Machine Learning Based Network Analysis using Millimeter-Wave Narrow-Band Energy Traces

Maria Scalabrin†, Guillermo Bielsa‡, Adrian Loch‡, Michele Rossi†, Joerg Widmer‡

***Abstract*—Next-generation wireless networks promise to provide extremely high data rates, especially exploiting the so-called millimeter-wave frequency range. Gaining information from spectrum usage is becoming important to provide smart adaptation capabilities to future network protocol stacks. Issues such as deafness, misaligned antennas, or blockage may severely impact network performance, and their identification is crucial. Despite the complexity of full analytical models, machine learning techniques are progressively being considered to improve spectrum usage at higher layers. In this paper, we design a signal processing technique that uses narrowband physical layer energy traces, obtained from one or multiple channel sniffers. The proposed technique utilizes a combination of template matching and an Explicit Duration Hidden Markov Model (EDHMM) to correctly classify frames, while coping with the non-stationarity of the traces. This leads to a protocol level monitor that does not need to decode the channel at the physical layer, but just infers the type of packets that are exchanged based on sub-sampled energy traces. The performance of this framework is evaluated using off-the-shelf mm-wave wireless devices, quantifying its detection performance in the presence of one or multiple sniffers, and assessing the impact of physical layer parameters such as noise power and signal levels.**

***Index Terms*—millimeter-wave, Hidden Markov Model, physical layer, energy traces, machine learning, protocol analyzer**

I. Introduction

Next-generation wireless networks are called to provide extremely high data rates, especially exploiting the so-called millimeter-wave frequency range [1]. Applications and services will benefit from these high rates and the radio spectrum will become more and more densely utilized. As wireless networks turn into increasingly complex systems, accurate and scalable analytical models to characterize their behavior are not yet available and very challenging to obtain. Instead, a promising approach is provided by machine learning tools that learn from data [2]–[4]. Gaining information from spectrum usage is deemed important to provide smart adaptation capabilities to future network protocol stacks. Possible applications include: (i) channel diagnosis: detect communication problems such as a link blockage, (ii) Quality of Service (QoS) tracking and adaptation, i.e., efficiently manage channel resources according to the detected energy level, (iii) information security: discovery of malicious signaling messages, etc.

As for the millimeter-wave (mm-wave) channel, its directional nature results in communication issues that strongly impact higher layers, but which are hard to identify without detailed information of the underlying physical layer effects. This includes, e.g., deafness [5], misaligned antennas, and link blockage. Commercial Off-The-Shelf (COTS) devices are typically a black box regarding this physical layer information. As a result, troubleshooting COTS-based, real-world mm-wave network deployments often translates into time-consuming "trial-and-error" analysis. While understanding performance issues in such deployments is challenging [6]–[8], the resulting knowledge is extremely valuable. It provides useful insights for network planners and administrators. For instance, a missing acknowledgment after a data packet hints at a deafness issue, overlapping packet frames suggest a collision, and so on. However, gaining such insights from a COTS node that forms part of the network is virtually impossible. On top of the aforementioned lack of lower-layer access, a single node would be restricted to its particular point of view—the directivity of the communication limits the insights that one could gain. To prevent this, we need to capture and compare the behavior of the network from multiple viewpoints. Given the extreme bandwidth available in mm-wave communications (e.g., 2 GHz per channel in the 60 GHz band), this requires an inordinate amount of data processing, and thus would be highly challenging.

In this paper, we start filling the above identified gaps through the design and evaluation of an automatic tool for mm-wave channel analysis based on COTS hardware. Specifically, the tool uses machine learning techniques to infer protocol-level details such as packet types, their energy level and duration, and can help detect performance bottlenecks in 60 GHz networks using *narrowband* physical layer energy traces from one or multiple (omnidirectional) sniffers. That is, we do not record and decode the full communication but only require the energy level that the sniffers receive.

Our key contribution is developing a machine learning framework that correctly classifies the transmitted frame types (data, acknowledgements and training sequences) and can help infer network issues. This is far from trivial due to (a) the non-stationarity of the traces and (b) the complexity of the IEEE 802.11ad protocol [9]. To address (a), we dynamically update the parameters of the underlying machine learning model such that it adjusts to variations in the received energy level due to, e.g., node movement. Regarding (b), we use a combination of template matching and an Explicit Duration Hidden Markov Model (EDHMM) to correctly classify frames. The core idea of our approach is also applicable to networks operating at lower frequencies such as IEEE 802.11ac. Still, in this paper we focus on the mm-wave case, which is more challenging due to the use of directional antennas and the large

†Department of Information Enginering, University of Padova, via Gradenigo 6B, 35131 Padova, Italy. Email: {scalabri, rossi}@dei.unipd.it. ‡IMDEA Networks Institute, Avenida del Mar Mediterraneo 22, 28918 Leganes, Madrid, Spain. Email: {guillermo.bielsa,adrian.loch,joerg.widmer}@imdea.org.

bandwidth. Since we do not need to decode any of the data, our approach preserves privacy, works regardless of whether the network uses encryption, and does not require accurate time/frequency synchronization. As a result, the proposed technique is simple yet highly effective. Our contributions are summarized as follows:

- We design a machine learning framework based on template matching and an EDHMM to automatically infer protocol layer information, e.g., transmitted packet types, their energy and duration, in 60 GHz networks. The main challenge lies in the variability of the traces and in the complexity of identifying the structural elements in the traces given their noisiness, aperiodicity, and unpredictable behavior. Here, this is solved through a combination of statistical and machine learning techniques.
- We introduce a time-adaptive learning mechanism to cope with the non-stationarity of energy traces due to gain control adjustments and node movement. This run-time adaptation is barely explored in specialized work in the field of statistics but is critical for our approach. It also sets our scheme apart from existing work based on simple clustering or thresholding, which is highly sensitive to non-stationary behavior and thus often fails.
- We extend the learning framework to *jointly* process mm-wave channel traces from multiple time-synchronized sniffers. The idea is that different viewpoints of the same channel can provide diverse information and lead to higher decoding accuracies.
- We evaluate our approach in an extensive measurement campaign using COTS 60 GHz hardware to analyze its performance in a range of practical scenarios. Besides, we numerically quantify the ability of the framework to correctly identify protocol sequences (beacon pairs, data and acknowledgment frames) from single and multiple channel sniffers, looking at the impact of channel noise and its distribution across different sniffers.

This paper is organized as follows. The related work is surveyed in Section II. In Section III, we discuss typical IEEE 802.11ad energy traces. Some mathematical background on the standard HMM and the extended EDHMM framework is provided in Section IV. The machine learning framework is introduced in Sections V, VI, VII and VIII. In Section IX, this framework is generalized to perform decoding from multiple sniffers and a mm-wave channel trace generator that helps to evaluate the accuracy of our approach is presented in Section X. We finally evaluate the proposed technique using experimental and simulated energy traces in Section XI and provide some concluding remarks in Section XII.

II. Related work

In the following, we give an overview of performance analysis and troubleshooting in mm-wave networking. As sketched in Section I, mm-wave networks suffer from high path-loss and high absorption. To overcome this, nodes typically use directional antennas and Line-Of-Sight (LOS) paths. However, this makes links very susceptible to blockage. State-of-the-art work in this field [10]–[12] focuses on correctly identifying such blockage at the nodes involved in the communication, and reacting in a timely manner. For instance, BeamSpy [11] measures the set of available paths between a transmitter and a receiver. This "path skeleton" serves as a reference whenever blockage occurs—the nodes compute which of the paths in the skeleton is most likely to be unaffected by the blockage and steer their antennas accordingly. As a result, BeamSpy can avoid costly beam steering overhead. Further, earlier work by the same authors [12] looks into differentiating device movement from blockage based on Received Signal Strength (RSS) measurements. This is key to ensure that nodes react correctly when links degrade. Similarly, MOCA [10] transmits a very short control message to assess the link state. If the transmitter does not obtain a reply, it assumes that the antennas are misaligned. Otherwise, it adapts the Modulation and Coding Scheme (MCS) according to the current channel state. All of the above approaches aim at improving the performance of mm-wave networks. In contrast, our work *troubleshoots* the operation of such approaches and is thus orthogonal to them. While BeamSpy and MOCA also try to identify specific issues in the communication, they are constrained to the specific "viewpoint" of a certain node. Our framework runs on one or more external sniffers which we can place at multiple locations, thus providing richer insights. Earlier work proposes an equivalent concept based on external sniffers. However, such approaches typically consider lower frequency bands, and focus on security issues [13], [14] such as realizing an Intrusion Detection System (IDS). The key difference to our study is that such security sniffers are designed to continuously operate along with the network, thus increasing the complexity of the deployment. In contrast, our tool does not need to be part of the network, and can be used on-demand only. Hence, we do not add complexity to the network. Moreover, we focus on performance issues in directional wireless networks while the above work deals with security in the omni-directional case. However, [14] and references therein also deal with raw physical layer data, similarly to our case. Specifically, they suggest overhearing the communication and jamming unwanted packets based on, for instance, header information. Our narrow band sniffer for wide band signals also overhears the communication but does not (and in fact cannot) decode any preambles and headers to identify the packets. Instead, we use machine learning on the timing of frames from simple energy traces to obtain the information required for network analysis.

In recent years, machine learning techniques are increasingly being used to address high-dimensional problems with multiple unpredictable factors, e.g., for traffic classification, and previous papers typically use it after demodulating and decoding frames at the physical layer [15], [16] (although these approaches are not tailored to IEEE 802.11ad). In contrast, our framework uses machine learning on the raw physical layer trace. Thus, it eliminates the need for the above operations, which are particularly complex and resource intensive in mm-wave networks and would require a sufficiently wide band channel sniffer. However, this poses additional challenges, such as identifying frames. In this paper, we provide solutions to these challenges, which sets us apart from existing work. On top of the latest trends in wireless communications, ma-

chine learning based solutions for spectrum sensing/sharing in Cognitive Radio (CR) represent a promising approach for improving the utilization of the radio electromagnetic spectrum [17], [18]. To promote this, the Defence Advance Research Projects Agency (DARPA) [19] intends to develop technologies for extensive spectrum sensing/sharing, both in the Radio Frequency Machine Learning Systems (RFMLS) program [2] as well as in another major DARPA effort known as the Spectrum Collaboration Challenge (SC2) [3], which is regarded as the first-of-its-kind collaborative machine-learning competition to overcome spectrum scarcity. Also the National Science Foundation (NSF) [4] is promoting projects to leverage machine learning solutions in CR. In the literature, automatic network recognition offers a promising framework for the integration of cognitive concepts at the network layer, bearing similarities with the mm-wave channel analyzer proposed in Section V. In [20], the authors address the problem of automatic classification of technologies, with particular focus on Wi-Fi vs. Bluetooth recognition. Previous work, as for example [21], has addressed a related problem, allowing cooperative spectrum sensing in peer-to-peer cognitive networks by using distributed detection theory [22] for identifying overlapping air interfaces based on time-frequency analysis and feature extraction. The same problem is tackled in [23], where two kinds of neural classifiers are adopted. Again, the authors focus on Wi-Fi vs. Bluetooth recognition. While this work is related to ours, none of these studies perform protocol analysis from narrow band channel traces. We remark that our tool is the first automatic classifier of IEEE 802.11ad energy traces for network diagnosis. The uniqueness of our approach prevents direct comparison with earlier work.

III. A LOOK INTO IEEE 802.11AD ENERGY TRACES

The IEEE 802.11ad standard operates at 60 GHz. In this band, propagation conditions are worse than at lower bands, such as at 2.4 or 5 Ghz, which are used by the IEEE 802.11n/ac standards [24]. Specifically, the path loss is much higher at 60 GHz than at 2.4 or 5 GHz. To compensate for this attenuation, IEEE 802.11ad provides a mechanism for the establishment of a directional communication link between a transmitter/receiver pair using a beam training process. As a result of this process, the transmitting station focuses its energy towards the intended receiver. This compensates for the high path loss and reduces the potential interference to other stations that are located nearby.

IEEE 802.11ad divides the channel access into so called Beacon Intervals (BIs). Each BI is split into different access periods, which have different access rules and provide specific functionalities to the stations (STAs) within communication range. A typical BI is composed of a Beacon Header Interval (BHI) and a Data Transmission Interval (DTI). The BHI contains several sub-intervals and is basically used to transmit control messages, such as *beacons* that enable beam training. In the DTI period, STAs exchange data frames either exploiting a contention-based access period or a scheduled service period. The former entails a contention mechanism ("floor acquisition") to acquire the medium, which uses the enhanced distributed coordination function. Conversely, in the scheduled service period, stations access the channel in a contention-free manner.

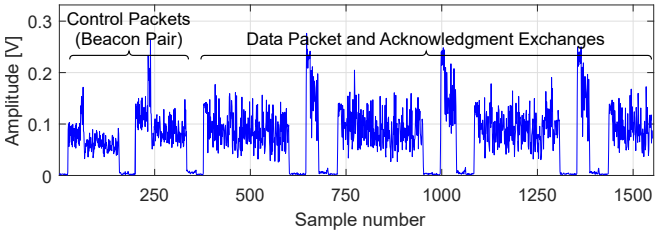

Fig. 1. Energy trace example: DATA/ACK burst.

An example IEEE 802.11ad energy trace corresponding to a data exchange is shown in Fig. 1. This trace depicts the start of a typical data burst. The data burst starts with a pair of beacons which contain control information. This pair of beacons is followed by a sequence of data (DATA) packets and acknowledgments (ACKs). In general, each DATA packet is followed by a corresponding ACK, which is the shorter packet in the figure and which has a higher energy level. Note that the higher amplitude of ACKs is due to the position of the sniffer, which in this case was near the receiver. The beam training process is composed of the following two phases.

- **Sector Level Sweep (SLS)**. During the SLS, a STA selects a *coarse grain* antenna sector. This phase can be implemented in two ways: 1) through a transmit sector sweep (TXSS), i.e., a STA tries to select the best transmit antenna sector towards a particular receiving STA by transmitting Sector Sweep (SSW) frames using each of its antenna sectors or 2) through a receive sector sweep (RXSS), i.e., a receiving STA trains its receive antenna sector by requesting its peer STA to transmit SSW frames using a fixed antenna pattern, while the receiving STA sweeps across its receive antenna sectors.
- **Beam Refinement (BR)**. To refine the sectors obtained in the SLS phase, multiple mechanisms are used. Basically, the two communicating STAs iteratively search for the optimal alignment starting from the coarse grain sector provided by the SLS. Occasional BR sequences retrain the antenna beams in case of, e.g., mobility, to ensure that both nodes remain in the boresight of each other.

Sequences of control packets are not difficult to identify within IEEE 802.11ad channel traces. The SLS sweeps that are used during the connection setup have 32 different energy levels. The BR sequences, which are used for re-alignement, e.g., when there is a drop in the link quality, have 35 levels [25]. The particular number of energy levels depends on the number of sectors of the antenna. Existing hardware implements the above number of sectors. An example energy trace corresponding to a BR phase is shown in Fig. 2.

In general, it is not difficult to recognize the individual frame types (beacons, DATA, ACKs, and BR sweeps) in the energy traces by visual inspection. This enables one to infer the dynamics of the communication. For instance, a missing acknowledgment after a data packet hints at a deafness issue, overlapping packet frames suggest a collision, and so on. However, while this is visually evident, manually inspecting the energy traces is infeasible given the number of packets

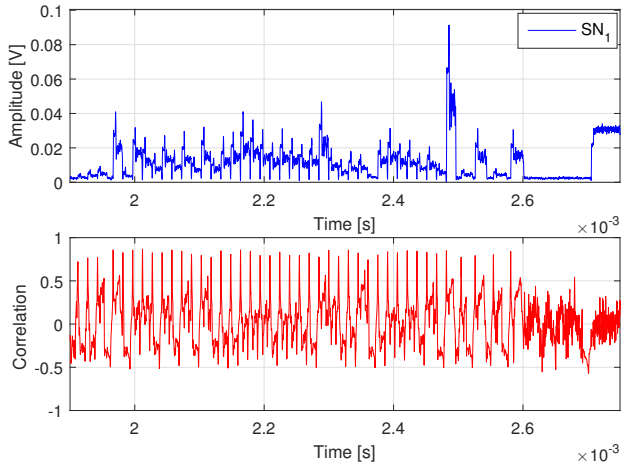

Fig. 2. Example IEEE 802.11ad beam refinement sequence (35 energy levels, top) and correlation coefficient (bottom) with respect to the beacon template.

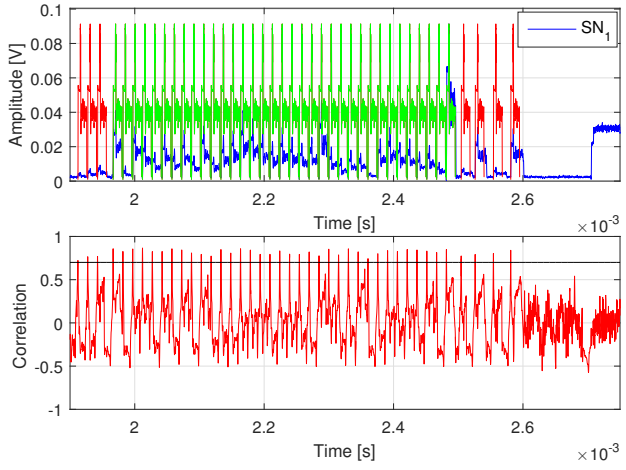

Fig. 3. Beam Refinement (BR) sequence of Fig. 2 (top) and corresponding correlation coefficient (bottom) with respect to the beacon template. A correlation threshold (horizontal line in the bottom plot) is used to single out beacon messages (top graph), whereas the inter-beacon distance reveals whether a beacon is part of the BR sweep. The BR sequence has 35 energy levels and is correctly identified, see the green line in the top plot.

when communicating at multi-gigabit-per-second rates. At the same time, recognizing frame types in an automated manner is hard. In this paper, our goal is to devise and validate a technique for the *automatic identification and labeling of IEEE 802.11ad energy traces*. Notably, BR/SLS sweeps can be reliably identified through a standard pattern matching technique, as we briefly describe next. Each sequence is composed of beacon frames, each having a different energy level, but all of them having the same (although noisy) *distinctive shape*. To capture this shape, we obtained a beacon template, that is basically a smoothed out version of the beacons that were measured experimentally. Hence, a standard convolution is performed between the input energy trace and the *beacon template*; for an example see the bottom plot in Fig. 2. As we show in Fig. 3, setting a proper threshold on the correlation signal allows one to single out the start of each beacon in the original sequence. It is then not difficult to check when exactly 32 or 35 properly spaced energy levels appear in a row and, in turn, detect the SLS/BR sweeps, see again Fig. 3. Further details on the template matching procedure are given in Section VI, whilst additional results on BR sequence detection are provided in Section XI-A.

While the identification of SLS/BR sweeps is doable through simple processing techniques, the characterization of DATA exchange phases is much more complex. In this case, we do not know in advance the duration of DATA frames. Similarly, we do not know the number of DATA/ACK exchanges in the data burst. There may be missing ACKs and the energy levels of ACKs and DATA frames can be arbitrary, as they depend on the location of the sniffer with respect to the transmitter and the receiver. In addition, the start of each data burst has to be reliably identified, and the start and end points of each frame transmitted therein have to be reliably assessed as well. All of this leads to a sequential estimation problem that is the subject of the work that we expound in the following sections.

IV. EDHMM PRELIMINARIES

Next, some mathematical background on the standard HMM and the extended EDHMM framework is provided as a basis for the machine learning framework, which is introduced in Sections V, VI, VII and VIII. Specifically, we demonstrate how the standard HMM is inadequate for our purpose. Still, we use it to calibrate the initial EDHMM.

We use uppercase and calligraphic fonts for sets, except for $\mathcal{N}(X;\mu,\sigma^2)$, which refers to a Gaussian random variable X with mean μ and variance σ^2. We denote a random sequence of length T by $X_{1:T}=(X_1,\ldots,X_T)$, where the random variable X_t at time index $t\in\{1,\ldots,T\}$ takes values in the set $\mathcal{X}$, with cardinality $|\mathcal{X}|$. Realizations are indicated by lowercase letters, i.e., x_t is the realization of X_t, and with $x_{1:T}=(x_1,\ldots,x_T)$ we denote a sequence of realizations. Vectors are indicated by bold letters, e.g., $\boldsymbol{b}$, and we refer to their elements as $\boldsymbol{b}=[b_1,\ldots,b_K]$, with $|\boldsymbol{b}|=K$. For matrices we use uppercase bold letters, e.g., $\boldsymbol{A}=\{a_{ij}\}$ is a matrix with elements a_{ij}.

Markov models, whose states correspond to observable events, are inadequate to solve our mm-wave channel estimation problem. The reason is that we measure a noisy version of the transmitted energy levels, as they are corrupted by random channel fluctuations. Instead, Hidden Markov Models (HMMs) [26] are a more appropriate tool, as their observations are probabilistic functions of the (hidden) state. Specifically, an HMM is composed of embedded stochastic processes, where an unobservable hidden random process is revealed to the observer through another set of random processes that produce the sequence of observations.

We now consider a data burst and aim to solve the following estimation problem. The observed channel samples in the data burst, $O_{1:T}=(O_1,\ldots,O_T)$, are modeled as a sequence of real-valued random variables corresponding to one of the following basic elements: "1" inter-frame space (IFS), "2" data packet (DATA) and "3" acknowledgement (ACK). Accordingly, the hidden state S_t at time t is a discrete random variable that can take values in the set $\mathcal{S}=\{1,2,3\}$. We define $S_{1:T}=(S_1,\ldots,S_T)$ as the sequence of random variables describing the hidden states in the data burst, i.e.,

$t \in \{1, \ldots, T\}$. Our objective is then to reliably estimate the sequence of hidden states $s_{1:T} = (s_1, \ldots, s_T)$ from observations $o_{1:T} = (o_1, \ldots, o_T)$. The standard HMM makes two basic assumptions regarding the embedded stochastic processes:

A1) The first assumption is that $S_{1:T}$ is a first-order Markov chain, i.e., $P(S_{t+1}|S_1, \ldots, S_t) = P(S_{t+1}|S_t)$. In particular, we have $P(S_{t+1} = j|S_t = i) = a_{ij}$, where $\boldsymbol{A} = \{a_{ij}\}$, $i, j \in \mathcal{S}$, is the single-step transition probability matrix of the HMM.[1]

A2) The second assumption is that the random variable O_t is statistically independent of $(O_1, \ldots, O_{t-1})$.[2]

Moreover, O_t is a probabilistic function of the hidden state S_t, i.e., it obeys a suitable conditional probability $P(O_t|S_t)$ and each random variable O_t can use a private distribution $P(O_t|S_t)$ over the hidden state. We use a Gaussian observation model with $P(O_t|S_t = i) = \mathcal{N}(O_t; \mu_i, \sigma_i^2)$, where μ_i and σ_i^2 specify the mean and the variance of the random variable O_t, given that the hidden state is $i \in \mathcal{S}$. This is known to well approximate the noise distribution for mm-wave channels [27]. For all hidden states $i \in \mathcal{S}$, we collect the parameter pairs $b_i = (\mu_i, \sigma_i^2)$ through vector $\boldsymbol{b} = [b_1, \ldots, b_{|\mathcal{S}|}]$. We define $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_{|\mathcal{S}|}]$, where π_i is the probability that the HMM is in state $i \in \mathcal{S}$ in the first time slot of the burst.

The HMM model is described through a further parameter vector $\boldsymbol{\Theta} = [\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{b}]$. Its maximum likelihood estimate given a sequence of observations is obtained through the Expectation-Maximization (EM) algorithm [28], which entails two-step iterations. Briefly, initial values for $\boldsymbol{\Theta}$ are chosen, and using assumptions A1 and A2 the posterior distribution for the whole sequence $P(S_{1:T}|O_{1:T}, \boldsymbol{\Theta})$ is computed. Hence, this posterior is used to compute the expected log-likelihood (the Baum's auxiliary function), $Q(\boldsymbol{\Theta}^{\text{new}}, \boldsymbol{\Theta})$, as

$$Q(\boldsymbol{\Theta}^{\text{new}}, \boldsymbol{\Theta}) = \sum_{S_{1:T} \in \mathcal{S}^T} P(S_{1:T}|O_{1:T}, \boldsymbol{\Theta}) \log P(S_{1:T}, O_{1:T}|\boldsymbol{\Theta}^{\text{new}}), \tag{1}$$

which is finally maximized with respect to $\boldsymbol{\Theta}^{\text{new}}$, where $\boldsymbol{\Theta}^{\text{new}}$ is the new parameter vector (HMM model) from the EM iteration. This process is repeated until convergence to a local maximum. A proper initialization of $\boldsymbol{\Theta}$ (with particular regard for $\boldsymbol{b}$) is crucial for a good convergence of the EM algorithm. For a Gaussian-observation model, applying the two-step iterations of the EM algorithm is equivalent to using Baum's re-estimation approach [29], which is as follows. Consider two new variables $\xi_t(i, j)$ and $\gamma_t(i)$, with $i, j \in \mathcal{S}$, that are defined as $\xi_t(i, j) = P(S_t = i, S_{t+1} = j|O_{1:T}, \boldsymbol{\Theta})$ and $\gamma_t(i) = \sum_{j=1}^{|\mathcal{S}|} \xi_t(i, j)$. We have:

$$\begin{aligned} \pi_i^{\text{new}} &= \gamma_1(i), \; a_{ij}^{\text{new}} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ \mu_i^{\text{new}} &= \frac{\sum_{t=1}^{T} \gamma_t(i) o_t}{\sum_{t=1}^{T} \gamma_t(i)}, \; \sigma_i^{2\,\text{new}} = \frac{\sum_{t=1}^{T} \gamma_t(i)(o_t - \mu_i)^2}{\sum_{t=1}^{T} \gamma_t(i)} \end{aligned} \tag{2}$$

where $\xi_t(i, j)$ and $\gamma_t(i)$ are computed using the Forward-Backward algorithm, see [30], [31].

We observe that the standard HMM is inadequate for our purpose. In fact, it uses a geometric Probability Mass Function (PMF) $g(d) = (a_{ii})^{d-1}(1 - a_{ii})$ to describe the dwell time of any hidden state $S_t = i \in \mathcal{S}$ with self-transition probability a_{ii}, i.e., $g(d)$ is the probability of staying in any hidden state $S_t = i \in \mathcal{S}$ for $d-1$ subsequent time steps and then leave the state (probability $(1 - a_{ii})$). It has been argued that this poorly models real phenomena, since most real-life applications do not obey this temporally-decaying function [32]. To tackle this, we consider the Extended Duration Hidden Markov Model (EDHMM), where for each hidden state $i \in \mathcal{S}$ we have $a_{ii} = 0$ and a state-specific distribution $p_i(d)$ is defined over the discrete set $\mathcal{D}_i = \{d_i^{\min}, \ldots, d_i^{\max}\}$, where $d_i^{\min}$ and $d_i^{\max}$ are the minimum and maximum durations for the protocol element transmitted when the EDHMM is in state i, respectively. Hence, upon entering state $i \in \mathcal{S}$, the sequence of observations in that state is assumed to be conditionally independent (i.e., i.i.d. once the state is entered), of length $d \in \mathcal{D}_i$ (sampled from $p_i(d)$), and is emitted from $P(O_t|S_t = i) = \mathcal{N}(O_t; \mu_i, \sigma_i^2)$. For the EDHMM, the duration distributions are collected into a vector $\boldsymbol{p}$, with $\boldsymbol{p} = [p_1(\cdot), \ldots, p_{|\mathcal{S}|}(\cdot)]$ and the EDHMM is described through the parameter vector $\boldsymbol{\Theta}_{\text{EDHMM}} = [\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{p}]$. In the following analysis, we use the HMM model to initialize the state duration distribution of the EDHMM (see Section VII for further details on the EDHMM training). Also, we use the forward-backward algorithm proposed by Yu and Kobayashi in [33], [34], as an alternative and efficient approach to solve Eq. (2).

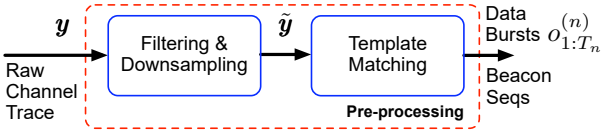

Fig. 4. Flow diagram of the mm-wave channel pre-processing phase.

V. High Level Description of the Framework

The aim of the mm-wave channel analyzer that we present in this paper is twofold. First, we want to track when data bursts are transmitted and, for each, detect which packets are exchanged, their duration, and average energy. This allows to obtain statistics on their number, duration, whether there are channel problems (which may be detected from missing ACKs). As a second objective, we track the transmission of control packets, which are sent for link management purposes. These control packets appear in two flavors as follows:

C1) Beacon pairs that mark the beginning of a data burst.

[1] Conversely, in the extended EDHMM, the entire process is not Markovian (memoryless). Instead the process is Markovian only at specified time instants.

[2] Specifically, one observation per state is assumed in the standard HMM model while in the EDHMM each state emits a sequence of observations. The length of the sequence while in state $i \in \mathcal{S}$ is determined by the length of time spent in state $i \in \mathcal{S}$, i.e., the duration d. Observations are assumed to be independent of time t, while in state $i \in \mathcal{S}$. Also, in the extended EDHMM, the transition probability a_{ij} is independent of the duration d of state $i \in \mathcal{S}$ and the duration d is only conditioned on the current state $j \in \mathcal{S}$.

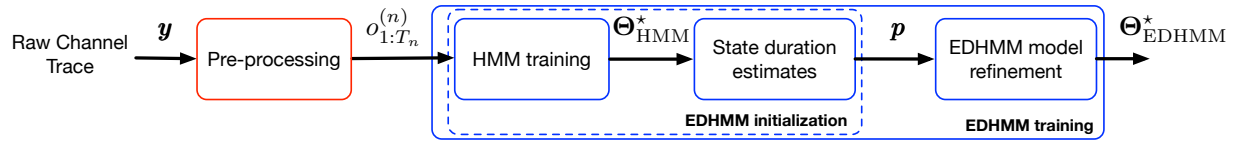

Fig. 5. EDHMM training procedure: initial state duration estimates are obtained through HMM training and are refined using EDHMM learning tools.

Fig. 6. EDHMM runtime mm-wave channel analysis.

C2) BR sequences that are utilized to maintain the radio link.

Our approach consists of three steps.

Step 1 – Pre-processing (Fig. 4): beacon detection and data burst extraction are implemented through the pre-processing chain of Fig. 4, which operates on the raw channel trace, through filtering, downsampling and template matching (see Section VI). We design the pre-processing chain for the case of 802.11ad but we can easily adapt it to suit other protocols. This pre-processing phase identifies all the beacons, classifies their occurrences into C1 and C2 and outputs a collection of N data bursts of the form $\{o^{(n)}_{1:T_n} | n = 1, \dots, N\}$, which are disjoint and contiguous channel subsequences.[3]

After Step 1, we delve into the semantic decoding of the protocol elements that are transmitted within each data burst, i.e., the elements in the above defined set $\mathcal{S}$. To assess which elements are transmitted, along with their average energy and timing, we utilize an EDHMM model, which is first trained (Fig. 5), and then used at runtime (Fig. 6) with non-stationary traces. Let $\boldsymbol{y} = (y_1, y_2, \dots)$ be a sequence of channel samples. In general, $\boldsymbol{y}$ can be written as $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{w}$ [35, Chapter 14], where $\boldsymbol{x} = (x_1, x_2, \dots)$ is the signal of interest *at the receiver*, that is, after transmission, and $\boldsymbol{w} = (w_1, w_2, \dots)$ is the background noise. From our experimental measurements, we know that $\boldsymbol{y}$ is highly non-stationary across data bursts, i.e., there are substantial variations in the energy associated with the signal $\boldsymbol{x}$ and the noise $\boldsymbol{w}$, which entail changes in μ_i and σ_i^2, for $i \in \mathcal{S}$. Moreover, they can also be caused by power control adjustments to compensate for channel attenuation and device mobility. Nevertheless, the transmission time of the elements in set $\mathcal{S}$ are channel and protocol-specific. We proceed through the following steps.

Step 2 – EDHMM training (Fig. 5): we use *stationary* channel traces[4] for a preliminary and robust training of the EDHMM parameters. Channel traces were picked so as to encompass a wide range of data rates and MCSs, which determine the different lengths of the physical layer data frames. The distance between transmitter and receiver is kept fixed and the surrounding environment (indoor for our experiments) is kept as stable as possible (i.e., no user mobility, etc.). From these stationary channels, the state-specific distributions $p_i(d)$ for $i \in \mathcal{S}$ do not undergo major changes during each trace and this allows their accurate estimation. Then, all the trace-specific distributions are combined into a global distribution considering a wide range of protocol settings, see Section VII. Note that training is needed only once for a given technology (e.g., IEEE 802.11ad).

Step 3 – Runtime trace analysis (Fig. 6): the EDHMM parameters μ_i and σ_i^2, for $i \in \mathcal{S}$ do depend on channel attenuation and noise. Thus, these parameters are estimated at runtime for each data burst using a clustering algorithm, whereas the $p_i(d)$ are known from Step 2. The so obtained EDHMM model is used to estimate the most likely sequence in $\mathcal{S}$ (called the Viterbi path) from the samples in the current data burst. This step is explained in Section VIII.

Steps 2 and 3 rely on the further assumption that:

A3) Channel attenuation and noise are stationary within bursts.

VI. Pre-processing

Data acquisition, filtering, and downsampling: to obtain the energy traces that we use as input for our machine learning algorithm, we overhear the communication of COTS 60 GHz devices using one or more external sniffers. Each sniffer consists of a Sivers IMA FC1005V/00 V-Band converter. The converter receives signals in the 60 GHz band either via a directional (20°) or omni-directional antenna and outputs them at 2 GHz intermediate frequency (IF). We capture the IF signal using a Universal Software Radio Peripheral (USRP) X310 Software Defined Radio (SDR) at a sample rate of 30 MHz. That is, we only need to capture a *fragment* of the bandwidth of the signal to obtain an energy trace which is suitable for our machine learning technique. To obtain a second trace from a different angle, we connect a second sniffer to the same USRP to ensure perfect time synchronization among traces. Since the coverage area of a mm-wave AP is limited due to high path loss, sniffers are typically close to each other and can thus be connected to the same USRP. Moreover, if traces are recorded on different USRPs, it is possible to synchronize them in post-processing using a variant of template matching, see Fig. 7, where a subsequence from SN_2 is used as a template. In Tab. I, we report the average synchronization error as a function of the template length τ. To obtain these results, we have run $1,000$ simulations for each value of τ

[3] A contiguous subsequence is made up of consecutive channel samples.

[4] These *stationary* channel traces do not exhibit any particular trend. This means that μ_i and σ_i^2 do not significantly vary across data bursts.

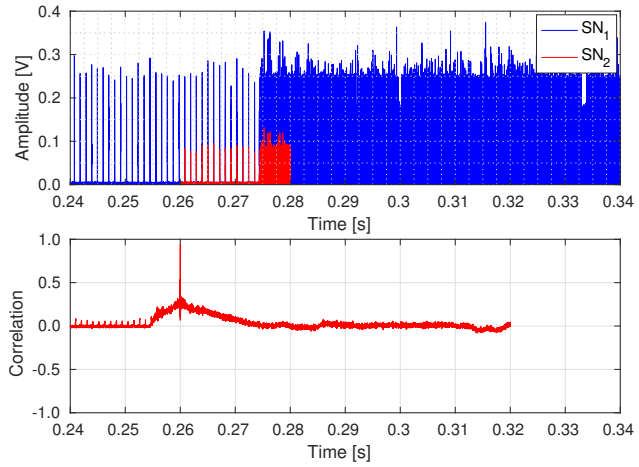

Fig. 7. Synchronization using a variant of template matching, where a subsequence from SN_2 is used as a template.

TABLE I
AVERAGE SYNCHRONIZATION ERROR VERSUS TEMPLATE LENGTH τ.

$\tau = 1$ ms	$\tau = 2$ ms	$\tau = 3$ ms	$\tau = 4$ ms	$\tau \geq 5$ ms
36.36 ms	13.43 ms	2.63 ms	1.17 ms	0

picking a random subsequence from SN_1 and a subsequence from SN_2 (used as a template) with random temporal offset with respect to the subsequence from SN_1. We obtain perfect synchronization by choosing $\tau \geq 5$ ms. Synchronizing traces from multiple sniffers is thus doable and only requires picking a sufficiently long template length.

Fig. 8 shows our measurement setup. The original raw trace $\boldsymbol{y}$ is first filtered and then downsampled to a lower rate for scaling purposes, so that each sample of the new trace is computed as the mean of three subsequent samples in the original raw trace. This new trace is then smoothed using a fast and robust discretized spline filtering algorithm for data of large size [36] [37], thus obtaining the trace $\tilde{\boldsymbol{y}}$. This pre-processing phase is needed to remove part of the noise due to hardware impairments during data acquisition. It does not harm the EDHMM classification performance, but generally improves it, as the noise variance in the energy traces is reduced.

Template matching algorithm: after the data acquisition, filtering, and downsampling, a collection of N data bursts of the form $\{o_{1:T_n}^{(n)} | n = 1, \ldots, N\}$ is extracted from the mm-wave trace $\tilde{\boldsymbol{y}}$. This requires a reliable identification technique for the data bursts and, recalling that each data burst is preceded by a pair of beacons, this corresponds to reliably detecting beacon pairs. What we observe from the collected channel traces is that the beacon duration and the inter-frame spacing between them are almost constant within and across experiments. Moreover, we note that the beacon shape is quite particular, showing different energy levels at the beginning and at the end. These characteristics make it possible to exploit a template matching technique for the beacon detection. Here, we are interested in finding C1) beacon pairs, and C2) BR sequences, as these are key to understand the protocol behavior.

At the core of our template matching approach, we use Pearson's correlation coefficient $r \in [-1, 1]$ [38], which is

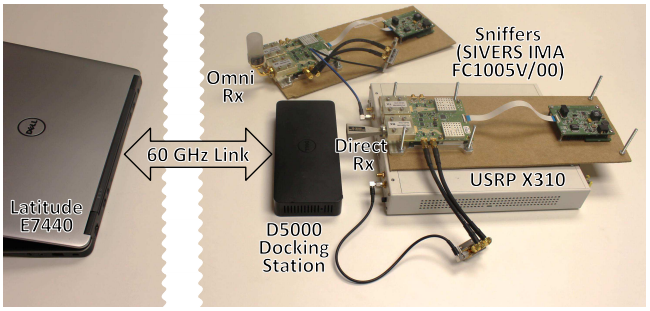

Fig. 8. Practical sniffer setup for trace capture. The antennas at the sniffers can be both directional or omni-directional, and sniffer location can be varied.

a statistical measure of the strength of a linear relationship between two vectors $\boldsymbol{u} = [u_1, \ldots, u_K]$ and $\boldsymbol{v} = [v_1, \ldots, v_K]$ (with mean μ_u and μ_v, respectively). It is defined as the ratio of their covariance C_{uv} and the square root of the product of their variances σ_u^2 and σ_v^2, i.e., $r = C_{uv}/(\sigma_u \sigma_v)$, where C_{uv} is the sample covariance, given by:

$$C_{uv} = \frac{1}{K-1} \sum_{k=1}^{K} (u_k - \mu_u)(v_k - \mu_v) \,. \tag{3}$$

Pearson's correlation coefficient is suitable to deal with the non-stationarity of the traces, since it just evaluates some internal relationship between the provided vectors. Moreover, template matching is known to be the optimal detection technique in the presence of white Gaussian noise [39], which we found to be a good assumption for our mm-wave channel traces [27]. Henceforth, for our template matching technique, $\boldsymbol{u}$ corresponds to the average shape of a beacon frame (i.e., the template with a length of K samples), which the system can easily obtain from channel idle times. During those idle times, nodes only transmit periodic beacons which can be clearly identified and used as a template. Vector $\boldsymbol{v}$ contains the channel samples from the current K-dimensional sliding window, which moves over the signal trace $\tilde{\boldsymbol{y}}$, obtained after the acquisition, filtering, and downsampling of $\boldsymbol{y}$. We adopted the fast template matching scheme of [40] [41], which exploits the Fast Fourier Transform (FFT), thus obtaining dot products in the frequency domain. For a generic channel sequence $\tilde{\boldsymbol{y}}$ of $L > K$ samples, this allows the computation of the covariance in $O(L \log L)$ time. Hence, the template matching operates on $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \ldots, \tilde{y}_L)$, outputting a sequence of correlation estimates $(r_1, \ldots, r_{L-K+1})$. We detect a possible beacon at sample ℓ if r_ℓ is greater than a threshold r_{th}. Then, since multiple trivial matches (i.e., $r_\ell > r_{\text{th}}$) are likely to occur within a window of samples, we perform a further peak detection within the regions containing multiple matches, by taking the default timing parameters of the IEEE 802.11ad communication standard into account [42]. That is, two beacons can never be placed at a distance smaller than the minimum allowed by the protocol rules. As the final step, we assess which beacon pairs actually mark the start of data bursts by assessing the distance between them, as this is constant. Through this, we can reliably detect false positives, such as isolated beacons due to communication errors or to packets that are erroneously detected as beacons as their shape closely resembles that of the template. We found excellent results

across all our experiments setting $r_{\rm th} = 0.75$. Note that $r_{\rm th}$ is independent of the trace amplitude. Thus, we do not need to readjust it for each scenario and/or trace.

The identification of pairs of beacons (C1) allows extracting the data bursts $\{o_{1:T_n}^{(n)} | n = 1, \ldots, N\}$ from $\tilde{\boldsymbol{y}}$, which are fed as input to the following EDHMM training phase. Longer beacon sequences (C2) are likewise detected by looking at the number of energy levels of the beacons therein and at their inter-frame spacing, as dictated by the standard [42]. These events are semantically decoded as described below.

VII. EDHMM TRAINING

For the EDHMM training we refer to Fig. 5. We recall that the objective of this training phase is to reliably estimate the distribution vector $\boldsymbol{p}$, modeling the duration of inter-frame spaces, packets and acknowledgements. This phase is executed once offline and is not scenario dependent. Essentially, it is a calibration step for the specific mm-wave technology used in the network, which in our case is IEEE 802.11ad. The traces used in this step should be as much as possible stationary. This means that μ_i and σ_i^2 do not significantly vary across data bursts. As a first processing stage, we use the pre-processing procedure of Section VI, which returns the data burst set $\{o_{1:T_n}^{(n)} | n = 1, \ldots, N\}$. Next, for illustration purposes we refer to the n-th data burst $o_{1:T_n}^{(n)} = (o_1, \ldots, o_{T_n})$, but in our implementation the HMM parameters are estimated using the entire burst set (i.e., the N bursts in the mm-wave trace). For burst n, each of the samples o_t, $t = 0, \ldots, T_n$, maps to an element $s_t \in \mathcal{S}$, where state "1" means IFS, "2" DATA and "3" ACK. Our goal is to accurately associate each o_t in the data burst with the actual protocol element $i \in \mathcal{S}$ and, most importantly, to reliably estimate its duration PMF $p_i(\cdot)$. This estimation is performed having access to the noisy observations $(o_1, \ldots, o_{T_n})$ of the actual protocol elements.

EDHMM initialization: we consider $o_{1:T_n}^{(n)}$ as training data and our aim is to get accurate state duration estimates for the EDHMM. This is achieved by deriving initial estimates for $\boldsymbol{p}$ through a simpler HMM model. Once this vector is found, it is refined using EDHMM training tools. The HMM parameter vector is $\boldsymbol{\Theta}_{\rm HMM}$ and the three fundamental steps involved in the HMM model estimation are:

E1) The forward-backward algorithm is used to compute metrics $\gamma_t(i)$ and $\xi_t(i,j)$ with $t = 1, \ldots, T_n$, $i, j \in \mathcal{S}$ (see Eq. (2)) for a given HMM transition structure and a list of observations. These weigh the probability of getting the observed sequence from the current model.

E2) The model parameter vector $\boldsymbol{\Theta}_{\rm HMM}$ is adjusted through the EM algorithm.

E3) The Viterbi algorithm [43] is used to compute the most probable path via a Maximum Likelihood (ML) approach.

Step E2 returns the optimal parameter vector $\boldsymbol{\Theta}_{\rm HMM}^{\star}$, whereas E3 outputs the sequence of hidden states $(s_1, \ldots, s_{T_n})$ that most likely generated the observed samples $(o_1, \ldots, o_{T_n})$.

Specifically, we assume $\pi_1 = 1$ as all the data bursts start with a silence, right after the beacon pair. Moreover, the HMM transition matrix $\boldsymbol{A}$ is constrained in the sense that the hidden state sequence evolves according to structured trajectories [44]. In particular, we have $a_{23} = a_{32} = 0$, as there must be some minimum inter-frame spacing between subsequent messages. Also, we set $a_{ii} = 1 - 1/T_s$ for $i \in \mathcal{S}$, where $T_s = 0.1$ μs is the channel sampling period after the downsampling of Section VI. The initialization implies geometrically distributed state dwell-time distributions. This serves as a sufficiently good initialization of the transition matrix, and increases the robustness of the HMM model against random fluctuations in the channel dynamics. Next, we use the Viterbi algorithm output (step E3) to initialize the state duration distribution of the EDHMM model.

The final parameter estimates $\boldsymbol{\Theta}_{\rm HMM}^{\star}$ strongly depend on the initial vector $\boldsymbol{\Theta}_{\rm HMM}^{0}$ for the EM evaluation (step E2). To obtain good initial parameter estimates, we use the K-means clustering algorithm, see [45], [46], which classifies the channel samples in the data bursts around $|\mathcal{S}|$ centers. The $|\mathcal{S}|$ initial values of the centers can be randomly picked or taken as the locations of the peaks in the empirical distribution of the observed samples. The latter approach was implemented and found to perform satisfactorily across all datasets. Upon completion, the K-means algorithm returns $|\mathcal{S}|$ values for the cluster centers, which are used as initial values for μ_i for $i \in \mathcal{S}$. The $|S|$ variances σ_i^2 are derived from the distribution of the samples clustered around the centers μ_i so obtained.

At this point, we use the Viterbi algorithm output (step E3) to fit $|\mathcal{S}|$ two-parameter inverse Gaussian distributions [47] [48] for vector $\boldsymbol{p}$, where the range $\mathcal{D}_i = \{d_i^{\min}, \ldots, d_i^{\max}\}$ for state $i \in \mathcal{S}$ is such that $d_1^{\min} = \cdots = d_{|\mathcal{S}|}^{\min} = 1$ and $d_1^{\max} = \cdots = d_{|\mathcal{S}|}^{\max} = D$. In particular, we set D according to the timing parameters of the IEEE 802.11ad communication standard [42] and filter out all the state durations that are outside these boundaries. The authors in [48] show how to find maximum likelihood solutions for the parameters of any family of exponential distributions. Since the exponential family is log-concave, the global maximum can be found by setting derivatives equal to zero, yielding the maximum likelihood equations. For some distributions in the exponential family (e.g., Gaussian), these equations can be solved analytically, while most distributions must be solved numerically. In the present work, two-parameter inverse Gaussian distributions have been preferred over non-parametric state duration distributions, as parametric models require far less training data and generalize better to new data. Note also that, even if we expect a fixed set of timing parameters for the IEEE 802.11ad communication standard [42],[5] it is still possible that the duration of the protocol frames in the training data differs from that in the test data, due to MCS adjustments. Thus, a rigid setting that only allows for a few fixed values, although correct in theory, may lead to unsatisfactory results in real cases, overfitting the training data and generalizing poorly over the test cases.

EDHMM model refinement: the initial estimate $\boldsymbol{p}$ that we have found with the above HMM model is subsequently refined through EDHMM training tools. Here, we opted for the forward-backward algorithm proposed by Yu and Kobayashi

[5]That is, only a few fixed durations for ACK, DATA and silence are possible, as DATA depend on the adopted modulation and coding scheme.

in [33], [34] as it is efficient and solves practical issues such as numerical underflows occurring in the EM iterations.

VIII. Runtime trace analysis

In this section, we present a runtime analyzer that effectively deals with the non-stationarity of the traces, i.e., variations in μ_i and σ_i^2 for $i \in \mathcal{S}$ across data bursts. As a first step, we run the pre-processing block of Section VI, which returns the data burst sequences $\{o_{1:T_n}^{(n)} | n = 1, 2, \dots\}$ through template matching. For each sequence, the energy levels associated with the states IFS, DATA and ACK are re-estimated, as explained in the following.

Gaussian-observation model update: we rely on assumption A3, i.e., that channel statistics are stationary within each data burst. Of course, μ_i and σ_i^2 may change considerably across data bursts and we tackle this by running the K-means clustering algorithm for each burst sequence $o_{1:T_n}^{(n)}$, so as to re-initialize vector $\boldsymbol{b}$ in an online fashion. The $|\mathcal{S}|$ final values of the centers initialize μ_i, whereas the variances of the samples clustered around these centers initialize σ_i^2, for $i \in \mathcal{S}$. Upon completing the K-means algorithm, we obtain the updated parameter set $\boldsymbol{\Theta}_{\text{EDHMM}}^{(n)}$ for the current data burst, whereas vector $\boldsymbol{p}$ (which represents the "average" time-frame duration statistics) remains fixed. We remark that, for the current burst n, $\boldsymbol{\Theta}_{\text{EDHMM}}^{(n)}$ may differ from the optimal parameter set $\boldsymbol{\Theta}_{\text{EDHMM}}^{\star}$, as for the latter vectors $\boldsymbol{p}$ and $\boldsymbol{b}$ would be obtained through the ML approach of [33], [34], whereas in $\boldsymbol{\Theta}_{\text{EDHMM}}^{(n)}$ the energy levels in $\boldsymbol{b}$ are estimated on-the-fly through K-means. Since the latter approach does not take into account the joint re-estimation of $\boldsymbol{p}$ and $\boldsymbol{b}$, the resulting energy levels are less accurate. However, this approach provides a substantial speedup as neither the re-estimation of the transition matrix $\boldsymbol{A}$ nor that of vector $\boldsymbol{p}$ are required and these computations account for most of the EDHMM complexity. Hence, the benefit due to the increased speed outweighs the loss in accuracy.

Online estimation via time-adaptive EDHMM: upon obtaining $\boldsymbol{\Theta}_{\text{EDHMM}}^{(n)}$ for the current data burst $o_{1:T_n}^{(n)}$, the corresponding hidden state sequence $(s_1, \dots, s_{T_n})$ is reconstructed using the Viterbi algorithm with samples $o_{1:T_n}^{(n)} = (o_1, \dots, o_{T_n})$, i.e., each sample o_t is mapped onto one of the elements in $\mathcal{S}$. As suggested in [32], we implemented the Viterbi algorithm using logarithms to avoid numerical underflows. Also, given the sequence of observations $o_{1:T_n}^{(n)}$, the time complexity of the Viterbi algorithm for EDHMM is $O(|\mathcal{S}| Z T_n D)$, where Z is the average number of predecessors for each state $i \in \mathcal{S}$. In our case, $Z < |\mathcal{S}|$, since we set $a_{23} = a_{32} = 0$ (states 2 and 3 respectively denote DATA and ACK). Moreover, since durations are explicitly accounted through $p_i(\cdot)$, we have $a_{ii} = 0$, $\forall\, i \in \mathcal{S}$ and $Z = 4/3$. Hence, the computational cost of the Viterbi algorithm is primarily affected by the data burst length T_n and by the maximum duration D.

IX. Generalization to multiple dimensions

Up to this point, we have considered that the observed channel samples in the data burst are modeled as a sequence of real-valued random variables corresponding to one of the following basic elements: "1" IFS, "2" DATA and "3" ACK. Accordingly, the hidden state S_t at time t is a discrete real-valued random variable that can take values in the set $\mathcal{S} = \{1, 2, 3\}$. This corresponds to capturing the channel from a single measurement point.

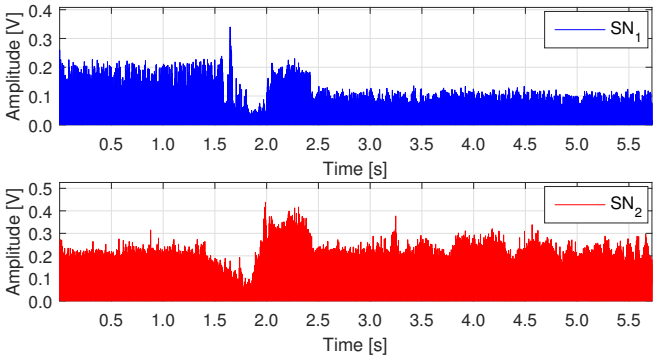

Fig. 9. Communication recorded from two different locations SN_1 and SN_2.

In this section, we are concerned with the case where multiple channel traces from the same source are concurrently monitored from different measurement points. This amounts to deploying multiple time-synchronized receivers (sniffers) and have them listening to the same transmitter. The rationale is that the channel realizations that they respectively see are likely to be uncorrelated. This can provide significant improvements to the detection performance. In Fig. 9, we show the same transmission captured by two different sniffers, labeled SN_1 and SN_2.

In this scenario, the observed channel samples in the data burst are modeled as a sequence of multi-dimensional, real-valued random variables, $\boldsymbol{O}_{1:T} = (\boldsymbol{O}_1, \dots, \boldsymbol{O}_T)$, where T is the data burst length. The observation vector associated with channel sample t, $\boldsymbol{O}_t$, is a probabilistic function of the hidden state S_t. According to the Gaussian-observation model, $P(\boldsymbol{O}_t | S_t = i) = \mathcal{N}(\boldsymbol{O}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ respectively specify the multi-dimensional mean and the diagonal covariance matrix of the random vector $\boldsymbol{O}_t$, given that the hidden state is $i \in \mathcal{S}$. For all hidden states $i \in \mathcal{S}$, we collect the parameter pairs $b_i = (\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ through vector $\boldsymbol{b} = [b_1, \dots, b_{|\mathcal{S}|}]$, as for the one-dimensional case. For Baum's re-estimation approach [29], we have:

$$\begin{aligned} \boldsymbol{\mu}_i^{\text{new}} &= \frac{\sum_{t=1}^{T} \gamma_t(i) \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t(i)} \\ \boldsymbol{\Sigma}_i^{\text{new}} &= \frac{\sum_{t=1}^{T} \gamma_t(i) (\boldsymbol{o}_t - \boldsymbol{\mu}_i)^{\text{T}} (\boldsymbol{o}_t - \boldsymbol{\mu}_i)}{\sum_{t=1}^{T} \gamma_t(i)}, \end{aligned} \tag{4}$$

where $\boldsymbol{o}_t$ is the realization of $\boldsymbol{O}_t$, $\xi_t(i, j)$ and $\gamma_t(i)$ are computed using the Forward-Backward algorithm, see [30], [31], and $(\cdot)^{\text{T}}$ denotes vector transpose. Here, we assume that there is no correlation among the observed channel samples from multiple viewpoints, hence $\boldsymbol{\Sigma}_i$ is a diagonal covariance matrix for $i \in \mathcal{S}$. In the following, the sniffers are denoted by SN_{dim}, where $\text{dim} = 1, 2, \dots$. Moreover, when the hidden state is $i \in \mathcal{S}$, we refer to the element in position dim in $\boldsymbol{\mu}_i$ as $\mu_i^{(\text{dim})}$ and to the dim-th diagonal element of $\boldsymbol{\Sigma}_i$ as $\Sigma_i^{(\text{dim})}$.

X. MM-WAVE TRACE GENERATOR

Need for the ground truth: a ground truth signal is necessary to precisely quantify the reconstruction performance of the one- and the multi-dimensional protocol analyzers. This would amount to acquire the actual protocol state that is associated with each sample in $\boldsymbol{O}_{1:T}$. In practice, this could be achieved using a tool such as Wireshark on a monitor node. Unfortunately, state-of-the-art 802.11ad hardware is still unable to reliably provide such information to the higher layers. The protocol state sequence that is extracted by the radio is usually incomplete (some frames are missing), the states are shifted in time (with distorted inter-spaces) and often their order is also affected. The only metric that current devices can reliably provide are cumulative counters of packet types (see Section XI-A for further discussion and experimental results).

To overcome this, we have developed a realistic mm-wave trace generator, with the goal of reproducing *narrowband* physical layer energy traces from one or multiple sniffers in a fast and accurate manner. This makes the evaluation of our diagnosis tool possible, providing quantitative results in a range of practical scenarios. The developed generator reproduces typical IEEE 802.11ad data burst sequences, mimicking random fluctuations in the channel dynamics, variability in the number and duration of DATA and ACK frames, etc. These burst sequences are separated by beacon pairs (also affected by channel noise), whereas other control messages, appearing outside the data bursts, are not modeled as they are not involved in our performance assessment.

This tool has been instrumental in the fine tuning of the EDHMM model, as it allows for a precise control of the energy levels associated with transmissions from the source and channel noise. Next, we detail its structure, which is organized into macro- and micro-states. Macro-states describe different instances of the channel transmission setup, i.e., a specific combination of coding and modulation schemes, and we assume that macro-state transitions can only occur at the end of data bursts. Instead, micro-states track the duration of DATA and ACK frames within a data burst, for any given setup (macro-state). Each data burst starts with a beacon pair and the system remains in the same macro-state for the entire duration of the burst. Once in a data burst, the micro-state model returns the sequence of DATA and ACK frames.

Notation: the superscript $^{(0)}$ is used for the macro-model parameters. The macro-state takes values in the set $\mathcal{M}^{(0)} = \{1, \ldots, |\mathcal{M}^{(0)}|\}$ and evolves according to the transition matrix $\boldsymbol{T}^{(0)}$, with steady state distribution $\boldsymbol{\pi}^{(0)}$. If the current macro-state is $M \in \mathcal{M}^{(0)}$, the micro-state m takes values in the set $\mathcal{M}^{(M)} = \{1, \ldots, |\mathcal{M}^{(M)}|\}$ and evolves according to the transition matrix $\boldsymbol{T}^{(M)}$, with steady state distribution $\boldsymbol{\pi}^{(M)}$.

The macro-model: macro-states capture different instances of the channel transmission setup, i.e., $|\mathcal{M}^{(0)}|$ different combinations of coding and modulation schemes. For each macro-state $M \in \mathcal{M}^{(0)}$, the following statistics are specified: (i) the PMF $P(d^{(M)}_{\mathrm{IDLE}})$ of the idle time $d^{(M)}_{\mathrm{IDLE}}$ between subsequent packets, (ii) the joint PMF $P(d^{(M)}_{\mathrm{DATA}}, d^{(M)}_{\mathrm{ACK}})$ of DATA and ACK durations, respectively termed $d^{(M)}_{\mathrm{DATA}}$ and $d^{(M)}_{\mathrm{ACK}}$, where the ACK is the frame following the DATA one in the hidden state sequence $(s_1, \ldots, s_{T_n})$, and (iii) the PMF $P(d^{(M)}_{\mathrm{burst}})$ of the duration of data bursts, $d^{(M)}_{\mathrm{burst}}$.

The following remarks are in order:

- We have experimentally verified that $P(d^{(M)}_{\mathrm{DATA}}, d^{(M)}_{\mathrm{ACK}}) \neq P(d^{(M)}_{\mathrm{DATA}})P(d^{(M)}_{\mathrm{ACK}})$, which means that there exists some correlation between the marginal random variables modeling DATA and ACK durations. This is due to frame aggregation, which results in block acknowledgments. Such acknowledgments are longer than the regular acknowledgments used for shorter, non-aggregated frames.
- Stationary channel traces are utilized to obtain the duration statistics of idle times, DATA packets and ACKs. As done for the online estimation via time-adaptive EDHMM, upon obtaining $\boldsymbol{\Theta}^{(n)}_{\mathrm{EDHMM}}$ for the current data burst $o^{(n)}_{1:T_n}$, the corresponding hidden state sequence $(s_1, \ldots, s_{T_n})$ is reconstructed using the Viterbi algorithm with samples $o^{(n)}_{1:T_n} = (o_1, \ldots, o_{T_n})$. From these estimates, duration statistics for the elements in the set $\mathcal{S} = \{1, 2, 3\}$ and for the data burst itself are obtained.
- Transitions between macro-states occur at the end of each data burst according to the transition matrix $\boldsymbol{T}^{(0)}$. This makes it possible to probabilistically model variations in the protocol behavior due to, e.g., soft link blockages (e.g., waiving a hand in the boresight of the antenna) or to modifications of the received energy due to a change in the orientation of the device.

The micro-model: consider a data burst in any macro-state M. Within this data burst, a sequence of DATA-ACK frames is exchanged, and the duration of such packets is controlled by the micro-model. Specifically, the domain of the PMF $P(d^{(M)}_{\mathrm{DATA}}, d^{(M)}_{\mathrm{ACK}})$ is clustered into $|\mathcal{M}^{(M)}|$ rectangular subdomains through the Elbow method, which is a clustering technique to automatically determine the number of clusters K [49]. We run this clustering algorithm twice: for the dimension associated with $\{d^{(M)}_{\mathrm{DATA}}\}$ and for that associated with $\{d^{(M)}_{\mathrm{ACK}}\}$, obtaining $K^{(M)}_{\mathrm{DATA}}$ and $K^{(M)}_{\mathrm{ACK}}$ clusters for DATA and ACK frames, respectively. This leads to $|\mathcal{M}^{(M)}|$ rectangular subdomains with $|\mathcal{M}^{(M)}| = K^{(M)}_{\mathrm{DATA}} \times K^{(M)}_{\mathrm{ACK}}$. Each of such domains $m \in \mathcal{M}^{(M)}$ defines a micro-state with conditional PMF $P(d^{(M)}_{\mathrm{DATA}}, d^{(M)}_{\mathrm{ACK}}|m)$, representing the joint distribution of DATA and ACK durations within that region (conditioned on the model being in region m). Transitions between micro-states occur according to the transition matrix $\boldsymbol{T}^{(M)}$, which is estimated from empirical data. The steady-state probability vector $\boldsymbol{\pi}^{(M)}$ is obtained through numerical integration of $P(d^{(M)}_{\mathrm{DATA}}, d^{(M)}_{\mathrm{ACK}})$ within region m, for all $m \in \mathcal{M}^{(M)}$.

Outline of the algorithm: the pseudo-code of the mm-wave trace generator is given in Algorithm 1. The algorithm's output consists of sequences of data bursts, delimited by beacon pairs. The algorithm starts by picking macro- and micro-states according to the respective steady-state distributions (see function `pick()` in lines 1 and 2). $\tilde{\boldsymbol{y}}^{(\mathrm{dim})}$ is the noisy output sequence, which is initialized as an empty vector (line 3). In line 5, the data burst duration T is sampled from the PMF $P(d^{(M)}_{\mathrm{burst}})$ and time-synchronized burst sequences are generated for two sniffers SN_1 and SN_2 ($\mathrm{dim} = 1$ and 2),

Algorithm 1 Pseudo-code of the mm-wave trace generator

1: $M = \texttt{pick}(\boldsymbol{\pi}^{(0)})$; // pick macro-state
2: $m = \texttt{pick}(\boldsymbol{\pi}^{(M)})$; // pick micro-state
3: Set $\ell = 0$; $\tilde{\boldsymbol{y}}^{(\dim)} = \texttt{empty\_vector}()$;
4: **while** $\ell < L$ **do**
5: $\quad T = \texttt{pick}(P(d^{(M)}_{\text{burst}}))$;
6: $\quad$ **for** dim $= 1$ to 2 **do**
7: $\quad\quad \tilde{\boldsymbol{y}}^{(\dim)} = \texttt{concatenate}(\tilde{\boldsymbol{y}}^{(\dim)}, \boldsymbol{tmp}^{(\dim)})$;
8: $\quad\quad$ Set $t = 0$; $\boldsymbol{o}^{(\dim)} = \texttt{empty\_vector}()$;
9: $\quad\quad$ **while** $t < T$ **do**
10: $\quad\quad\quad d_1 = \texttt{pick}(P(d^{(M)}_{IDLE}))$;
11: $\quad\quad\quad (d_2, d_3) = \texttt{pick}(P(d^{(M)}_{\text{DATA}}, d^{(M)}_{\text{ACK}})|m)$;
12: $\quad\quad\quad \boldsymbol{o}' = \texttt{create\_frames}(d_1, d_2, d_3)$;
13: $\quad\quad\quad \boldsymbol{o}^{(\dim)} = \texttt{concatenate}(\boldsymbol{o}^{(\dim)}, \boldsymbol{o}')$;
14: $\quad\quad\quad$ // Change current micro-state?
15: $\quad\quad\quad m = \texttt{next\_state}(\boldsymbol{T}^{(M)}, m)$;
16: $\quad\quad\quad t = \texttt{length}(\boldsymbol{o}^{(\dim)})$;
17: $\quad\quad$ **end while**
18: $\quad\quad \tilde{\boldsymbol{y}}^{(\dim)} = \texttt{concatenate}(\tilde{\boldsymbol{y}}^{(\dim)}, \boldsymbol{o}^{(\dim)})$;
19: $\quad$ **end for**
20: $\quad M_{\text{prev}} = M$;
21: $\quad$ // Change current macro-state?
22: $\quad M = \texttt{next\_state}(\boldsymbol{T}^{(0)}, M)$;
23: $\quad$ **if** $M \neq M_{\text{prev}}$ **then**
24: $\quad\quad$ // resample from steady-state distribution
25: $\quad\quad m = \texttt{pick}(\boldsymbol{\pi}^{(M)})$;
26: $\quad$ **end if**
27: $\quad \ell = \texttt{length}(\tilde{\boldsymbol{y}}^{(\dim)})$;
28: **end while**

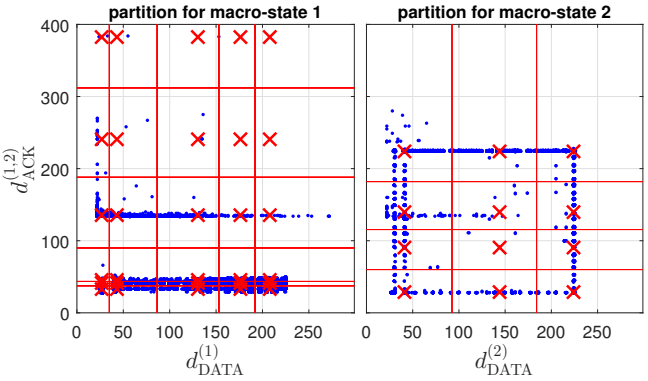

Fig. 10. Empirical measurements $(d^{(M)}_{\text{DATA}}, d^{(M)}_{\text{ACK}})$ for two marco-states. Rectangular regions are obtained using the Elbow method. Values in the axes are expressed in number of channel samples (the sampling frequency is $T_s = 0.1$ μs).

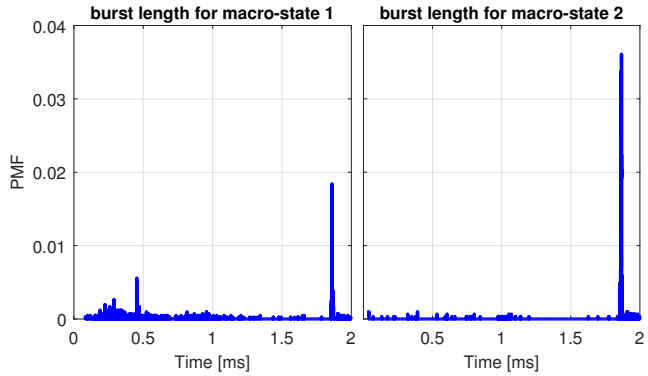

Fig. 11. PMF of the burst length $P(d^{(M)}_{\text{burst}})$ for macro-models 1 and 2.

see line 6. Each burst starts with a beacon pair, denoted by $\boldsymbol{tmp}^{(\dim)}$, which is a noisy version of the template used by the template matching algorithm of Section VI. The beacons are concatenated to the output sequence $\tilde{\boldsymbol{y}}^{(\dim)}$ in line 7, and the noisy data burst is created through the "while" cycle starting from line 9. Durations of DATA (d_2), ACK (d_3) frames and of the IDLE time between them (d_1) are respectively sampled from the PMFs $P(d^{(M)}_{\text{DATA}}, d^{(M)}_{\text{ACK}}|m)$ and $P(d^{(M)}_{\text{IDLE}})$. A noisy sequence composed of DATA (d_2 samples), IDLE (d_1 samples), ACK (d_3 samples) and IDLE (d_1 samples) is generated through the "$\texttt{create\_frames}()$" function of line 12. The so obtained output samples $\boldsymbol{o}'$ (of length $2d_1 + d_2 + d_3$ samples) is then appended to the current sequence $\boldsymbol{o}^{(\dim)}$ (line 13). When the while cycle ends, $\boldsymbol{o}^{(\dim)}$ contains the noisy samples associated with the new data burst. The noisy samples in the sequence, which are associated with hidden state $i \in \{1, 2, 3\}$ (respectively IDLE, DATA and ACK), are computed as (additive Gaussian noise): $\mu_i^{(\dim)} + \sqrt{\Sigma_i^{(\dim)}}\texttt{randn}(1, d_i)$, where "$\texttt{randn}(1, d_i)$" denotes a random vector of d_i elements, with Gaussian distributed entries $\mathcal{N}(0, 1)$. Although not explicitly indicated, the sequence of hidden states is saved along with the noisy version $\tilde{\boldsymbol{y}}^{(\dim)}$ and used as ground truth for the performance evaluation of Section XI-B.

We now discuss some example results for the case of two macro-states. $M = 1$: distance TX-RX 1.5 m, MCS 11. $M = 2$: distance TX-RX 2.5 m, MCS 10. Empirical $(d^{(M)}_{\text{DATA}}, d^{(M)}_{\text{ACK}})$ pairs are shown in Fig. 10 for $M \in \{1, 2\}$, along with the rectangular regions obtained using the Elbow clustering algorithm. We observe that the duration statistics are more spread in macro-model 1 with respect to macro-model 2, meaning less data aggregation. Also, the length of the data burst is shorter in macro-model 1, see Fig. 11. For each macro-state M, the domain $(d^{(M)}_{\text{DATA}}, d^{(M)}_{\text{ACK}})$ is split into clusters using the Elbow method: the number of clusters is greater in macro-model 1, i.e., $K^{(1)}_{\text{DATA}} = K^{(1)}_{\text{ACK}} = 6$ and $K^{(2)}_{\text{DATA}} = K^{(2)}_{\text{ACK}} = 3$.[6] Fig. 12 shows empirical points $(\mu_i^{(\dim)}, \Sigma_i^{(\dim)})$ for different channel setups. In this plot, we do not distinguish between states $i \in \{1, 2, 3\}$, as our purpose is to establish a suitable relation between mean (μ) and variance (Σ) of the received energy levels, and the difference in the received energy levels captured by the sniffers depends on the relative position of the sniffers with respect to the communicating devices. These empirical points were fitted through the following curve (the red solid curve in the plot):

$$\Sigma_i = c_1 \mu_i^{c_2}\,,\; i \in \{1, 2, 3\}\,, \tag{5}$$

with $c_1 = 0.105$ and $c_2 = 1.905$. The coefficients c_1 and c_2 were found through a linear regression in the logarithmic domain, i.e., we fit the dataset taking into account the

[6]The number of clusters is chosen such that the percentage of variance explained by them is greater than 90%, where this percentage represents the ratio between the group variance and the total variance.

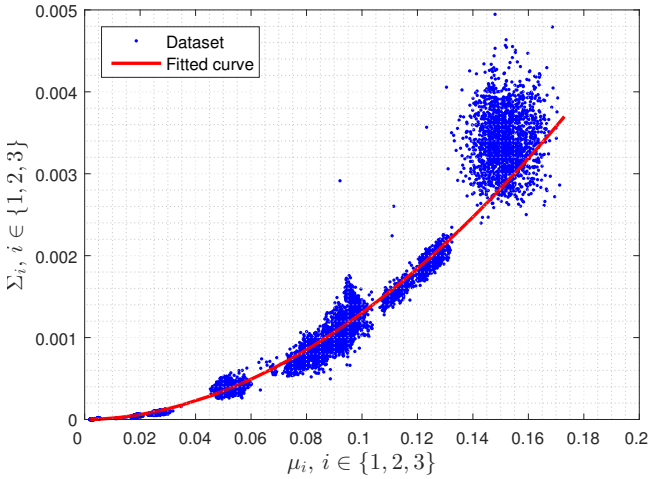

Fig. 12. Gaussian observation model: empirical values and fitting curve for mean (μ) and variance (Σ) of the received energy levels associated with IDLE periods, DATA and ACK frames.

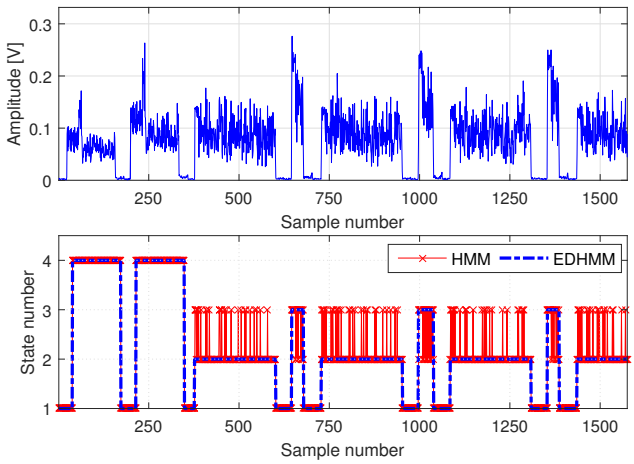

Fig. 13. Trace decoding example for our machine learning framework.

logarithmic counterpart of the datapoints and minimize the total residual error, obtaining an excellent goodness of fit ($R^2 = 0.9595$, where R^2 is the coefficient of determination). The linear relationship in the logarithmic domain of Eq. (5), that we obtained empirically, is also confirmed by previous analytical work on RSS localization, see, e.g., [50], [51].

As a final consideration, we note that the average energy levels μ_i and their variances Σ_i remain constant for the entire duration of the data bursts, which is a key assumption in the developed EDHMM algorithm (see assumption A3, in Section V). In the numerical results, we assess the performance of our algorithms when assumption A3 is no longer verified, i.e., when μ_i and Σ_i do change within a DATA burst. This is achieved through an *additive* sinusoidal noise of frequency f, which is added to the generated energy traces, by tuning f and the noise amplitude.

XI. Performance Results

Next, we present some selected performance results. Experimental results are discussed in Section XI-A, considering single and multiple time-synchronized sniffers. The performance of our tool is further quantified in Section XI-B, using the mm-wave trace generator of Section X.

A. Evaluation with experimental data

We consider the setup in Section VI. First, we validate our machine learning framework in controlled scenarios. Next, we study the behavior of indoor links during regular operation to check how our framework can identify and characterize effects such as beam misalignment.

Validation in controlled scenarios: Fig. 13 shows a trace decoding example for our diagnosis tool. In the upper part of the figure, we show the raw trace as captured by the Sivers IMA converter. The two initial frames are beacons that indicate the start of a data burst. After that, we observe a sequence of data and acknowledgment frames (c.f. Fig. 1). The lower part of the figure shows that our framework can correctly identify all frames in the trace. We observe that the framework successfully classifies data packets, acknowledgments, beacons, and inter-frame spacing. Moreover, Fig. 13 also demonstrates the need for our EDHMM approach. The HMM method wrongly classifies many of the samples—within a data or acknowledgment frame, it often fluctuates between states. In contrast, the EDHMM classifies all samples correctly, even in case of varying data packet lengths.

In addition to the visual inspection in Fig. 13, we validate our framework using two approaches. First, we compare the number of data packets that our tool identifies with the number of packets that the driver of our 60 GHz device reports. For the case without blockage in Fig. 14, the driver reports 31960 sent packets at the end of the trace. This matches the data packet counter in our results. Second, we record the same data exchange using two independent sniffers SN_1 and SN_2, and process the resulting traces using our framework. For no blockage, Fig. 14 shows that both sniffers count the same number of both data and control packets. This again validates that our framework is correctly decoding the trace. For data packets, the counter stabilizes at one second at which point we stop the data transmission. Still, the control packet counter increases steadily because the devices continue to exchange control packets even if no data transmission is taking place.

Fig. 14 also depicts similar measurements for two blockage cases. The first is a "hard blockage", i.e., crossing the link and thus interrupting it completely for a few milliseconds. The second is a "soft blockage", which refers to partial blockage such as waiving a hand in the boresight of the antenna. These blockages cause a drop in the energy levels captured by the two independent sniffers SN_1 and SN_2, which actually perceive a different number of both data and control packets due to the different relative positions of the sniffers with respect to the mm-wave link.

Regular operation: in the following, we show some selected diagnosis capabilities of our tool for regular link operation. Fig. 15 depicts the Empirical Cumulative Distribution Function (ECDF) of the packet and burst lengths that our tool computes for different links. All links have the same length and are deployed in the same location. However, we change their orientation to induce different antenna beam patterns which

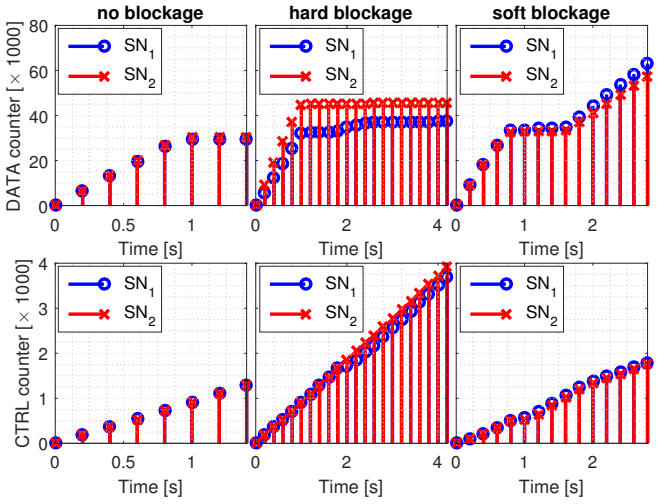

Fig. 14. Number of data and control packets identified by our tool. We show the results for two sniffers SN_1 and SN_2 placed at different locations.

result in suboptimal performance, and which our framework can identify. The protocol used by our 60 GHz test devices defines that the maximum burst length is two milliseconds and the maximum aggregated packet length is 20 microseconds [6]. Since we perform this experiment with full transmission buffer at the nodes, the burst and packet lengths should match the maximum values.

For the different measurements, the link distance is maintained to be equal to 3 meters, while the rotation of the nodes varies, resulting in changes in the MCS and frame duration. Specifically, for Link 3 the antennas of the devices are facing one another, whereas for Link 1 and 2 they are not. While the MCS of Link 1 and 2 are the same, for Link 1 in Fig. 15 we observe smaller packet durations. To reduce the packet error rate when the link quality is worse, the MAC reduces the level of aggregation, i.e., the MAC layer aggregates fewer data packets than the maximum into a single MAC packet. Indeed, our framework also reveals that the trace energy level differs compared to Link 2, which suggests antenna misalignment. We omit the energy trace level in the interest of space but the device driver reveals that both Links 1 and 2 operate otherwise identically in terms of MCS and traffic load. In other words, our framework successfully identifies the suboptimal device orientation for Link 1. Fig. 15 shows that Link 3 performs even better in terms of packet length. Again, the device driver confirms this insight since Link 3 uses a more robust MCS than Link 2. Thus, Link 3 is more likely to succeed when transmitting longer packets.

External disturbance: regarding external disturbances, we focus on the case of link blockage. Our tool is able to identify and classify such blockage. This provides means for network operators to determine how often blockage actually occurs for a certain mm-wave link during a certain time-frame, for instance, a day.

Identifying blockage is challenging because it may block the LOS path to the sniffer, too. To prevent this, our framework can record and compare the channel activity from two or more sniffers at different locations, as shown in Fig. 16. We

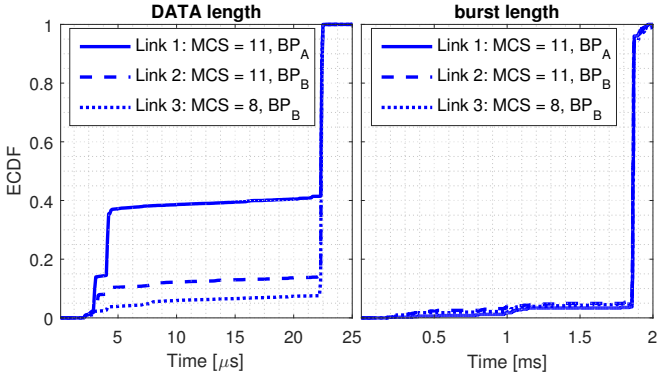

Fig. 15. CDF of packet and burst lengths for three links deployed in the same environment but with varying performance. "BP" stands for beampattern.

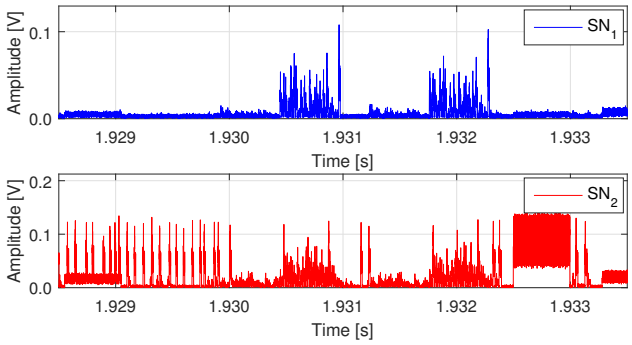

Fig. 16. Blockage recorded from two different locations SN_1 and SN_2. The figure shows a fraction of the blockage, i.e., the blockage affects all samples.

observe that while sniffer SN_1 barely receives any of the activity prior to second 1.93, SN_2 is able to receive all frames during the blockage. This allows our framework to obtain a much more complete view of the activity on the channel. Based on this information, we automatically identify beam refinement (BR) sequences. Such sequences are rare in static scenarios but are likely to occur if the link is impaired. Fig. 17 depicts a segment of the trace in Fig. 16, overlapped with the locations at which our framework identifies BR sequences. We observe that the BRs identified by both sniffers match but that not all sniffers capture all sequences due to the blockage. This highlights again the benefit of being able to analyze the network behavior from multiple viewpoints. Moreover, Fig. 16 depicts a soft blockage. Thus, the connection does not break and the device continuously adapts its beampattern, resulting in a large number of BRs. In contrast, hard blockage results in less BRs since the transmitter and the receiver cannot communicate during the blockage. As per our measurements, the average number of BRs per trace for no blockage, hard blockage, and soft blockage is 0 BRs/trace, 0.42 BRs/trace, and 4.02 BRs/trace, respectively. The difference in terms of BR frequency allows our diagnosis tool to classify blockage. This is highly valuable to determine why a mm-wave link is performing poorly.

In Fig. 14, we also show packet counters for the case of blockage. The data packet counter for hard blockage stabilizes at roughly $40,000$ packets because we stop transmission at that point. For soft blockage, we transmit continuously and thus the packet counter increases throughout the trace. We observe

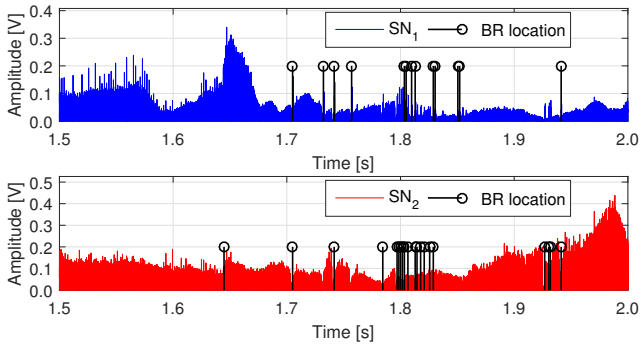

Fig. 17. Beam refinement sequences during soft link blockage.

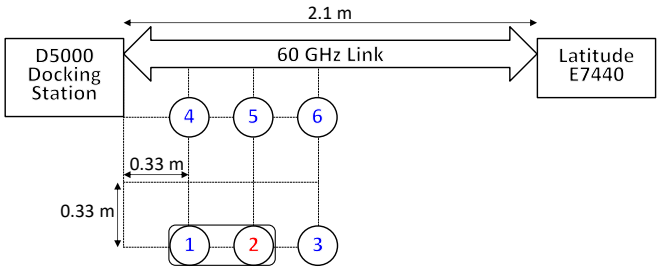

Fig. 18. Indoor measurement setup: SN_1 and SN_2 are omnidirectional, thus we do not experience errors in the energy levels due to antenna misalignment.

that the data packet counters for each sniffer disagree as soon as blockage occurs. The underlying reason is that one of the sniffers does not receive the full channel activity while the other one does. While not as unambiguous as the detection of BRs, this also hints at potential blockage scenarios. We observe that hard blockage causes a stronger disagreement than soft blockage, providing means to differentiate them. The mismatch among sniffers is less explicit for control packets since the shape of such patterns is easier to identify than the shape of data packets. Hence, both sniffers are more likely to correctly classify such control packets even in case of blockage.

Combining different viewpoints: next, we jointly process mm-wave channel traces from multiple time-synchronized sniffers, since different viewpoints of the same channel will provide complementary information and thus can lead to higher decoding accuracies. We performed several experiments deploying the transmitter, the receiver and the sniffers as shown in Fig. 18. Out of several tests/combinations, for the sake of brevity we only discuss the results for the sniffer pairs SN_1 and SN_2, as it is deemed sufficient to reveal the dynamics behind the decoding process. For such sniffer pair, in Fig. 19 we compare the number of data packets that our tool identifies in a controlled scenario with no link blockage: while both sniffers count the same number of control packets (as beacon pairs are robustly detected via template matching), the number of data packets differ, i.e., SN_2 provides better decoding accuracy than SN_1, as can also be observed from the ECDF in Fig. 20 (left plot). The multi-dimensional (joint) processing of SN_1 and SN_2 allows us to estimate the packet duration statistics in a more reliable fashion, see the left plot of

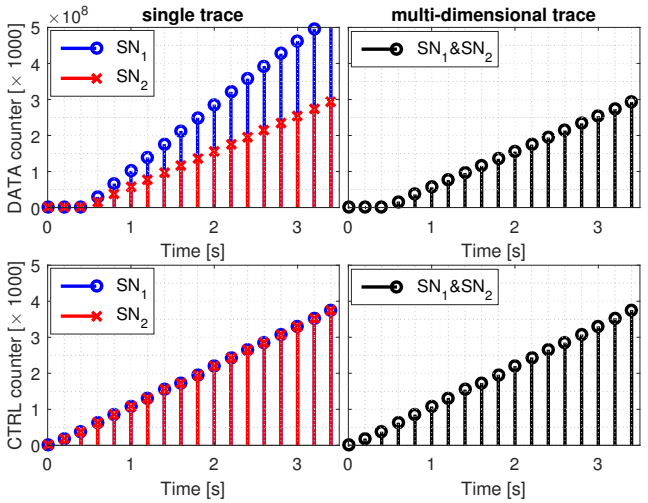

Fig. 19. Number of identified data and control packets for the two sniffers SN_1 and SN_2. The left and right plot respectively show the results from decoding the two traces independently and jointly.

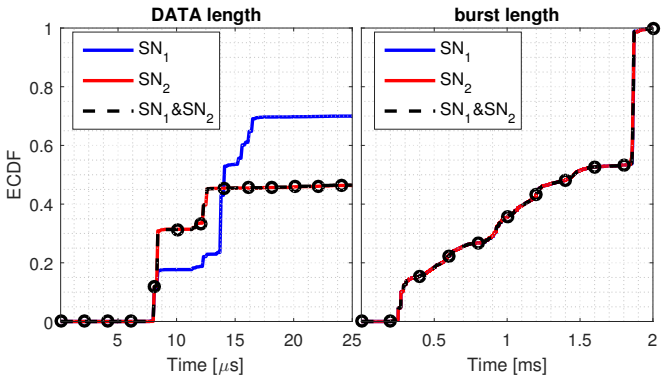

Fig. 20. ECDF of packet and burst lengths for the single trace and the multi-dimensional (joint) trace processing (SN_1 & SN_2).

Fig. 20 (dashed line), and to correct the bias in the DATA count for SN_1, see Fig. 19 (right plot). The reason why SN_2 provides higher decoding performance compared to SN_1 is that SN_2 records more distinctive values for the amplitude mean and standard deviation (std) of the EDHMM states $i \in \{1, 2, 3\}$, whereas for SN_1 it is difficult to discriminate among different energy levels at the receiver. This is shown in Fig. 21, where the estimated amplitude mean and standard deviation (std) of the EDHMM states $i \in \{1, 2, 3\}$ are plotted for all the sniffers from the measurement setup of Fig. 18. From this plot, we see that DATA and ACK packets for SN_1 are more likely to be erroneously classified by the EDHMM, as their energy levels are almost indistinguishable. Note that the difference in the received energy levels captured by the sniffers depends on their relative position with respect to the communicating devices.

B. Reconstruction from generated traces

For the performance evaluation in this section, we define the reconstruction factor $\rho \in [0, 1]$ as the fraction of samples in the received sequence that are correctly reconstructed by the algorithm. To this end, we compare the reconstructed sample sequence obtained by the Viterbi algorithm against the ground truth, see Section X. $\rho = 1$ means perfect reconstruction.

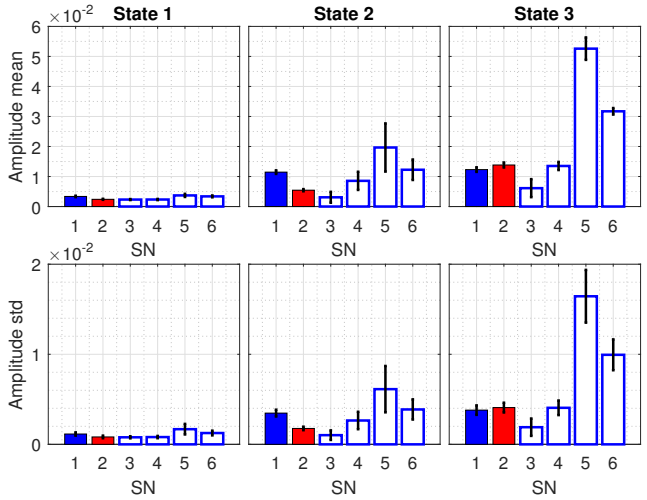

Fig. 21. Estimated amplitude mean and standard deviation (std) of the EDHMM states $i \in \{1, 2, 3\}$ for sniffers $\text{SN}_1, \text{SN}_2, \ldots, \text{SN}_6$ in Fig. 18.

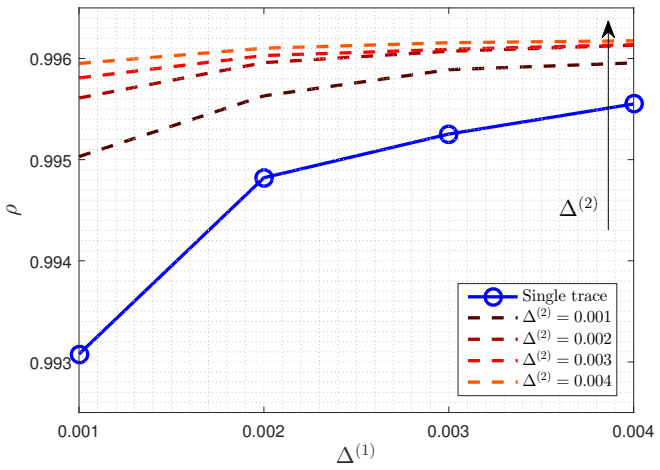

Fig. 22. ρ as a function of $\Delta^{(1)}$, keeping $\Delta^{(2)}$ fixed for each curve. For this plot, $\mu_1^{(\text{dim})} = 0.001$ for $\text{dim} = 1, 2$, the remaining energy levels follow from $\Delta^{(\text{dim})}$, whereas the variances $\Sigma_i^{(\text{dim})}$ are obtained through Eq. (5).

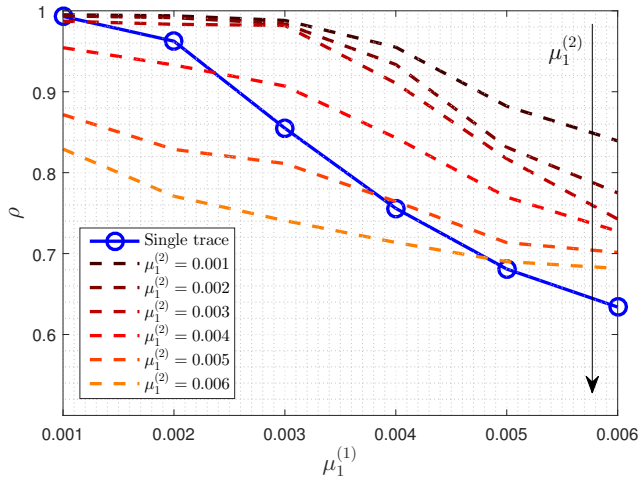

Fig. 23. ρ as a function of $\mu^{(1)}$ with $\Delta^{(1)} = \Delta^{(2)} = 0.001$, whereas $\mu_1^{(1)}$ and $\mu_1^{(2)}$ are allowed to change.

We consider two time-synchronized sniffers, $\text{dim} = 1, 2$. Moreover, with $\Delta_{ij}^{(\text{dim})}$ we indicate the difference in the received energy levels associated with IDLE periods, DATA and ACKs frames for the channel acquired by sniffer dim, i.e., $\Delta_{ij}^{(\text{dim})} = \mu_j^{(\text{dim})} - \mu_i^{(\text{dim})}$, with $i, j \in \{1, 2, 3\}$ and $i < j$. Specifically, we aim to evaluate the reconstruction factor $\rho \in [0, 1]$ as a function of $\Delta_{12}^{(\text{dim})}$ and $\Delta_{23}^{(\text{dim})}$, as $\Delta_{13}^{(\text{dim})}$ can be obtained from $\Delta_{13}^{(\text{dim})} = \Delta_{12}^{(\text{dim})} + \Delta_{23}^{(\text{dim})}$. In principle, the difference in the received energy levels captured by the sniffers depends on the relative position of the latter with respect to the communicating devices. For the performance evaluation in this section, without loss of generality, we consider $\Delta^{(\text{dim})} = \Delta_{12}^{(\text{dim})} = \Delta_{23}^{(\text{dim})}$ to conveniently evaluate the reconstruction factor $\rho \in [0, 1]$ through a single free parameter $\Delta^{(\text{dim})}$, simplifying the parameter space, and allowing a compact graphical representation of the results.

In Fig. 22, we plot ρ when the decoding is jointly performed over the traces from two sniffers as a function of $\Delta^{(1)}$ (energy gaps for sniffer 1, on the abscissa), keeping $\Delta^{(2)}$ fixed for each curve. For this plot, $\mu_1^{(\text{dim})} = 0.001$ for $\text{dim} = 1, 2$, the remaining energy levels follow from $\Delta^{(\text{dim})}$, whereas the variances $\Sigma_i^{(\text{dim})}$ are obtained through Eq. (5). For comparison, the case of a single sniffer ($\text{dim} = 1$) is also shown through the solid blue curve. As expected, the reconstruction factor ρ increases with an increasing $\Delta^{(1)}$, since the runtime mm-wave analyzer in this case better initializes the EDHMM parameters for each data burst. A single trace ($\text{dim} = 1$) leads to excellent results even for $\Delta^{(1)} = 0.001$, and using two traces allows for a further improvement.

However, we remark that this good performance is not always possible and this very much depends on the chosen initial levels $\mu_1^{(\text{dim})}$, $\text{dim} = 1, 2$. To see this, we have considered a scenario where the distance between energy levels is rather small, i.e., $\Delta^{(1)} = \Delta^{(2)} = 0.001$, whereas $\mu_1^{(1)}$ and $\mu_1^{(2)}$ are allowed to change. These results are plotted in Fig. 23, where $\mu_1^{(1)}$ appears on the x-axis and $\mu_1^{(2)}$ is kept fixed for each curve. The single trace performance is shown for comparison through a solid blue line. In this case, we observe that the reconstruction performance is heavily affected by an increasing background noise level μ_1. In fact, as μ_1 gets larger, according to Eq. (5), the noise variance also increases (almost quadratically) and since the energy gap (Δ) between levels remains constant, it becomes increasingly difficult to discriminate among different energy levels at the receiver. (ii) Moreover, there is a fundamental tradeoff in the number of sniffers to use. For example, for $\mu_1^{(1)} = 0.003$, the addition of a second sniffer helps if $\mu_1^{(2)}$ is smaller than or equal to 0.004. However, if the second trace has a very poor quality (e.g., $\mu_1^{(2)} = 0.005$) a single trace provides a better reconstruction performance. Interestingly, when $\mu_1^{(1)} \approx \mu_1^{(2)}$, it always pays off to use two sniffers, no matter the value of μ_1.

In the above paragraph (referring to Fig. 23), we have considered a scenario where the distance between energy levels is rather small, i.e., $\Delta^{(1)} = \Delta^{(2)} = 0.001$, and the background noise dominates the overall performance. It turns out that this choice of parameters is critical for proper EDHMM

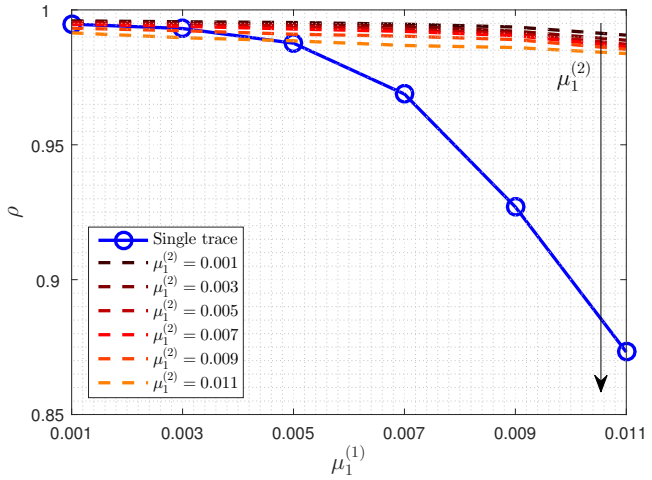

Fig. 24. ρ as a function of $\mu^{(1)}$ with $\Delta^{(1)} = \Delta^{(2)} = 0.002$, whereas $\mu_1^{(1)}$ and $\mu_1^{(2)}$ are allowed to change.

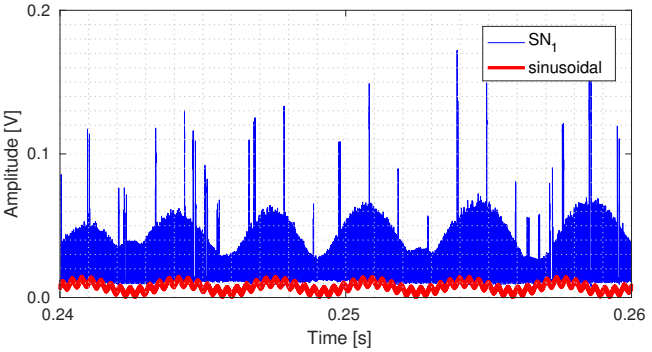

Fig. 25. An example of sinusoidal noise which was found in the channel dynamics. The reconstructed signal is given as a sum of $N = 3$ sine waves, by looking at the $N = 3$ highest peaks in the Fourier transform of the trace. Specifically, the main component has frequency 300 Hz, amplitude 0.0035 V.

functionality. Note that, even though these are extreme cases, such small values of Δ have been found in our measurements. However, if we allow for a slightly greater distance between energy levels, i.e., $\Delta^{(1)} = \Delta^{(2)} = 0.002$, then the overall performance increases significantly (see Fig. 24). Using two traces can lead to much better results, especially when the noise affecting the first one is significant, e.g., $\mu_1^{(1)} \geq 0.005$.

Now, we test the performance of our diagnosis tool by also inducing random fluctuations in the channel dynamics. That is, we account for an *additive* sinusoidal noise signal with frequency f, which is added to the generated traces. Then, we compare the performance of our diagnosis tool as a function of the frequency f, which is related to the average length of the data burst T_B. To do this, we proceed as follows: 1) given the channel transmission setup $M \in \mathcal{M}^{(0)}$, we compute the average length of the data burst T_B (in seconds); 2) we design an additive sinusoidal noise signal $\sin(2\pi ft)$ as a function of the frequency f, which is related to the average length of the data burst T_B, i.e., $\sin(2\pi ft) = \sin(2\pi k/T_B \ell T_s)$, where $\ell = 1, \ldots, L$ counts the samples in vector $\tilde{\boldsymbol{y}}^{(\dim)}$ and T_s is the sampling time of the generated traces (we use $T_s = 0.1\ \mu s$); 3) we test the runtime mm-wave analyzer for varying k, where

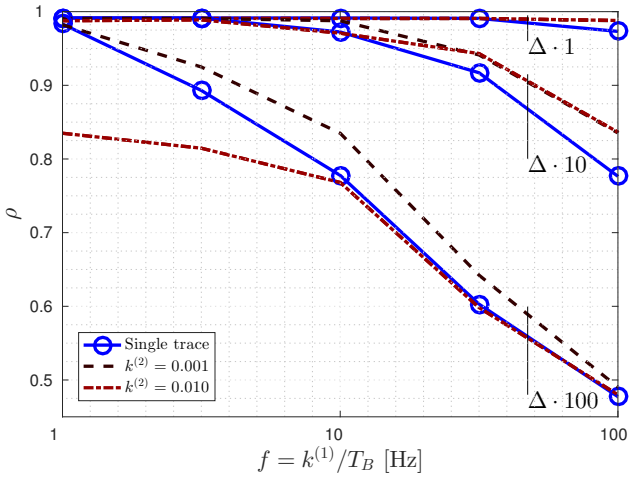

Fig. 26. ρ in the presence of an *additive* sinusoidal noise with frequency f.

$f = k/T_B$, and T_B depends on $P(d_{\text{burst}}^{(M)})$. Specifically, the sinusoidal (noise) signal $A \cdot [\sin(2\pi k/T_B \ell T_s) + 1]/2$ is added to the generated traces, where A is the maximum amplitude of the sine wave. In principle, superimposed sinusoidal noise can occur for several reasons, such as interference from another transmission, and mobility. Also, hardware impairments during data acquisition are a major cause of superimposed sinusoidal noise. These distortion effects are very hardware specific and, as such, the value of f can vary significantly, depending on the experimental setting. An example of sinusoidal noise which was found in the measured channel traces is shown in Fig. 25, together with a reconstructed signal which is representative of the superimposed sinusoidal noise.

In Fig. 26, we plot ρ when the decoding is jointly performed over the traces from two sniffers as a function of $k^{(1)}$ (the value of k associated with sniffer 1), thus $f = k^{(1)}/T_B$, whereas $k^{(2)}$ (sniffer 2) is kept fixed for each curve. Specifically, we considered $T_B = 1$ ms, which is close to the average length of the data burst, measured from real channel traces (see Fig. 11). For this plot, $\mu_1^{(\dim)} = 0.001$ for $\dim = 1, 2$, the remaining energy levels follow from $\Delta^{(\dim)}$, whereas the variances $\Sigma_i^{(\dim)}$ are obtained through Eq. (5). Also, $\Delta = 0.001$ for both traces. If we do not consider any additive sinusoidal noise signal, this choice of parameters leads to $\rho \simeq 1$. On the contrary, the performance of the runtime mm-wave analyzer decreases as a function of frequency f. This is reasonable, since channel attenuation and noise are no longer stationary within each data burst (see assumption A3, in Section V). Then, taking advantage of the joint information from two different viewpoints can lead to better results (with respect to a single trace) as long as at least one of the traces is not affected by heavy channel fluctuations, see, for example the curves with $k^{(2)} = 0.001$ ($f = 1$ Hz). Finally, the overall performance drastically drops for an increasing A, where A is the maximum amplitude of the sine wave. For this plot, we set A equal to 1, 10, and 100 times the gap between energy levels Δ. It results that $A = \Delta \cdot 100$, combined with values of f above 100 Hz, always results in $\rho < 0.5$.

XII. Conclusion

In this paper we have designed and evaluated a machine learning framework to automatically perform protocol layer analysis and diagnose physical layer issues in 60 GHz networks. The main challenge lies in the variability of the channel traces and in the complexity of identifying their structural elements given their noisiness, aperiodicity, and unpredictable behavior. Standard machine learning approaches fail in such a scenario. Our tool uses narrowband physical layer energy traces from one or multiple sniffers to identify lower-layer performance issues. Network planners, administrators, as well as researchers can use it to improve the performance of mm-wave networks even though COTS networking devices provide very limited access to lower-layer information. The developed approach provides a convenient trade-off between the efficient but limited monitoring capabilities of IEEE 802.11ad devices in monitor mode, and the highly detailed but costly decoding of full bandwidth signals through software-defined radios. Our algorithms are tested through an extensive measurement campaign using COTS 60 GHz hardware, considering a range of practical scenarios. Besides, the ability of the framework to correctly identify protocol sequences (beacon pairs, data and acknowledgment frames) from single and multiple channel sniffers is quantified, looking at the impact of channel noise, energy levels and their distribution across different sniffers.

XIII. Acknowledgements

This work was supported in part by the European Research Council grant ERC CoG 617721, the Ramon y Cajal grant from the Spanish Ministry of Economy and Competitiveness RYC-2012-10788, and the Madrid Regional Government through the TIGRE5-CM program (S2013/ICE-2919).

References

[1] S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter-Wave Cellular Wireless Networks: Potentials and Challenges," *Proc. of the IEEE*, vol. 102, no. 3, pp. 366–385, 2014.

[2] DARPA, "News." https://www.darpa.mil/news-events/2017-08-11a. Accessed: 2018-08-30.

[3] DARPA, "News." https://www.darpa.mil/news-events/2016-07-19a. Accessed: 2018-08-30.

[4] NSF, "Homepage." http://www.nsf.org/. Accessed: 2018-08-30.

[5] R. R. Choudhury and N. H. Vaidya, "Deafness: a MAC Problem in Ad Hoc Networks when using Directional Antennas," in *Proc. of IEEE ICNP*, (Berlin, Germany), Oct 2004.

[6] T. Nitsche, G. Bielsa, I. Tejado, A. Loch, and J. Widmer, "Boon and Bane of 60 GHz Networks: Practical Insights into Beamforming, Interference, and Frame Level Operation," in *Proc. of ACM CoNEXT*, (Heidelberg, Germany), Dec 2015.

[7] Y. Zhu, Z. Zhang, Z. Marzi, C. Nelson, U. Madhow, B. Y. Zhao, and H. Zheng, "Demystifying 60GHz Outdoor Picocells," in *Proc. of MobiCom*, Sep 2014.

[8] S. K. Saha and D. Koutsonikolas, "Towards Multi-gigabit 60 GHz Indoor WLANs," in *Proc. of IEEE ICNP*, (San Francisco, CA, USA), Nov 2015.

[9] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band," *IEEE Std 802.11ad-2012*, 2012.

[10] M. K. Haider and E. W. Knightly, "Mobility Resilience and Overhead Constrained Adaptation in Directional 60 GHz WLANs: Protocol Design and System Implementation," in *Proc. of ACM MobiHoc*, (Paderborn, Germany), Jul 2016.

[11] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra, "BeamSpy: Enabling Robust 60 GHz Links Under Blockage," in *Proc. of USENIX NSDI*, (Santa Clara, CA, USA), Mar 2016.

[12] S. Sur, V. Venkateswaran, X. Zhang, and P. Ramanathan, "60 GHz Indoor Networking Through Flexible Beams: A Link-Level Profiling," in *Proc. of ACM SIGMETRICS*, (Portland, Oregon, USA), Jun 2015.

[13] R. do Carmo and M. Hollick, "DogoIDS: A Mobile and Active Intrusion Detection System for IEEE 802.11s Wireless Mesh Networks," in *Proc. of ACM HotWiSec*, (Budapest, Hungary), Apr 2013.

[14] D. S. Berger, F. Gringoli, N. Facchi, I. Martinovic, and J. Schmitt, "Gaining Insight on Friendly Jamming in a Real-world IEEE 802.11 Network," in *Proc. of ACM WiSec*, (Oxford, UK), Jul 2014.

[15] T. T. T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification Using Machine Learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[16] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow Clustering Using Machine Learning Techniques," in *Passive and Active Network Measurement (PAM)*, Springer, Berlin, Heidelberg, 2004.

[17] J. Mitola and G. Q. Maguire, "Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.

[18] S. Haykin, "Cognitive Radio: Brain-Empowered Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, 2005.

[19] DARPA, "Homepage." https://www.darpa.mil/. Accessed: 2018-08-30.

[20] M. G. D. Benedetto, S. Boldrini, C. J. M. Martin, and J. R. Diaz, "Automatic Network Recognition by Feature Extraction: A Case Study in the ISM Band," in *Proc. of IEEE CROWNCOM*, (Cannes, France), Jun 2010.

[21] M. Gandetto and C. Regazzoni, "Spectrum Sensing: A Distributed Approach for Cognitive Terminals," *IEEE JSAC*, vol. 25, no. 3, pp. 546–557, 2007.

[22] P. Varshney, *Distributed Detection and Data Fusion*. Springer New York, 2012.

[23] M. Gandetto, M. Guainazzo, and C. S. Regazzoni, "Use of Time-frequency Analysis and Neural Networks for Mode Identification in a Wire-less Software-defined Radio Approach," *EURASIP JASP*, pp. 1778–1790, 2004.

[24] H. Assasa and J. Widmer, "Implementation and Evaluation of a WLAN IEEE 802.11ad Model in ns-3," in *Proc. of the Workshop on ns-3*, (Seattle, WA, USA), Jun 2016.

[25] D. Steinmetzer, D. Wegemer, M. Schulz, J. Widmer, and M. Hollick, "Compressive Millimeter-Wave Sector Selection in Off-the-Shelf IEEE 802.11ad Devices," in *Proc of. ACM CoNEXT*, (Incheon, Republic of Korea), Dec 2017.

[26] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[27] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos, "A Survey of Millimeter Wave Communications (mmWave) for 5G: Opportunities and Challenges," *Wireless Networks*, vol. 21, no. 8, pp. 2657–2676, 2015.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood From Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[29] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[30] L. E. Baum and J. A. Eagon, "An Inequality With Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology," *Bulletin of the American Mathematical Society*, vol. 73, no. 3, pp. 360–363, 1967.

[31] L. E. Baum and G. R. Sell, "Growth Transformations for Functions on Manifolds," *Pacific Journal of Mathematics*, vol. 27, no. 2, pp. 211–227, 1968.

[32] S.-Z. Yu, "Hidden Semi-Markov Models," *Artificial Intelligence*, vol. 174, no. 2, pp. 215–243, 2010.

[33] S.-Z. Yu and H. Kobayashi, "An Efficient Forward-Backward Algorithm for an Explicit-Duration Hidden Markov Model," *IEEE Signal Processing Letters*, vol. 10, no. 1, pp. 11–14, 2003.

[34] S.-Z. Yu and H. Kobayashi, "Practical Implementation of an Efficient Forward-Backward Algorithm for an Explicit-Duration Hidden Markov Model," *IEEE Trans. on Signal Processing*, vol. 54, no. 5, pp. 1947–1951, 2006.

[35] A. V. Oppenheim and G. C. Verghese, *Signals, Systems and Interference*. Pearson, 2015.

[36] D. Garcia, "Robust Smoothing of Gridded Data in One and Higher Dimensions with Missing Values," *Computational Statistics & Data Analysis*, vol. 54, no. 4, pp. 1167–1178, 2010.

[37] D. Garcia, "A Fast All-in-One Method for Automated Post-Processing of PIV Data," *Experiments in Fluids*, vol. 50, no. 5, pp. 1247–1259, 2011.

[38] J. Lee Rodgers and W. A. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[39] T. Kailath and H. V. Poor, "Detection of Stochastic Processes," *IEEE Trans. on Information Theory*, vol. 44, Oct 1988.

[40] A. Mueen, H. Hamooni, and T. Estrada, "Time Series Join on Subsequence Correlation," in *Proc. of IEEE ICDM*, (Shenzhen, China), Dec 2014.

[41] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: an expressive primitive for time series classification," in *Proc. of ACM SIGKDD*, (San Diego, CA, USA), Aug 2011.

[42] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band," *IEEE Std 802.11ad-2012*, Dec 2012.

[43] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[44] S. T. Roweis, "Constrained Hidden Markov Models," in *Proc. of NIPS*, (Denver, Colorado, USA), Nov 1999.

[45] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proc. of BSMSP*, vol. 1, pp. 281–297, 1967.

[46] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," in *Proc. of ACM-SIAM SODA*, (New Orleans, Louisiana, USA), Jan 2007.

[47] S. E. Levinson, "Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition," *Computer Speech & Language*, vol. 1, no. 1, pp. 29–45, 1986.

[48] C. Mitchell and L. Jamieson, "Modeling Duration in a Hidden Markov Model with the Exponential Family," in *Proc. of IEEE ICASSP*, (Minneapolis, MN, USA), Apr 1993.

[49] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the Number of Clusters in a Data Set via the Gap Statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[50] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, "Locating the Nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine* , vol. 22, no. 4, pp. 54–69, 2005.

[51] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A Ubiquitous WiFi-based Gesture Recognition System," in *Proc. of IEEE INFOCOM*, (Kowloon, Hong Kong), Apr 2015.

Maria Scalabrin is currently pursuing the Ph.D. in Information Engineering within the SIGnal processing and NETworking (SIGNET) research group, University of Padova, Italy. She received the B.Sc. degree in Information Engineering and the M.Sc. degree in Telecommunication Engineering from the University of Padova, Italy, in 2013 and 2015, respectively. In 2016, she was on leave at IMDEA Networks Institute, Madrid, Spain. In 2018, she was on leave at Purdue University, West Lafayette, Indiana, US, working on adaptive communication strategies in millimeter-wave vehicular networks. Her present research interests include dynamic programming, signal processing, and machine learning with application to wireless networks and millimeter-wave channels.

Guillermo Bielsa is currently a Research Assistant at IMDEA Networks in Madrid, Spain. He is also with Universidad Carlos III de Madrid (UC3M), where he is enrolled as a Ph.D. student. He obtained both his bachelor's degree in Communication System Engineering and his master's degree in Multimedia and Communications from UC3M in 2015 and 2016, respectively. His current research activity involves the study of millimeter-wave communications in the 60 GHz band, characterizing the mm-wave channel and developing algorithms.

Adrian Loch is a post-doc researcher at IMDEA Networks in Madrid, Spain. He graduated in Electrical Engineering from Universidad Politécnica de Madrid and Technische Universität Darmstadt in 2011 after completing an international double degree program. After that, he obtained a Ph.D. in Computer Science from Technische Universität Darmstadt in March 2015. During his Ph.D., he was a research associate at the Secure Mobile Networking Lab. His main areas of interest lie in cooperative communications for both wireless access and wireless multihop networks, including routing issues as well as practical validation on wireless testbeds. Currently, he focuses on millimeter-wave communications and, in particular, wireless LANs such as in the 802.11ad standard.

Michele Rossi is an Associate Professor with the Dept. of Information Engineering at the University of Padova, Italy. His research interests lie in wireless sensing, green mobile networks and wearable computing. In the last few years, he has been actively involved in EU projects on IoT technology (IOT-A, FP7-ICT- 2009-5, project no. 257521) and has collaborated with SMEs such as Worldsensing (Barcelona, ES) in the design of optimized IoT solutions for smart cities and, with SAMSUNG and INTEL. In 2014, he has been the recipient of a SAMSUNG GRO award with a project entitled "Boosting Efficiency in Biometric Signal Processing for Smart Wearable Devices". Since 2016, he has been collaborating with INTEL on the design of IoT protocols exploiting cognition and machine learning, as part of the INTEL Strategic Research Alliance (ISRA) R&D program. His research is also supported by the European Commission through the H2020 ITN SCAVENGE project (MSCA-ITN- ETN, project no. 675891) on green 5G networks. He is author of more than 130 scientific papers and has been the recipient of five best paper awards from the IEEE. Dr. Rossi currently serves on the Editorial Board of the IEEE Transactions on Mobile Computing and is a Senior Member of the IEEE.

Joerg Widmer is Research Professor as well as Research Director of IMDEA Networks in Madrid, Spain. His research focuses on wireless networks, ranging from extremely high frequency millimeter-wave communication and MAC layer design to mobile network architectures. From 2005 to 2010, he was manager of the Ubiquitous Networking Research Group at DOCOMO Euro-Labs in Munich, Germany, leading several projects in the area of mobile and cellular networks. Before, he worked as post-doctoral researcher at EPFL, Switzerland on ultra-wide band communication and network coding. He was a visiting researcher at the International Computer Science Institute in Berkeley, USA, University College London, UK, and TU Darmstadt, Germany. Joerg Widmer authored more than 150 conference and journal papers and three IETF RFCs, and holds 13 patents. He serves or served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Communications, Elsevier Computer Networks and the program committees of several major conferences. He was awarded an ERC consolidator grant, the Friedrich Wilhelm Bessel Research Award of the Alexander von Humboldt Foundation, Mercator Fellowship of the German Research Foundation, a Spanish Ramon y Cajal grant, as well as seven best paper awards. He is senior member of IEEE and ACM.