POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Passive Solution for Interference Estimation in WiFi Networks

Original
A Passive Solution for Interference Estimation in WiFi Networks / Rossi C.; Casetti C.; Chiasserini C.F.. - STAMPA. - 8487(2014), pp. 156-168. ((Intervento presentato al convegno 13th International Conference on Ad-Hoc Networks and Wireless tenutosi a Benidorm (Spain) nel June 2014.

Availability:
This version is available at: 11583/2534705 since:

Publisher:
Springer

Published
DOI:10.1007/978-3-319-07425-2_12

Terms of use:
openAccess

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 September 2020

A Passive Solution for Interference Estimation in WiFi Networks

Claudio Rossi, Claudio Casetti, Carla-Fabiana Chiasserini

Politecnico di Torino, Italy

Abstract. Identifying the cause of an underperforming WLAN can be challenging due to the presence of a plethora of devices and networks operating in the ISM band. Such devices cause electromagnetic interference that can potentially undermine the throughput of residential and enterprise WiFi networks. In this paper, we propose an interference estimation and monitoring technique for 802.11 Access Points that can be implemented without specialized hardware and without any modification to wireless stations. We validate our technique with commercial hardware, evaluating its accuracy with different types of interferers.

1 Introduction

In today's overcrowded, arbitrary deployment of home WLANs, interference is likely coming from a neighboring Access Point (AP), operating on either the same or a different frequency channel. In addition, there are many non-802.11 devices working on ISM bands and representing possible sources of interference. It follows that, in spite of the increasing availability of planning, deploying and managing tools, radio interference remains a key performance bottleneck for home and enterprise WLANs alike.

Very few tools are really helpful to understand how much interference affects the operation of a given wireless network, and how interference patterns evolve over time. To further compound this problem, whatever tools are available require expert usage and only operate as spectrum scanners, often providing little insight on the nature, causes and effects of interference. Among commercial solutions, Airmagnet Spectrum XT [1] and AirMaestro [2] are examples of custom hardware systems that integrate spectrum analyzer functionality to facilitate non-WiFi device detection. In the scientific literature, several examples of interference estimation tools based on available bandwidth testing [3–5] require traffic injection. The downside of these approaches is that they affect normal network operations and require certain traffic patterns to test interfering links, which may be incompatible with realistic traffic scenarios.

A different approach is based on trace collection and subsequent analysis. Proposed solutions aim at analyzing specific aspects of a 802.11 wireless network, ranging from physical and link-level behavior [6–8], wireless station (WS) location and coverage [9], to transport and network layer performance [10]. In [11], traces are captured using several sniffers in a WLAN and a state machine-based learning approach is proposed to identify interference. Similarly, the authors of [12] exploit a large wireless monitoring infrastructure to monitor a production WLAN and perform a cross-layer analysis to diagnose performance problems. While some of these approaches appear to be effective,

they only have offline applicability, as they require postprocessing of wireless traces to identify interfering signals. They fail to evaluate the accuracy and agility of interference estimation mechanisms, especially in presence of WS mobility and sophisticated bit rate adaption mechanisms. Also, they do not discuss the integration of their interference estimation mechanisms with applications like power control and channel assignment.

An example of online, passive interference estimation is given in [13], which presents a methodology to dynamically generate fine-grained interference estimates across an entire WLAN. However, the solution in [13] requires both a second wireless card on the APs and to compute the real-time graph of all interfering nodes. The latter implies the presence of a centralized controller for the entire WLAN, which may not be always available, especially in residential networks. Similarly, [19] uses a specific functionality provided by a recent WiFi chipset to perform online detection of multiple non-WiFi devices including fixed frequency devices (e.g., ZigBee), frequency hoppers (e.g., Bluetooth) and broadband interferers (e.g., microwave ovens).

In this work, we propose a MAC-layer approach to interference estimation by adopting passive measuring techniques. Our solution is implemented at the AP and accounts for all possible causes of interference, specifically: transmissions originated within neighboring Basic Service Sets (BSSs), either operating on the same or on a different frequency channel, and transmissions from non-802.11 devices. The key point of the proposed technique resides in the comparison that the AP performs, for each of the data frames it sends, between the expected time required to successfully transmit the frame and the actual time measured by the AP. In order to understand the impact of interference on the BSS throughput performance, we then extend the computation of saturation throughput [14,15] by accounting for the interference effects. We implement our solution in a testbed and validate it via experimental results. Unlike previous solutions, ours can be implemented on commercial APs with any WiFi chipset, without requiring either specialized hardware or modifications to the WS.

2 Inferring Interference

We consider an IEEE 802.11 BSS managed by an AP capable of monitoring local wireless resources. The AP and its associated WSs may transmit frames with different payload size and their data rate may vary according to the experienced channel propagation conditions. The AP collects statistics within its BSS and it carries out such measurements periodically over a time interval, hereinafter referred to as measurement period.

Inferring interference requires that the AP estimates whether the channel is sensed busy because of "legitimate" ongoing transmissions or because of interference. This procedure consists in computing, for each frame k sent by the AP, what is the *expected* time interval $T_e(k)$ that a transmission (either successful at the first attempt or subject to collisions/errors, hence repeated one or more times) should take, were it not affected by interference. This time computation should then be compared against actual *measurements* of transmission interval $T_m(k)$, taken by the AP driver, in order to infer whether unaccounted-for signals (interferers) are cluttering the channel. The latter unduly lengthen the transmission because the channel is sensed busy even if no legitimate BSS node is actively sending data. The estimation of the fraction of time taken by

interferers, I, can be computed as follows:

$$I = \frac{\sum_{k=1}^{K} \left(\frac{T_m(k) - T_e(k)}{T_e(k)} \right)}{K}, \tag{1}$$

where K is the total number of frames transmitted by the AP within the current measurement period. $T_m(k) = t_a(k) - t_x(k)$, where $t_a(k)$ corresponds to the ACK notification time, while $t_x(k)$ is the time at which frame k is handed over to the driver for transmission. As for $T_e(k)$, for the sake of clarity, we describe how it is computed by referring to Fig. 1.

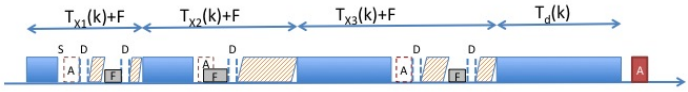


Fig. 1. Timeline of a repeated transmission by the AP. "S", "A", "D" and "F" stand for, respectively, SIFS, ACK, DIFS, and Freeze.

The figure portrays the case where the AP repeats the transmission of a frame four times before success. Each attempt is renewed after the mandatory random backoff period. It is to be underlined that each retransmission attempt could occur at a lower bit rate than the previous one. Indeed, the MAC-layer rate control procedure implemented in most 802.11 drivers mandates for each MAC frame to be associated with a retry vector. This vector specifies the number of retries to be performed at decreasing bit rates, successively attempted in case a transmission fails (i.e., no ACK is received). These rates are known by the driver. Thus, the duration of each frame transmission can be easily computed by the AP itself. We denote by $T_d(k)$ the duration of the successful transmission, which depends on the data rate and the payload size of the frame and is computed as specified in the 802.11 standard [16]. As for the failed transmissions, we indicate the i-th transmission attempt of frame k as $T_{Xi}(k)$. Such a quantity is composed by: the transmission duration of the data frame, the retransmission timeout T_o (which is set equal to SIFS plus the ACK duration), the backoff time associated with the i-th attempt, and the DIFS time intervals needed to declare the channel idle. The contribution due to the backoff time is set to half the contention window used at the i-th attempt. However, it is important to recall that the backoff counter is frozen by an 802.11 interface whenever the channel is sensed busy. Carrier sensing may be triggered by: (i) transmissions from other WSs in the same BSS, (ii) transmissions from other BSSs operating on the same channel, (iii) transmissions from neighboring BSSs using a different channel or from non-802.11 devices. Note that the first case is a "legitimate" interruption and, as such, is not classified as interference. As for data frames transmitted within other BSSs operating on the same channel, they can be received by the AP of the tagged BSS through a virtual interface operating in monitoring mode. In our computation, we separately account for such contribution and denote it by $\delta(k)$.

Based on the description and definitions above, the expected transmission interval $T_e(k)$ can be obtained as:

$$T_e(k) = \sum_{i=1}^{A} T_{Xi}(k) + T_d(k) + \text{SIFS} + \text{ACK} + \sum_{j=1}^{N} (T_{RXj}(k) + T_{NRj}(k)) + \delta(k) + \epsilon, \tag{2}$$

where:

- A is the number of failed transmissions for frame k;
- SIFS and ACK are, respectively, SIFS and ACK durations. The ACK duration is computed by considering its actual transmission rate;
- N is the number of WSs in the tagged BSS;
- $T_{RXj}(k)$ and $T_{NRj}(k)$ are the duration of, respectively, successful and failed transmission cycles by other WSs within the tagged BSS (during which the AP has to freeze its own backoff while trying to transmit frame k);
- $\delta(k)$ is the airtime taken by non-colliding transmissions from neighboring BSSs operating on the same channel, while the AP tries to transmit frame k;
- ϵ is the approximation error due to the granularity with which time intervals are detected by the AP's driver.

We remark that the AP has no knowledge of failed transmissions by WSs. However, assuming a symmetrical channel between AP and WSs, the same packet error rate (PER) may apply to any transmission, thus yielding a rough estimate of the percentage of WS transmissions that ultimately fail.

In conclusion, by computing $T_e(k)$ and measuring $T_m(k)$ for each data frame, the AP can obtain an estimate of I at each measurement interval. *This method can be effectively implemented at run time and it does not require any knowledge of the past*. Since per-frame processing and statistics are already included in any WiFi driver, our method adds only a negligible complexity. The technology we refer to can be any among a, b, and g; also, the proposed technique can be easily extended to DCF with handshake as well as to the 802.11e/n EDCA.

3 Computation of the Saturation Throughput

For a practical understanding of the impact of interference, we now introduce a simple methodology to compute the theoretical saturation throughput that would be achieved if the BSS operated in an unhindered scenario. We will then estimate the contribution of interferers and derive the *theoretical saturation throughput in presence of interference*. In Sec. 5, the latter quantity will be compared against actual live measurements.

Again, we consider an IEEE 802.11 BSS managed by an AP capable of monitoring local wireless resources at the MAC layer. As in [17,18] during each measurement period, the AP computes: average size of the frame payload (P), maximum payload size (P_{max}), average data rate for data frames and for ACKs, average PER (p_e), and

number of active WSs within the BSS (N). The AP considers a node in the BSS (either itself or a WS) to be active if the node has successfully transmitted at least one data frame within the last measurement period. The average PER could be computed based on the modulations used for the transmissions in the measurement period, their associated signal-to-noise (SNR) ratio, and assuming independent bit errors on the channel. Since this method gives poor results due to multipath effects and inaccurate SNR measurements by the hardware, we estimate p_e as the ratio of the number of erroneously received frames to the number of transmitted frames. To compute the numerator, at the receiver we count the CRC errors (at the PHY and MAC layer), while at the transmitter we count all unsuccessful transmission attempts at the physical layer. The latter results in a worst case PER estimation, as collisions are also included in the count [1]. Conversely, the computation at the receiver underestimates the actual number as the physical layer cannot always decode a corrupted frame and hand it to the MAC layer.

We now introduce the theoretical saturation throughput S_{th}, defined as the value of maximum achievable throughput in the BSS given the current traffic load. The saturation throughput is given in [15], which extends the original model in [14] to account for errors due to channel propagation conditions:

$$S_{th} = \frac{N\tau(1-\tau)^{N-1}P(1-p_e)}{E[T]}. \tag{3}$$

In (3), τ is the probability that a node (either a WS or the AP) accesses the medium at a generic time slot[2] and $E[T]$ is the average duration of a time interval in which an event occurs, namely, an empty slot, a successful transmission, a transmission failed due to channel errors, or a collision. $E[T]$ can be computed as:

$$\begin{aligned} E[T] = (1-\tau)^N\,\sigma + \\ [N\tau(1-\tau)^{N-1}(1-p_e)]T_s + \\ [1-(1-\tau)^N - N\tau(1-\tau)^{N-1}]T_c + \\ [N\tau(1-\tau)^{N-1}p_e]T_{err} \end{aligned} \tag{4}$$

where σ is the slot time duration as defined in the 802.11 standard. By assuming that the retransmission timeout is equal to SIFS plus ACK, the average duration of a successful transmission, T_s, and of an erroneous transmission, T_{err}, are equal and given by:

$$T_s = T_{err} = T_d + \text{ACK} + \text{SIFS} + \text{DIFS} \tag{5}$$

where T_d is the average frame duration computed as specified by the standard, according to the BSS type, and using the average payload size and average rate measured by the AP at the BSS level.

As far as the average collision duration is concerned, its exact computation would require the AP to be aware of the number of nodes that are hidden with respect to each

[1] Collisions cannot be discriminated from errors caused by harsh channel conditions without changing the WS software or the 802.11 protocol.

[2] Considering a slotted time is the main approximation of this model.

other. The work in [14, 15] does not account for hidden WSs and the approaches proposed in the literature, e.g., [20], are not viable in our set up, as we do not require the AP to have knowledge of the users distribution within its coverage area. Thus, we approximate the average collision duration by making a worst-case assumption. Each collision involves a data frame of maximum payload size P_{max} among those observed by the AP during the measurement period. Clearly, the average collision time is overestimated in absence of hidden WSs, leading to underestimating the saturation throughput. It follows that T_c is computed as T_{err} but using P_{max} instead of P. We also observe that the AP can easily compute τ using the following equations [15]:

$$p = 1 - [(1-\tau)^{N-1}(1-p_e)]$$
$$\tau = \frac{2(1-2p)(1-p^{m+1})}{W(1-(2p)^{m+1})(1-p) + (1-2p)(1-p^{m+1})}$$

where p is the the conditional probability that a transmitted data frame encounters a collision or is received in error in saturation conditions; W is the minimum contention window; m is the retransmission limit. While deriving our results, we set $W = 31$ and $m = 5$. Note that p and τ have to be obtained through numerical methods, as described in [14, 15]. We can pre-compute all values of τ as a function of N and p_e, and perform at each measurement period a simple look-up. For instance, if we consider N varying from 1 to 30, τ from 0.05 to 1 with 0.05 resolution, and a half precision floating point representation (16-bit) for τ, we would need only 1.2 MB memory space to store all values. As τ ranges from 0 to 1, this requirement can be further reduced by using an ad hoc code.

We stress that, although S_{th} represents the saturation throughput considering the node average behavior, it accounts for the different air time that WSs take to transmit their frames. Indeed, the average payload size P and $E[T]$ in (3) depend on the payload, data rate and access rate of each single WS.

In order to reflect the effects of all types of interferers, we let the AP keep track of I, computed as in (1), and of the average δ, computed as $\left(\sum_{k=1}^{K} \delta(k)\right)/M$ where M is the measurement period duration. We then discount from S_{th} the portion of throughput that cannot be achieved due to the two components above and, finally, compute the saturation throughput in presence of interference, S_{in}:

$$S = (1-\delta)S_{th}$$
$$S_{in} = (1-I)S\,.$$

4 System Implementation: A MAC-layer Approach

As mentioned, our solution has the following desirable properties: (i) it allows online interference detection, from both WiFi and non-WiFi devices; (ii) it does not need specialized hardware; (iii) it runs on an AP without additional software (or hardware) modification to WSs.

We implement the estimation procedure described in Sec. 2 at the MAC layer, specifically within the mac80211 module of the Linux wireless driver *compact-wireless*

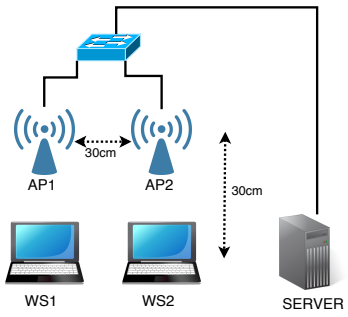


Fig. 2. Testbed deployment used for validation.

2011-21-01 [21]. We select the MAC layer because it is the highest layer in the stack from which we can retrieve the data rate for every data transmission attempt as well as for ACKs.

The estimation procedure implies the implementation in the AP driver of: (i) the additional passive measurements described in Secs. 2 and 3, and (ii) the estimation of I. All required measurements, along with I, are made available to the application layer by using debugfs at each measurement period. Note that, since we modify only the mac80211 module and not the AP hardware or physical driver, such measurements can work on any wireless chipset. We then implement the computation of S and S_{in} with a simple user-space program that reads the measurements from file system and process them as described in Sec. 3. Given its low complexity, such computation can be implemented on any device.

5 Experimental Evaluation

We evaluate the validity of our approach through a testbed deployed in a university laboratory at Politecnico di Torino. In the laboratory, there are 18 detectable APs, which are part of 6 different SSID, whose signal is received at an average strength of -83 dBm. We use two stations, WS1 and WS2, each associated to a 802.11g AP. We name the access points, AP1 and AP2, and refer to their BSSs as BSS1 and BSS2, respectively. Each AP runs the modified driver for the computation of I and the application for the computation of S and S_{in}. We connect the APs to a switch, which, in its turn, is connected to a desktop PC that we use as a traffic sink. All equipment is placed on a desk, at approximately 1 m height. AP1, AP2, WS2 and WS1 are placed at the vertices of a square of 30 cm side, in clockwise order starting from the right-top vertex (see Fig. 2).

APs are implemented in embedded wireless nodes featuring an Alix PC Engines motherboard, equipped with an AMD Geode 500 MHz processor and a IEEE 802.11 b/g compliant Wistron DCMA-82 Atheros wireless card. Each Alix runs OpenWrt Backfire, a Linux distribution for embedded devices. WSs are represented by ASUS notebooks, model P52F, with Ubuntu 12.04. Both APs are powered through PoE (Power

over Ethernet). BSS1 operates on channel 6 and is our tagged BSS, while BSS2 acts as an interferer. In particular, BSS2 operates either on the same channel as BSS1, or on another among the 802.11 standard ones (in the 2.4 GHz band). We set the transmit power of both APs and WSs to 20 dBm. Since we are interested in evaluating the throughput loss caused by interferers and the accuracy of our solution in estimating I and S_{in}, we saturate BSS1's capacity with a downlink UDP flow from the server to WS1 (through AP1), at a rate of 30 Mbps. The choice of UDP, rather than TCP, traffic allows us to precisely control the load without unpredictable effects due to congestion control mechanisms.

We start by setting BSS2 on channel 11 so as to assess the performance of our solution in presence of 802.11 interferers operating on a different frequency channel. Specifically, we consider a dynamic traffic scenario where no traffic is generated within BSS2 in the interval [0, 25 s], then the server starts transmitting a UDP downlink flow at 30 Mbps to WS2 (through AP2). The results are shown in the top plot of Fig. 3, which compares the throughput achieved by AP1 (Thr) to the saturation throughput estimations S and S_{in}. Recall that S_{in} differs from S as it accounts for the interference term I. The bottom plot of Fig. 3 instead depicts (i) the estimated time fraction I during which the channel within BSS1 is sensed as busy due to BSS2, (ii) the average PER (p_e), and (iii) the time fraction δ during which AP1 senses the medium as busy due to other BSSs operating on the same channel. Note that δ is determined by those BSSs, out of the 18 that are present, that operate on the same channel as BSS1 (channel 6). All these quantities are expressed as percentages.

We observe that S_{in} closely matches the throughput measured by AP1 (Thr) before and after the interfering flow is enabled within BSS2. This clearly indicates that I correctly reflects the negative effect of a flow activated within a BSS operating on a different frequency channel (with BSS2 achieving an average aggregate throughput of 15.7 Mbps). It is also important to remark that the quantitative impact of interferers operating on non-overlapping channels may be severe, especially in the case of devices in close proximity (as in our case). Furthermore, when the interferer is enabled, not only I but also p_e increases. This suggests that transmissions within BSS2 may cause collisions at AP1, beside making AP1 detect the channel as busy.

We now extend our evaluation by varying both the channel used by the interferer (BSS2) and its offered load. We set AP2 to operate on a different channel at each run, namely, 6, 7, 9 and 11. For each channel, we carry out several experiments (each lasting 50 s) so as to vary the generation rate of the interfering traffic (i.e., the rate of the UDP downlink traffic flowing from the server to WS2). In all cases, we obtain an excellent match between the throughput achieved by AP1 and the estimated saturation throughput S_{in}. Due to lack of space, we only plot in Fig. 4 the comparison between the throughput measured by AP1 (Thr), S and S_{in} for channel 6 (top) and 11 (bottom). The results are presented as functions of the offered load in BSS2 (i.e., the interfering traffic generation rate). Each point plotted in the figure is the average over time of the values obtained during a 50-s experiment. Clearly, Thr decreases as the interferer traffic load increases. Again, S accurately estimates the throughput of BSS1 only in case of co-channel interference, i.e., when BSS2 operates on channel 6. Indeed, in this case the majority of the frames transmitted by AP2 are received by AP1's monitoring interface, which can

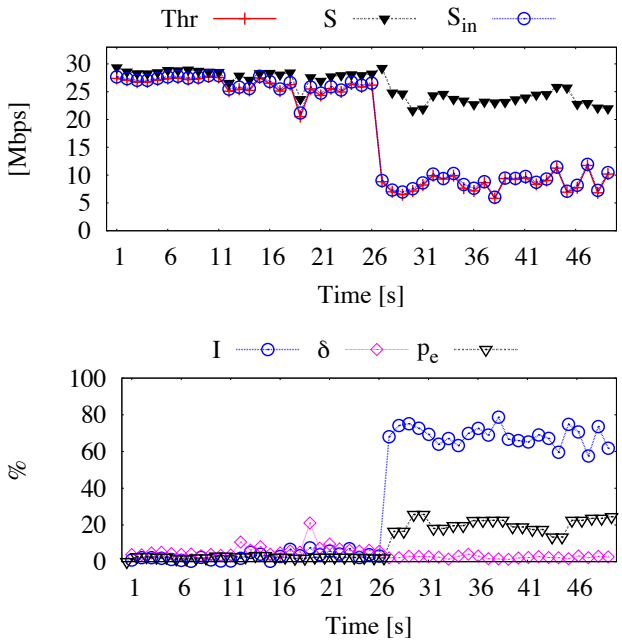
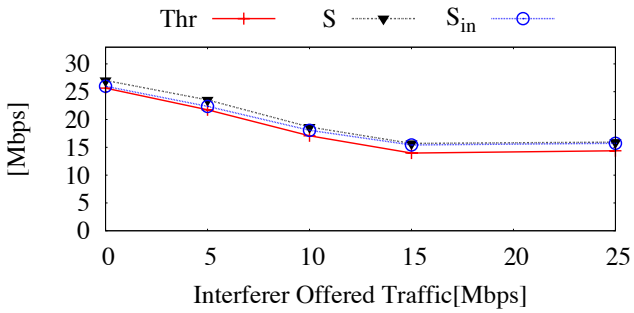


Fig. 3. 802.11 interferer enabled at second 26 on a non-overlapping channel (ch11). Top: comparison between BSS1 throughput (Thr), the estimated saturation throughput S and that accounting for interference (S_{in}). Bottom: average PER (p_e) and time fractions during which the channel is sensed as busy due to co-channel interference (δ) and to BSS2 operating on ch11 (I).

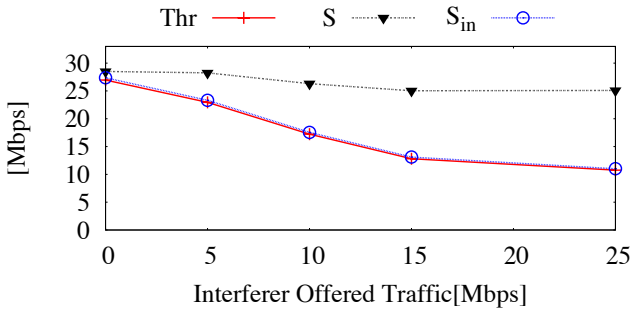
correctly account for it through the quantity δ. Conversely, when the interferer operates on channel 11, S cannot reveal its presence, while S_{in} perfectly reflects its effects.

Figs. 5 (a) and (b) show the value of the time fractions I and δ, as the channel on which BSS2 operates and BSS2's traffic load vary. In case of co-channel interference (i.e., BSS2 on ch6), I is low as this quantity does not account for it; conversely, δ well captures such interference. Furthermore, we observe that the traffic load within the interfering BSS has a significant impact and this is correctly represented by both I (for channels other than 6) and δ (for channel 6).

Fig. 5(c) presents the results obtained with BSS2 operating also on channels 1, 3, and 5 and setting the rate of its traffic flow to 25 Mbps. We note that the behavior of S_{in} closely matches that of the measured throughput Thr on all channels. The values of Thr however change significantly depending on the considered channel: this is due to the different multipath conditions affecting the channels. We repeated the experiments placing the devices at different locations and we obtained similar results, which we omit for brevity. The fact that S_{in} and Thr consistently match suggests that our methodology provides an accurate estimation no matter the working environment that is considered.



(a) Achieved vs. saturation throughput on ch6

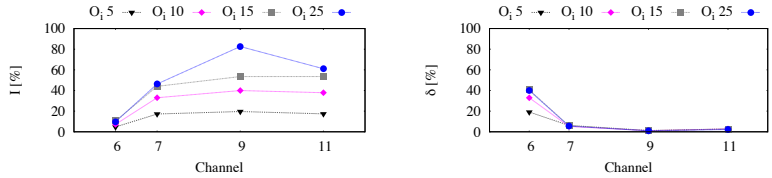


(b) Achieved vs. saturation throughput on ch11

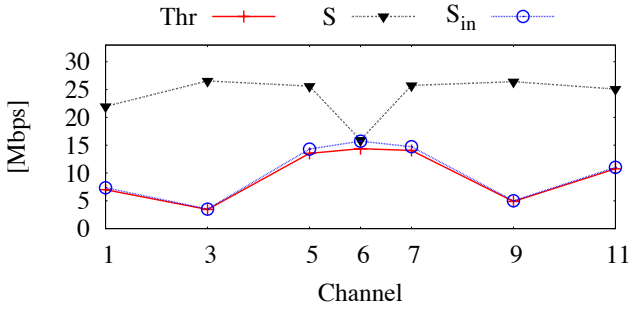
Fig. 4. Achieved vs. saturation throughput in BSS1 when BSS2 operates on the same channel (ch6) (a) and on a different channel (ch11) (b).

Next, we focus on non-802.11 interferers and employ a pair of Bluetooth nodes and an analog video sender. First, we start a file transfer between the Bluetooth nodes, with the sender and the receiver placed, respectively, at the left and right side of WS1 and equally spaced from WS1 by 30 cm. As before, the file transfer starts at second 26 and the experiment lasts 50 s. As shown in Fig. 6, initially the interfering traffic flow causes a noticeable throughput degradation at AP1, but after a couple of seconds the performance improves again. This effect is due to the Bluetooth adaptive frequency hopping (AFH) scheme, which tends to avoid channels characterized by high PER. Again, S_{in} closely follows the behavior of Thr in all phases of the experiments, confirming the validity of our technique.

We then let the video sender act as interferer. It is placed on the left of WS1, at 30 cm distance, turned on at second 26. The interference generated is so strong that the throughput drops almost to zero after a couple of seconds. Again, S_{in} is able to quickly reflect the behavior of the measured throughput, as highlighted by the results in Fig. 7.



(a) Interferer channel occupation, I (b) δ



(c) Achieved vs. estimated saturation throughput

Fig. 5. (a) Interference estimation I and (b) time fraction during which AP1 detects the channel busy due to co-channel interference (δ). The results are plotted as the channel used by BSS2 and its traffic load vary. (c) BSS1 throughput (Thr) vs. estimated saturation throughput S and S_{in}, as the channel used by BSS2 varies.

6 Conclusions

We have designed and implemented a technique for interference estimation in 802.11 WLANs, which accounts for all possible sources of interference. It can be implemented at the access point and does not require any specialized hardware, changes in the 802.11 standard or in the wireless stations. We have validated our technique in a 802.11g network with different types of interferers. Experimental results show that our solution can estimate the impact of interference with excellent accuracy, under different scenarios.

Acknowledgements

This work was partly supported by Telecom Italia – Home Network and Wireline Devices and partly by the Italian Ministry of Education through the PRIN project GATECOM.

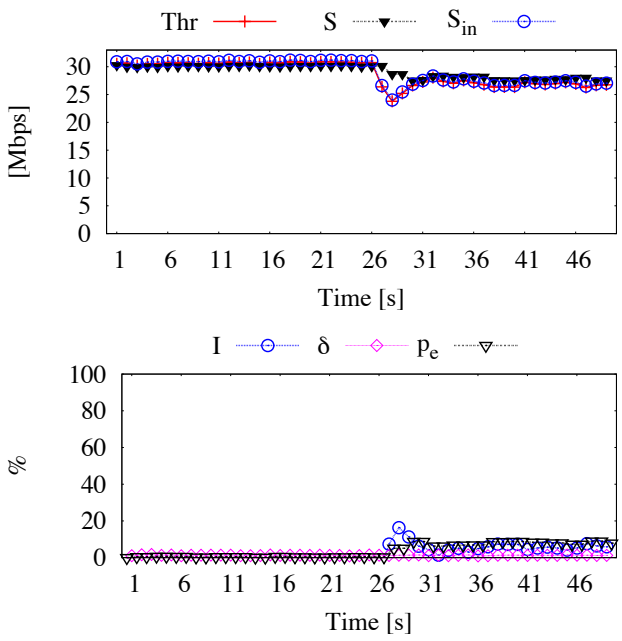


Fig. 6. Bluetooth file transfer enabled at second 26. Top: comparison between BSS1 throughput (Thr), estimated saturation throughput (S) and that accounting for the interference effect (S_{in}). Bottom: average PER (p_e), and time fractions during which AP1 senses the channel as busy due to co-channel interference (δ) and to the Bluetooth interferer (I).

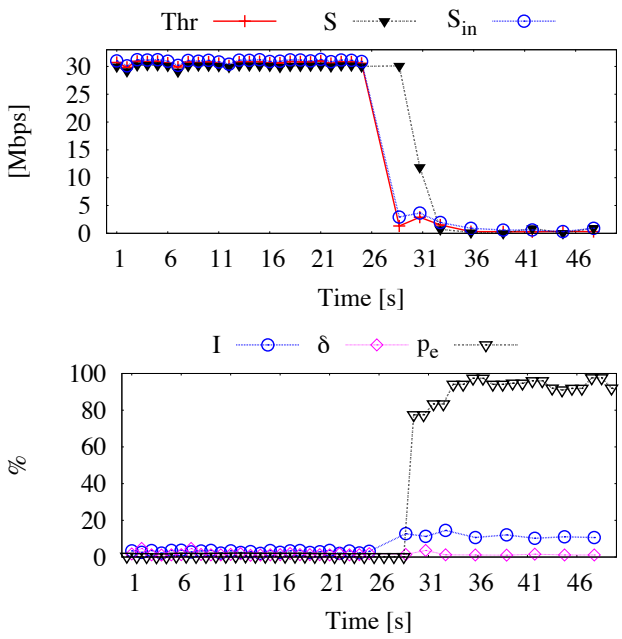


Fig. 7. Analogue video sender enabled at second 26. Top: comparison between BSS1 throughput (Thr), estimated saturation throughput (S), and that accounting for the interference effect (S_{in}). Bottom: average PER (p_e), and time fractions during which AP1 senses the channel as busy due to co-channel interference (δ) and to the interference caused by the video sender (I).

References

1. AirMagnet Spectrum XT, www.airmagnet.net/products.
2. Bandspeed AirMaestro spectrum analysis solution. http://www.bandspeed.com/.
3. D. Niculescu, "Interference map for 802.11 networks," IMC 2007.
4. J. Padhye et al., "Estimation of link interference in static multi- hop wireless networks", *IMC,* 2005.
5. N. Ahmed et al., "Online estimation of RF interference," *ACM CoNext,* 2008.
6. A. Sheth et al., "MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs," *MobiSys,* 2006.
7. D. Aguayo et al., "Link-level measurements from an 802.11b mesh network," *SIGCOMM,* 2004.
8. M. Vutukuru, K. Jamieson, and H. Balakrishnan, "Harnessing exposed terminals in wireless networks," *NSDI,* 2008.
9. R. Chandra, J. Padhye, A. Wolman, and B. Zill, "A location- based management system for enterprise wireless LANs," *NSDI,* 2007.
10. Y.-C. Cheng et al., "Automating cross-layer diagnosis of enterprise wireless networks," *ACM SIGCOMM,* 2007.
11. R. Mahajan et al., "Analyzing the MAC-level behavior of wireless networks in the wild," *ACM SIGCOMM,* 2006.
12. Y.-C. Cheng et al., "Jigsaw: solving the puzzle of enterprise 802.11 analysis," *ACM SIGCOMM,* 2006.
13. V. Shrivastava, S. Rayanchu, S. Banerjee, and D. Papagiannaki, "PIE in the sky: online passive interference estimation for enterprise WLANs," *USENIX,* 2011.
14. G. Bianchi, "Performance analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE JSAC,* vol. 18, no. 3, pp. 535–547, Mar. 2000.
15. P. Chatzimisios, A. C. Boucouvalas, V. Vistas, "Performance analysis of the IEEE 802.11 DCF in presence of transmission errors," *IEEE ICC,* 2004.
16. 802.11 standard, http://standards.ieee.org/about/get/802/802.11.html
17. C. Rossi, C. Casetti, C.-F. Chiasserini, "Bandwidth Monitoring in Multi-rate 802.11 WLANs with Elastic Traffic Awareness," *GLOBECOM*, 2011.
18. C. Rossi, C. Casetti, C.-F. Chiasserini, "Energy-efficient Wireless Resource Sharing for Federated Residential Networks," *WoWMoM*, 2012.
19. S. Rayanchu, A. Patro, and S. Banerjee, "Airshark: Detecting non-WiFi RF devices using commodity WiFi hardware," *IMC,* 2011.
20. A. K. Mahani, M. Naderi, C. Casetti, C.-F. Chiasserini, "MAC layer channel utilization enhancements for wireless mesh networks," *IET Communications,* vol. 3, no. 5, pp. 794–807, May 2009.
21. Wireless driver, http://linuxwireless.org/