to receive data, albeit at lower rates. We opt to provide a **fair** bandwidth allocation solution; our twofold objective is to simultaneously achieve the following:

[2] The CCA (Clear Channel Assessment) threshold specifies the RSSI value below which, receptions are ignored with regards to carrier sensing [8].

[3] The rate scaling overhead accounts for the higher delays incurred due to transient lower rates that the rate adaptation algorithm invokes.

- **Objective 1:** For each of the healthy clients we seek to provide the same throughput that they would have enjoyed in the absence of the jammer, i.e., prior to the attack.
- **Objective 2:** A jammed client typically cannot receive much throughput as long as the jammer is active. Hence we want to provide to every jammed client the maximum possible throughput that it can receive, given that objective 1 is satisfied.

We refer to the state where these objectives are met as the ***optimal state***.

We propose a real-time, cross-layer software system to mitigate the effects of the implicit-jamming attack. The system is implemented partly in the Click module [5] and partly in the wireless driver/firmware. Click receives information from the MAC Layer with regards to the properties of the jammed clients. The AP→client traffic is then appropriately shaped and forwarded down to the MAC layer at the AP.

**i) DPT: Controlling the data packet size:** With this strategy, the AP fragments the packets destined to jammed clients; each such smaller fragment is now an independent packet. We call this approach DPT for *Data Packet Tuning*. With DPT, the rate at which these smaller packets are sent to the MAC layer is equal to the rate at which normal packets were forwarded to the MAC layer, prior to jamming. DPT is expected to have the following effects: *(a)* The transmission of small data packets is more robust to interference due to jamming; hence these small packets are more likely to be correctly deciphered by the jammed clients. *(b)* The rate at which the AP accesses the medium for the jammed clients remains unchanged; however, the channel occupancy time that is spent for them is reduced, due to transmitting smaller packets to jammed clients. Hence, the AP will allocate a larger fraction of time for healthy clients.

***Deriving the optimal data packet sizes:*** Our target is to determine the right packet size such that the optimal state is reached. The problem of achieving this state is formulated as follows.

Let us suppose that AP $\alpha$ has $\kappa$ associated clients, and that $n$ clients are being jammed, with $n \leq \kappa$. Our objective is to *minimize the aggregate transmission delay $D_\alpha^J$ of all the jammed clients $c_i^J$, $i \in \{1, .., n\}$ of AP $\alpha$*. In other words, we seek to minimize

$$D_\alpha^J = \sum_{i=1}^{n} d_{c_i^J} = \sum_{i=1}^{n} \frac{J_i}{f_{c_i^J}} ,$$

where $J_i$ is the data unit length for jammed client $c_i^J$, while $f_{c_i^J}$ is the deliverable rate at $c_i^J$.

*Constraint:* The $d_{c_i^J}$ value of each jammed client $c_i^J$ must be at least equal (and as close as possible) to its data unit transmission delay $d_{c_i}$ in benign conditions:

$$X1 : \;\; d_{c_i}^J \geq d_{c_i} \Rightarrow \frac{J_i}{f_{c_i^J}} \geq \frac{B}{f_{c_i}}, \; \forall i \in n ,$$

where $B$ is the default data unit length that the AP is using for all clients, and $f_{c_i}$ is the deliverable rate to $c_i^J$ in benign conditions. As explained earlier, the value of $D_\alpha$ is the same for all clients that are associated with AP $\alpha$. If we sum constraint $X1$ over all jammed clients, the left hand side of the inequality is our objective function. With this we make sure that the healthy $\kappa - n$ clients will indeed experience an aggregate transmission delay very close to $D_\alpha = \sum_{i=1}^{\kappa}(B/f_{c_i})$; note that this is the aggregate transmission delay that was experienced by these clients prior to the jamming attack. Hence, by choosing the packet size $J_i$ that results in a transmission delay that is as close to $d_{c_i}$ as possible, we ensure that the throughput of the healthy clients remains unaffected (we elaborate on this later with an example).

Based on the above constraint, our optimization problem can be formulated as follows:

$$minimize : D_\alpha^J = \sum_{i=1}^{n} d_{c_i^J} = \sum_{i=1}^{n} \frac{J_i}{f_{c_i^J}} \tag{2}$$

$$subject\; to: \quad 1 \leq J_i \leq B,\; \forall i \in \{1, 2, ..., n\}, \tag{3}$$

$$and\; X1. \tag{4}$$

The solution to the above problem provides the values of $J_i$ that minimize (2). Although the problem is an integer programming problem, it is easy to see that its special form ensures that it always has a solution, which can be found in polynomial time w.r.t. the number of variables.

***How does DPT operate?*** Let us consider a case study with AP $\alpha$, $\kappa$ = 3, $n$ = 1 and default packet size $B$. The transmission delays for the healthy clients $c_1$ and $c_2$ are $d_1$ and $d_2$, respectively; for the jammed client $c_3$, it is $d_3$. The long-term throughput of every client in benign conditions will be: $T_b = \frac{B}{d_1+d_2+d_3}$. If $c_3$ is now being jammed, its transmission delay will be $d_3^J > d_3$ and the new throughput will be: $T_J = \frac{B}{d_1+d_2+d_3^J}$. By applying DPT, the packet size towards $c_3$ will be $J_3^{dpt}$ and its new transmission delay will be $d_3^{dpt}$. Since the rest of the clients are to maintain their old transmission delays (they are not explicitly jammed), the throughput with DPT will be: $T_{dpt} = \frac{B}{d_1+d_2+d_3^{dpt}}$. Our minimization problem ensures that $d_3^{dpt} \approx d_3$. Thus, for clients $c_1$ and $c_2$: $T_{dpt_1} = T_{dpt_2} \approx T_b$. In other words, DPT restores the throughput at the healthy clients.

Next, we show that the jammed client cannot receive a higher throughput if we further decrease the packet size[4] to a value $J_3^l < J_3^{dpt}$. With packet size $J_3^{dpt}$ the throughput at $c_3$ will be: $T_{dpt_3} = \frac{J_3^{dpt}}{d_1+d_2+d_3^{dpt}}$. Let us assume that with packet size $J_3^l < J_3^{dpt}$ the transmission delay of $c_3$ is $d_3^l$. The throughput at $c_3$ will then be $T_{l_3} = \frac{J_3^l}{d_1+d_2+d_3^l}$. The required condition $T_{l_3} < T_{dpt_3}$ can be simplified as:

$$T_{l_3} < T_{dpt_3} \Leftrightarrow d_3^l > \frac{J_3^l}{J_3^{dpt}} \cdot (d_1 + d_2 + d_3^{dpt}) - d_1 - d_2.$$

Since the packet delivery rate $f_{c_3}$ is the same, we have:

$$\frac{J_3^l}{J_3^{dpt}} = \frac{d_3^l}{d_3^{dpt}} \Leftrightarrow d_3^l = d_3^{dpt} \cdot \frac{J_3^l}{J_3^{dpt}}$$

[4] For larger packet sizes, objective 1 cannot be satisfied; hence we do not need to consider such a case.

$$\text{Thus: } \frac{J_3^l}{J_3^{dpt}} \cdot d_3^{dpt} > \frac{J_3^l}{J_3^{dpt}} \cdot (d_1 + d_2 + d_3^{dpt}) - d_1 - d_2 \Leftrightarrow$$

$$0 > (\frac{J_3^l}{J_3^{dpt}} - 1)(d_1 + d_2).$$

The last inequality is always true; hence, $T_{l_3} < T_{dpt_3}$.

Similar steps can be followed in order to show that DPT operates in the same manner in scenarios with multiple jammed clients. We adopt DPT in FIJI.

**ii) DRT: An alternate approach.** An alternative strategy would be to *explicitly tune the rate at which the packets are delivered* at the MAC layer (the packet size is now kept unchanged), destined to jammed clients. Fewer packets would arrive at the MAC layer for transmission towards the jammed clients, thereby allowing the AP to send traffic to healthy clients more frequently. Let us call this approach DRT for *Data Rate Tuning*. DRT operates as follows. Based on the measured $d_{c_i}$ for each client $c_i$, the deliverable rate to every jammed client would be:

$$f_{c_i^J} = B/d_{c_i^J}. \quad (5)$$

DRT would bound the packet generation rate such that the data rate to the jammed client $c_i^J$ is at most $f_{c_i^J}$. As a result, the rest of the (healthy) clients would share the remaining bandwidth. Thus, they would enjoy a share that is in fact higher than what they had prior to the attack. However, the packets destined to the jammed clients could be potentially lost due to channel or interference effects. Hence with DRT, the jammed clients will eventually receive *lower long-term throughput* than the specified (by DRT) rate of $f_{c_i^J}$. Clearly, while both DPT and DRT shape the traffic in order to overcome the implicit jamming effects, they essentially differ in the way they allocate the bandwidth. With DPT the healthy clients receive the *same throughput as before the attack*, while the jammed clients achieve the *maximum possible* throughput under the circumstances. On the other hand, with DRT the healthy clients have a higher share of the bandwidth than in benign settings and receive *more throughput than before the attack*; the APs will spend more time serving the healthy clients, since most of the traffic is now destined to them. However, since the jammed clients do not reach their capacity, they are treated rather “unfairly”. We evaluate this fairness versus throughput trade-off in section 4.3.

## 4 Implementation and Evaluation

In this section, we first describe our implementation of FIJI. Next we apply FIJI on a WLAN testbed and evaluate its efficacy in overcoming the implicit jamming attack.

### 4.1 The implementation of FIJI

FIJI is implemented entirely at the AP; no client software modifications are needed. In addition, FIJI does not require any special functionalities at the APs or at the clients; the only requirement is for the AP to be able to measure the $d_{c_i}$ value for each affiliated client. Hence, FIJI can be applied on commercial APs through a driver/firmware update. In order to implement the two modules of FIJI we perform modifications on the driver and firmware of the AP, and we develop specific traffic shaping functionalities on the Click framework [5].

**Implementing the implicit-jamming detection module:** As explained in section 3.1, the AP needs to measure $d_{c_i}$ for every client $c_i$. This will reveal, with high probability, the set of jammed clients. However, the value of $d_{c_i}$ cannot be directly obtained from the driver of the wireless card; modifications in the firmware are required in order to compute this value. We use a prototype version of the *Intel ipw2200* AP driver/firmware; for every client we measure the time duration between the placement of the packet at the head of the MAC queue until an 802.11 ACK frame is received for this packet. The value is then passed up to the driver. The AP maintains a table in the driver space with the $d_{c_i}$ value for every client $c_i$. It also computes $D_\alpha^J$ (when jammers are active) and $D_\alpha$ (when jammers are inactive), by summing up the corresponding client delays. Temporary variations of the $d_{c_i}$ values are handled by FIJI by using weighted moving average filtering; the previously maintained average is assigned a weight of 0.9 while the new sample has an associated weight of 0.1 (similar values are used in [3, 6]). Using these values, the AP constructs a table with the appropriate data packet sizes for the jammed clients. If the weighted $d_{c_i(new)}/d_{c_i(old)}$ value (for one or more clients) exceeds a pre-specified threshold $\delta$, the AP computes the new packet sizes, updates the table and subsequently feeds it into the traffic shaping module, described below.

**Implementation of the traffic shaping module:** We implement the traffic shaper in Click. The module receives the table from the driver with suggested parameter settings for every client and shapes the traffic accordingly. We implement both DPT and DRT for comparison purposes. For DPT we have also developed an application-level script, which reads the table with the suggested packet sizes and inputs these values to the rude/crude measurement tool [28]. For DRT one may use two different Click elements, namely either the `BandwidthShaper(bandwidth)` or the `LinkUnqueue(latency, bandwidth)` element; we utilize the latter. Finally, we configure the AP to periodically flush the stored transmission delay values for every client and perform fresh delay measurements, using the default packet size. With this, we address scenarios of mobile jammers, which may move to the proximity of different clients, jammers with variable transmission power as well as jammers that stop operating.

### 4.2 Experimental set-up and methodology

**Testbed description:** Our testbed consists of 28 Soekris net4826 nodes [29], which mount a Debian Linux distribution with kernel v2.6 over NFS. The testbed is deployed in the 3rd floor of our campus building; the node layout is depicted in Fig. 2. Each node is equipped with an *Intel-2915* mini PCI WiFi card, connected to two 5-dBi gain external omnidirectional antennae. We use both the $main$ and $aux$ antenna connectors of the card for diversity. As mentioned earlier, we use a proprietary version of the *ipw2200* AP driver/firmware of the *Intel-2915* card. With this version we are able to (a) measure the $D_\alpha$ and $D_\alpha^J$ values at the AP, and (b) experiment with both 802.11a and 802.11g.

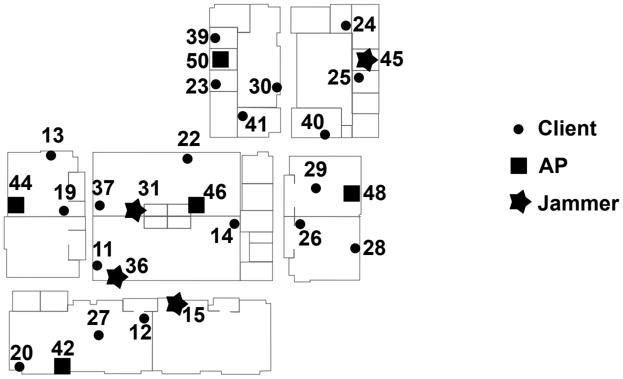

**Fig. 2.** The deployment of our indoor 802.11a/g WLAN testbed in the 3rd floor of a campus building.

**Constant jammer implementation:** We have implemented our own deceptive jammer (instead of purchasing a commercial one [2]) since this gives us the freedom of tuning various jamming parameters. Our implementation of a constant jammer is based on a specific card configuration and a user space utility that sends broadcast packets as fast as possible. Our jammers are also equipped with the Intel-2915 cards; our ipw2200 prototype firmware for these cards allows the tuning of the CCA threshold parameter. By setting the CCA threshold to 0 dBm, we force the WiFi card to ignore all 802.11 signals during carrier sensing (packets arrive at the jammer's circuitry with powers much less than 0 dBm, even if the distances between the jammer and the legitimate transceivers are very small). The jammer transmits broadcast UDP traffic. This ensures that its packets are transmitted back-to-back and that the jammer does not wait for any ACK messages (the back-off functionality is disabled in 802.11 for broadcast traffic). We have developed an application-layer utility that employs *raw sockets*, allowing the construction of UDP packets and the forwarding of each packet directly down to the hardware.

**Experimental methodology:** For each experiment we first enable traffic from the AP to its clients and subsequently we activate the jammer(s). The duration of each experiment is 10 minutes; during each minute, the jammer is inactive for the first $k$ sec, where $k \in [5, 20]$, and active for the other $60 - k$ sec. We use a subset of 4 nodes as the jamming devices (nodes 15, 31, 36 and 45 in Fig. 2). We collect throughput and transmission delay ($d_{c_i}$) measurements once every 500 msec, for each client. We experiment with many different topological settings, with different numbers of APs and clients. By default all legitimate nodes set their transmission powers to the maximum value of 20 dBm and their CCA thresholds to -80 dBm. We examine both 802.11a and 802.11g links (unless otherwise stated, we observe the same behavior for 802.11a and 802.11g). The experiments are performed late at night in order to avoid interference from co-located WLANs, as well as not to cause interference to them. We use saturated UDP traffic with a default data packet size $B$ = 1500 bytes. We also experiment with TCP traffic[5]. We use the *iperf* measurement tool to generate data traffic among legitimate nodes. We also use the *rude* tool to test DPT.

### 4.3 Does FIJI Deliver?

Next, we apply our anti-jamming framework on the testbed and evaluate its efficiency in alleviating the effects of implicit-jamming on the WLAN performance.

**i) The efficacy of the detection module.** We seek to observe two properties of this module:

1. *Efficiency of Detection*: How quickly can FIJI detect the presence of implicit jammers?
2. *Accuracy of Detection*: How accurately can FIJI determine if there is an ongoing jamming attack?

We conduct experiments with 5 APs and different numbers of clients with various link qualities. We configure the jammers to transmit at 0 dBm (1 mW) with CCA = 0 dBm, such that they affect one or more clients without affecting the APs.

***a) On the speed of detection:*** Our measurements indicate that the transmission delay $d_{c_i^J}$ of a client increases sharply upon experiencing the implicit jamming attack. This increase is seen in less than 700 msecs; this time includes the transient periods before the weighted average $d_{c_i^J}$ converges to a stable value. Fig. 3 depicts a delay snapshot with one AP and four clients with moderate-quality links. We observe that the $d_{c_1^J}$ value increases significantly (by 26 times in this experiment). Other experiments provided similar results. In summary, these results show that FIJI can quickly detect implicit jamming attacks.

***b) On the accuracy of detection:*** We seek to evaluate FIJI in terms of its ability to detect an implicit jamming attack in the presence of interference. Note that the $d_{c_i}$ value for a client $c_i$ is affected by the levels of interference on the AP $\rightarrow c_i$ link. The higher the level of interference, the higher the $d_{c_i}$ value. In order to evaluate this ability of FIJI, we perform experiments with multiple overlapping cells (each with its own AP), so that some clients suffer interference from one or more APs; in this setting, we activate our low-power jammers.

***Detecting jamming on good quality links:*** We first consider links that have a high SINR. Fig. 4 depicts sample experimental results. In the snapshot of Fig. 4, a jammer is placed such that it affects 2 out of the 4 clients of an AP. We observe that FIJI is able to perform a successful detection. In general, our empirical observations suggest that when threshold $\delta \geq 9$, FIJI can effectively detect the attack (Fig. 4). In the experiment described above, the value of $\delta$ was 9.

***FIJI and poor quality links:*** With poor quality links (SINR is low), FIJI cannot easily decide if a client is under attack or not. This effect is captured in Fig. 5, where the jammer affects a very poor link. In particular, the link 46→25 is considered with

[5] The anomaly exists with TCP traffic as well [4]. Even though we do not present our TCP measurements, we observe that FIJI is similarly efficient with TCP traffic; we discuss this briefly in section 5.

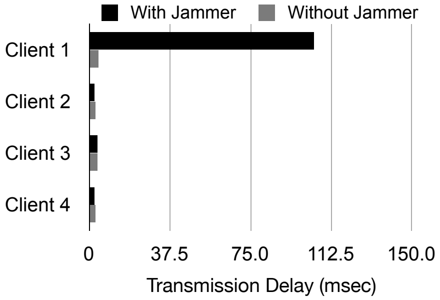

**Fig. 3.** FIJI detects jammed clients by measuring their data unit transmission delays.

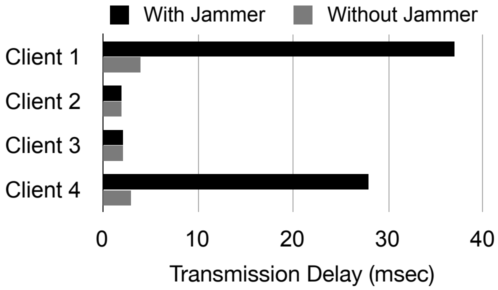

**Fig. 4.** The jammer detection functionality of FIJI is accurate in most cases.

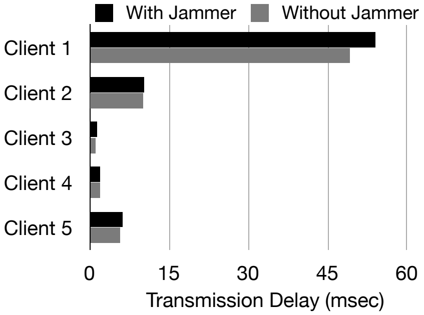

**Fig. 5.** The jammer detection with FIJI is less accurate in scenarios with very poor links.

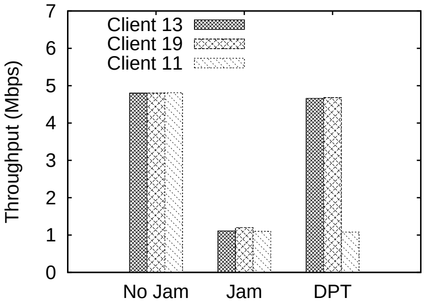

**Fig. 6.** DPT restores the performance of healthy clients to that in benign settings.

the node 45 acting as a jammer (Fig. 2). The link achieves 190 Kbits/sec in the absence of jamming and 164 Kbits/sec under jamming. Since the jammer does not significantly increase the delay experienced on such poor links, FIJI cannot decipher whether the increased $d_{node-25}$ value is due to jamming or legitimate interference. However, in such conditions, the overall change in the network performance due to the jammer is unlikely to be significant; the presence of the poor link already hurts the network performance. Furthermore note that a jammer is unlikely to attack such poor quality links if it aims to harm the network to the extent possible.

In some extreme cases, a poor quality link (exposed perhaps to other interfering APs that are hidden from its own AP) might cause a client to experience large delays. In such scenarios with healthy but poor-quality links, FIJI may incorrectly classify such links as being *jammed*. Classifying such cases as attacks, though, is perhaps appealing in terms of improving performance for the rest of the network.

***FIJI and high power jammers:*** An implicit-jamming attacker is likely to place its jammer(s) very close to one or more clients so as to:

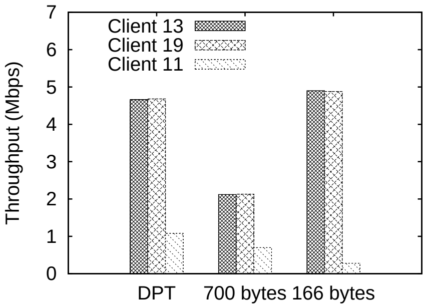

**Fig. 7.** DPT always manages to provide a fair allocation of throughput among clients .

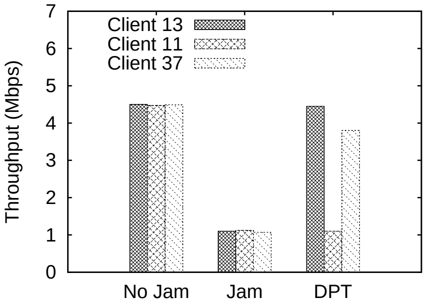

**Fig. 8.** DPT can easily handle scenarios with multiple clients that are simultaneously jammed.

– degrade the client's observed SINR value to the extent possible, and
– use a very low transmission power, in order to conserve energy and avoid detection.

As our experiments indicate, under these conditions, FIJI can identify the jammed clients in real time since all measured $d_{c_i^J}$ values are usually extremely high for those clients. In contrast, a jammer could use high transmission power (although this could increase the chance of its detection and result in high energy consumption). Such a high power jammer is likely to affect multiple clients and even the AP itself, directly. The delays of all these clients may go up and in this case, given its design principles, FIJI may not be able to detect the jammer. However, there are other jammer detection techniques that can be used in conjunction with FIJI to detect such jammers [20].

**ii) The traffic shaping module in action.** Next we evaluate the efficacy of DPT and compare it against DRT.

***DPT is the most fair solution:*** In a nutshell we observe that as long as the jammer is successfully detected, DPT restores the throughput at the healthy clients. A sample case is depicted in Fig. 6. Here, AP 44 transmits unicast traffic to clients 11, 13 and 19; node 36 is jamming client 11. In the absence of jamming each client receives 4.8 Mbits/sec on average. When the jammer is active, without enabling DPT, all clients receive 1.1 Mbits/sec on average. The solution to the problem formulated in (2) suggests that $J_{11}$ should be set to 345 bytes. When DPT is enabled and this packet size is chosen for the jammed client, we observe that the throughput of the healthy clients 13 and 19 is restored to 4.66 Mbits/sec, while the jammed client 11 achieves about 1.1 Mbits/sec. Note that the healthy clients do not achieve their jamming-free throughput of 4.8 Mbits/sec. This is because in our solution the equality in the constraint $X1$ is achieved for a non-integral value of $J_{11}$; we round the value of $J_{11}$ up to the nearest integer. With this, the transmission delay for the jammed client is a bit higher as compared to the delay under benign conditions and this slightly degrades the throughput at the healthy clients.

In order to validate that DPT provides the most fair bandwidth allocation, we experiment with many different $J_{11}$ values. Fig. 7 depicts the results that correspond to the settings with two $J_{11}$ values: 166 and 700 bytes. We observe that:

– With packet sizes smaller than $J_{11}^{dpt}$ (case with 166 bytes), the jammed client does not reach its capacity (receives 360 Kbits/sec) and the AP spends more time serving the healthy clients (as discussed in section 3): each healthy client now receives 5.1 Mbits/sec. Note that the value $J_{11}$ = 166 bytes is computed using the approach proposed in [15] for the considered scenario and *it clearly does not provide the desirable fairness in terms of throughput*.

– When the packet size is higher than $J_{11}^{dpt}$ (case with 700 bytes), the throughput at the jammed client is lower than 1.1 Mbits/sec; the healthy clients also underperform. This is again conformant with our analytical assessments in section 3 with regards to the maximum achievable throughput.

**Multiple jammed clients:** We have so far considered scenarios wherein a single client was jammed. Next, we examine scenarios with multiple jammed clients per AP. Our experiments reveal that DPT is also able to effectively mitigate the implicit jamming attack in such scenarios. Fig. 8 presents a sample case with AP 46 and clients 11, 37 and 14; the jammer-node 36 explicitly affects both clients 11 and 37. Under benign conditions all clients receive approximately 4.5 Mbits/sec on average. As soon as the jammer is activated, without enabling DPT, all clients receive about 1.1 Mbits/sec. DPT sets the value of $J_{11}$ to be 367 bytes and $J_{37}$ to be 1266 bytes. With this, *DPT is able to restore the throughput at the healthy clients*.

**DPT vs. DRT:** Using the same methodology, we examine the effectiveness of the DRT solution. Our measurements demonstrate that DRT provides much higher throughput to healthy clients. On the other hand, DRT results in an additional unfair degradation at the jammed client. Fig. 9 represents the behaviors in an example scenario, with the same topological configuration as before (AP 44, clients 11, 13 and 19, jammer 36); the figure depicts the throughput prior to the attack (benign settings), with the jammer without DRT, and after the application of DRT. We observe that DRT overcomes the implicit impacts of the attack. Upon enabling DRT, clients 13 and 19 are no longer affected by the jammer and they receive 5.12 Mbits/sec each. Although DRT sets the maximum allowable data rate towards client 11 to be 1.1 Mbits/sec, the observed throughput at this client is significantly lower i.e., 680 Kbits/sec on average. This behavior of DRT conforms with our discussion in section 3.2; we observe similar trends in all our measurements with one or more jammed clients. *To summarize, with DRT the healthy clients receive more throughput than before the attack; however the jammed clients are penalized further*.

The choice between DPT and DRT depends on the performance objectives; one has to decide between fairness (with DPT) and bandwidth utilization (with DRT). DPT is fair: the healthy clients receive the same throughput as before the attack, while the jammed clients achieve the maximum possible throughput under the circumstances. On the other hand, DRT increases the throughput at the healthy clients and potentially, the total network throughput. However, the jammed clients cannot receive the maximum throughput that they can achieve in the presence of the jammer.

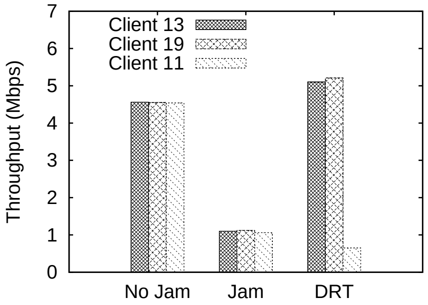

**Fig. 9.** With DRT healthy clients receive more throughput than before the attack.

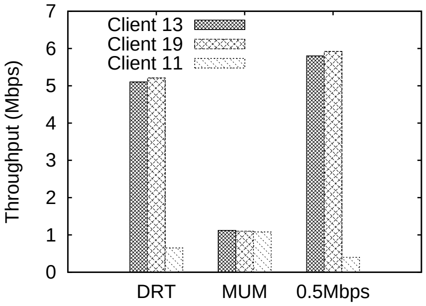

**Fig. 10.** DRT satisfies our objectives better than other data rate allocation approaches.

Note that DRT also relies on the online measurement and use of $d_{c_i}$. With this, DRT seeks to eliminate the effects of implicit jamming at healthy clients, while at the same time not degrade the throughput at jammed clients. Fig. 10 depicts a case with 802.11a where DRT sets the data rate at 1.1 Mbits/sec, while MUM [12] (recall our discussion in section 2) sets 6 Mbits/sec. We observe that by using data rates higher than the one chosen by DRT, the healthy clients are still affected by the attack, since in this case the downlink traffic for the jammed client is still saturated. Moreover, if we use lower data rates than the one chosen by DRT, the healthy clients get more service time, however the jammed clients receive much lower throughput than with DRT.

## 5 The Scope of Our Study

**FIJI and previous studies on traffic shaping:** Our work is the first to analytically derive the *optimal* settings for traffic shaping at the AP to mitigate the implicit-jamming attack. Traffic shapers have also been previously proposed in [12, 16, 17, 15]. Clearly, FIJI could also be considered as another traffic shaper, simply to overcome the performance degradation due to the 802.11 anomaly. Unlike FIJI however, previous traffic shaping schemes cannot overcome the effects of an implicit-jamming attack, as explained in sections 2 and 4. Other schemes that provide fair access to the WLAN resources [30, 3] would also be inadequate in combating an implicit-jamming attack since they are not designed for this purpose.

**FIJI versus power control:** Power control has been suggested as a means of mitigating legitimate interference [7, 31]. Typically with power control, nodes tune their transmission power and CCA settings in order to reduce the amount of interference from/to their neighbors. However, if the jammer is very close to one or more clients, its signal cannot be ignored through CCA adaptation. If a client increases its CCA threshold to a high level (to ignore the jammer's signal), the connectivity to the AP will be lost.

**Addressing random and reactive jammers:** FIJI can mitigate the interference due to any type of jammer, even random or reactive jammers. With prolonged random jamming and sleeping periods (order of seconds), FIJI can perform a rapid detection and then customize the data packet size, as per the observed data unit transmission delay $d_{c_i^J}$. If the sleep and active periods of the random jammer are of the order of milliseconds, FIJI can monitor the *average* $d_{c_i^J}$ value instead. FIJI is expected to alleviate reactive jammers, too, since it only needs to monitor the impact of reactive jamming by measuring $d_{c_i^J}$. We have not experimented with reactive jammers, since implementing such a jammer is a very difficult task.

**FIJI against other attacks:** The two modules of FIJI can arguably be effective against any attempt to exploit the 802.11 performance anomaly in order to degrade the client throughput. As examples, a *compromised* device $x$ could *deliberately* decide to *(a)* associate to a very distant AP $\alpha$, or *(b)* accept traffic at a very low reception rate only (e.g. by discarding a large volume of correctly received packets). In both cases, $x$ would receive a few Kbits/sec. Note here that, legitimate, non-compromised devices would follow such an approach only if they cannot associate with a better APs. However, given that (a) dense deployments of WLANs make the presence of an AP with a good quality link likely [7], and (b) distant poor quality APs are likely to be beyond the administrative domain of the client (the client will not be able to associate with such APs), the possibility of this is small in practice. FIJI can arguably be effective against such attacks. In particular, FIJI considers such clients to be jammed clients and ensures that the other clients remain unaffected.

**FIJI and TCP:** FIJI is implemented above the 802.11 MAC and below the transport layer at the AP. We have done measurements with TCP, which have demonstrated that: *(a)* Without FIJI, the performance anomaly also exists with downlink TCP traffic. The TCP packets that are destined to the jammed clients require a significant amount of time for successful delivery. As a consequence, the healthy clients are affected; they do not achieve the same throughput as before the attack. *(b)* Our experiments also demonstrated that the application of FIJI in TCP traffic scenarios is beneficial. By reducing the rate at which packets are delivered to the MAC for the jammed clients, DPT shapes the TCP traffic in a way that the healthy clients are unfettered. Note that the packet fragmentation with FIJI is executed after any TCP layer fragmentation; hence, FIJI does not intervene with TCP operations.

## 6 Conclusion

In this paper we identify a low-power jamming attack that we call the *implicit jamming attack*. With this attack, a jammer exploits a performance trait of the IEEE 802.11 MAC protocol to cause starvation to not only an explicitly jammed client, but all the clients associated with the same AP as that client. Since the 802.11 MAC provides long term fairness (under saturation conditions) to the associated clients in terms of equal throughput, the attacker can nullify the AP throughput by affecting only one or at most a few clients.

We design, implement and evaluate FIJI, a cross layer software system for mitigating the implicit-jamming attack. FIJI is comprised of two modules, for detecting such an attack and shaping the traffic appropriately in order to alleviate the jamming effects. We evaluate FIJI on an 802.11a/g testbed, and under many different jamming scenarios.

We show that FIJI can quickly detect the attack and effectively restore the throughput at the implicitly affected clients. FIJI also ensures that the jammed clients get as much throughput as they can under the circumstances.

*Acknowledgment:* We thank Intel Research for providing the wireless driver.

## References

1. SESP jammers. http://www.sesp.com/.
2. ISM wideband jammers. http://69.6.206.229/e-commerce-solutions-catalog1.0.4.html.
3. K. Sundaresan and K. Papagiannaki. The Need for Cross-Layer Information in Access Point Selection Algorithms. In *ACM IMC*, 2006.
4. M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *IEEE INFOCOM*, 2003.
5. Click web page. http://read.cs.ucla.edu/click/.
6. B. Kauffmann et al. Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks. In *IEEE INFOCOM*, 2007.
7. I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-Driven Guidelines for 802.11 WLAN Design. In *ACM MOBICOM*, 2007.
8. V. Mhatre, K. Papagiannaki, and F. Baccelli. Interference Mitigation through Power Control in High Density 802.11 WLANs. In *IEEE INFOCOM*, 2007.
9. T. Razafindralambo, I. G. Lassous, L. Iannone, and S. Fdida. Dynamic Packet Aggregation to Solve Performance Anomaly in 802.11 Wireless Networks. In *ACM MSWiM*, October 2006.
10. ANSI/IEEE 802.11-Standard. 1999 edition.
11. H. Kim, S. Yun, I. Kang, and S. Bahk. Resolving 802.11 Performance Anomalies through QoS Differentiation. In *IEEE Comm. Letters, Vol. 9, No. 7*, July 2005.
12. P. Bellavista, A. Corradi, and L. Foschini. The MUM Middleware to Counteract IEEE 802.11 Performance Anomaly in Context-aware Multimedia Provisioning. In *International Journal of Multimedia and Ubiquitous Engineering, Vol. 2, No. 2*, July 2007.
13. Hierarchical Token Bucket. http://luxik.cdi.cz/∼devik/qos/htb/.
14. A. Vlavianos, E. Law, I. Broustis, S. V. Krishnamurthy, and M. Faloutsos. Assessing Link Quality in IEEE 802.11 Wireless Networks: Which is the Right Metric? In *IEEE PIMRC*, 2008.
15. J. Dunn, M. Neufeld, A. Sheth, D. Grunwald, and J. Bennett. A Practical Cross-Layer Mechanism For Fairness in 802.11 Networks. In *IEEE BROADNETS*, 2004.
16. M. Portoles, Z. Zhong, and S. Choi. IEEE 802.11 Downlink Traffic Shaping Scheme for Multi-User Service. In *IEEE PIMRC*, 2003.
17. L. Iannone and S. Fdida. Sdt. 11b: Un Schema a Division de Temps Pour Eviter l'anomalie de la Couche MAC 802.11b. In *CFIP*, April 2005.
18. S. Yoo, J. Choi, J.-H. Hwang, and C. Yoo. Eliminating the Performance Anomaly of 802.11b. In *ICN*, 2005.
19. D. Yang et al. Performance Enhancement of Multi-Rate IEEE 802.11 WLANs with Geographically-Scattered Stations. In *IEEE Trans. Mob. Comp., Vol. 5, Iss. 7*, July 2006.
20. W. Xu, W. Trappe, Y. Zhang, and T. Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *ACM MOBIHOC*, 2005.
21. R. Gummadi et al. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In *ACM SIGCOMM*, 2007.
22. V. Navda et al. Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks. In *IEEE INFOCOM mini-conference*, 2007.

23. W. Hu, T. Wood, W. Trappe, and Y. Zhang. Channel Surfing and Spatial Retreats: Defenses Against Wireless Denial of Service. In *WISE*, 2004.
24. W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming Sensor Networks: Attacks and Defense Strategies. In *IEEE Network*, May/June 2006.
25. G. Lin and G. Noubir. On Link Layer Denial of Service in Data Wireless LANs. In *Wireless Communications and Mobile Computing*, May 2003.
26. G. Noubir and G. Lin. Low-power DoS Attacks in Data Wireless LANs and Countermeasures. In *ACM MOBIHOC (poster)*, 2003.
27. G. Noubir. On Connectivity in Ad Hoc Network under Jamming Using Directional Antennas and Mobility. In *Wired/Wireless Internet Communications, Vol. 2957/2004, pp. 186-200, 2004*.
28. Rude/Crude measurement tool. http://rude.sourceforge.net/.
29. Soekris-net4826. http://www.soekris.com/net4826.htm.
30. A. Jardosh et al. IQU: Practical Queue-Based User Association. In *ACM MOBICOM*, 2006.
31. A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-Management in Chaotic Wireless Deployments. In *ACM MOBICOM*, 2005.