Aletheia: A Lightweight Tool for WiFi Medium Analysis on The Edge

Mohammed Elbadry, Fan Ye, Peter Milder
Electrical and Computer Engineering
Stony Brook University
New York, USA
{mohammed.elbadry, fan.ye, peter.milder}@stonybrook.edu

***Abstract*—With the plethora of wireless devices in limited spaces running multiple WiFi standards (802.11 a/b/g/n/ac), gaining an understanding of network latency, loss, and medium utilization becomes extremely challenging. Microsecond timing fidelity with protocols is critical for network performance evaluation and design; such fine-grained timing offers insights that simulations cannot deliver (e.g., precise timing through hardware and software implementations). However, currently there is no suitable efficient, lightweight tool for such purposes. This paper introduces Aletheia, an open-source tool that enables users to select their interested attributes in WiFi frames, quantify and visualize microsecond granularity medium utilization using low-cost commodity edge devices for easy deployment. Aletheia uses *selective attribute extraction* to filter frame fields; it reduces data storage and CPU overhead by 1 and 2 orders of magnitude, respectively, compared to existing tools (e.g., Wireshark, tshark). It provides flexible tagging and visualization features to examine the medium and perform different analysis to understand protocol behavior under different environments. We use Aletheia to capture and analyze 120M frames in 24 hours at 4 locations to demonstrate its value in production and research network performance evaluation and troubleshooting. We find that WiFi management beacons can consume medium heavily (up to 40%); the common practice of categorizing networks based on environment types (e.g., office vs. home) is problematic, calling for a different evaluation methodology and new designs.**

I. Introduction

Internet of Things (IoT) devices leveraging wireless networks based on 802.11 standards [1], [2] are ubiquitous in enterprise environments. Today's enterprise WiFi environments (e.g., office buildings, hospitals, college campuses, and technology parks) are teeming with access points. These deployments usually support hundreds of devices (with various link characteristics) concurrently on different frequencies (5/2.4 GHz) of 802.11 standards (b/g/n/ac). Concurrent operation of many wireless devices on multiple standards impact and change medium utilization rapidly at fine time granularity (microsecond). As a result, the dynamics of today's enterprise wireless networks are extremely complex and difficult to understand, yet such understanding is critical in wireless network designs (e.g., improving latency and loss).

Access points are usually distributed within buildings to provide coverage; they are not always in the same aisle or room where the user is located. Network coverage in some places can be very low due to either fading characteristics and signal attenuation (e.g., obstacles like walls), or too many wireless devices heavily consuming the medium. Although the first case might be discovered easily by walking into that area and looking at RSSI or CSI values, no suitable tools exist to measure and detect the second case, which is complicated and time-varying, thus far more challenging to diagnose.

Further, IoT researchers and wireless network industry professionals care about latency and throughput; they are critical to access control and future generations of systems [3], [4], [5]. The Medium Access Control (MAC) layer gets to decide when to transmit based on algorithms designed (either scheduled access control or based on sensing like in 802.11a/b/g/n/ac). Most of existing WiFi IoT platforms rely on CSMA; the latency and throughput are highly dependant on wireless medium utilization.

Thus, there is the need for an efficient, lightweight wireless medium utilization tool that can quantify the timing of each frame, visualize their statistics to help identify patterns, and derive insights to troubleshoot networks or improve designs. The closest tools are Wireshark/tshark [6] and kismet [7]. Wireshark and tshark dump all frame headers and payload. In busy WiFi environments, one monitoring point can generate tens of gigabytes of data in a few hours, making both storage and analysis difficult and thus infeasible for long-term (e.g., weeks or months) monitoring. Kismet is mainly an Intrusion Detection System (IDS) and does not provide any capability to the user to visualize, customize filtering, or view medium utilization.

Contributions. In this paper, we design and build Aletheia, an efficient, lightweight open-source tool [1] for wireless medium utilization and analysis at microsecond granularity. We summarize our contributions below:

- Aletheia's edge monitor enables users to collect data on the edge; the tool runs on small, low-cost commodity hardware (e.g., embedded systems running Linux and WiFi dongles such as Raspberry Pi), making it convenient for deployment and monitoring. The tool uses *selective attribute extraction* (SAE) on the edge to store only certain fields (using simple predicate based policies) instead of the whole frame to reduce the size of stored data by an order of magnitude compared to existing tools; the filtered data gets stored in a filter log locally.

[1] https://github.com/SBU-MoCA/Aletheia-WiFi

- Aletheia's protocol analyzer can run on the edge or cloud. Using the filter log (transferred by user), the analyzer provides full freedom through a command line interface to add attributes based on operations among logged attributes, and tags based on multiple policies that can span different attributes with various Boolean expressions. It provides visual and textual analysis with flexible parameters for obtaining information from logged data.
- We analyze 120M packets over 24 hours in 4 different environments (2 homes and 2 offices), and observe several fundamental issues in current WiFi standards and evaluation practices: medium utilization greatly impacts latency (e.g., tripled when medium utilization goes from 20% to 70%); management beacon frames incur heavy overhead (up to 40% utilization); the common methodology of evaluating designs under different types of environments is problematic, because medium utilization is the determinant factor, yet it varies in the same type of environment, and at different hours/minutes of the day.

Although our evaluation and toolset are strictly for the WiFi medium, the same analysis and concepts can apply on any wireless medium frequency and technology. We believe that changes to the current WiFi beacon design are needed to alleviate the heavy overhead and a rigorous medium measurement tool based evaluation methodology is necessary for proper understanding of protocol behavior and designs.

II. Background & Related Work

A. 802.11

802.11 is an IEEE standard for WiFi. It consists of multiple layers; each has its own unique headers and offers different functionality. The medium access control layer (MAC) facilitates shared access to the wireless medium. The physical layer (PHY) handles data modulation such that other nodes can receive and decode the transmission. The timing of frame transmissions is controlled by PHY. The PHY layer header consists of two parts: 1) *preamble* is used to synchronize antenna circuits on boundaries of symbols for correct decoding; it is always transmitted at the slowest most reliable base rate: 1 Mbps. 2) *scheme indicator* specifies the frame rate, modulation scheme, and coding scheme. The *scheme indicator* determines the *preamble* duration, which is critical to find the exact start time of a frame.

B. Related Work

None of the existing monitoring tools provide real-time attribute extraction or visualizing medium with custom tags. One of the most commonly used tools, Wireshark/tshark [6], provides a user-friendly, frame-level data view of header values over multiple layers (i.e. MAC, network, and upper layers) with microsecond timing. However, these tools can either store the whole frame including headers/payload that are not always of interest to the user, which can lead to tens of gigabytes in few hours in busy medium; or filter by frames, which leads to incorrect representation of medium utilization in an environment. Further, it does not have flexibility to distinguish frames from custom designed protocol header fields, and is limited to only those known and built-in by tool developers (MAC and network headers). It does not calculate the begin/end times or duration of a frame's air-time, nor provide the capabilities to tag specific kinds of frames by attribute and visualize their medium utilization. There are other tools similar to Kismet [7] which provide Layer analysis and Intrusion Detection Systems (IDS) by analyzing specific headers. It also is unique by associating stations (addr2) by access points (addr3) enabling it to detect hidden networks. Our work has such capabilities; it focuses on providing a flexible interface, allowing for customized attribute extraction and analysis. Thus, we find that Kismet can entirely use our edge monitor (explained in section III-A), and run its algorithms to extract information it seeks to analyze for higher efficiency.

Jigsaw [8] is a work that focuses on frame timestamp unification from multiple monitoring points and deals with clock skews and drifts. It does not provide medium utilization. The code only supports 802.11g, and is no longer available. There has been some work [9], [10], [11] on larger scale networks that analyze data at second granularity from hundreds of nodes, for detecting specific network misconfigurations [12], or identifying types of packet losses [13]. All existing work focuses on individual packets, and none of them offers comprehensive medium utilization quantification and analysis functions. Aletheia fills this void, providing an efficient, lightweight tool for WiFi medium utilization analysis using low cost embedded devices with limited resources.

III. Design

We start by introducing the design goals, assumptions, and challenges, then provide a brief system architecture overview.

Goals and Assumptions. The system's goals are the following: *i)* provide researchers and wireless industry professionals with a toolset that enables them to analyze wireless medium in any environment with ease, *ii)* support long-term medium analysis that may exceed one day of continuous analysis, *iii)* work with most low-cost commodity radio devices; *iv)* understand the environment's medium utilization and extract key observations by providing the capability to draw graphs from logged information with few parameters quickly. Based on these goals, we make the assumption that the user is willing to distribute extra commodity hardware around the environment they wish to analyze.

Challenges. We face several challenges designing an efficient, lightweight monitoring tool: *i)* logging full frames in busy enterprise traffic can produce hundreds of gigabytes of data in less than one day, quickly exceeding the storage of commodity devices. Uploading all such data to a backend requires wiring and complicates deployment. Further, out of a full-size frame, usually only tens of bytes per frame are of interest to the user. Recording all data then repeatedly extracting a tiny fraction from millions of captured frames is wasteful and inefficient in both storage and processing. Thus an efficient

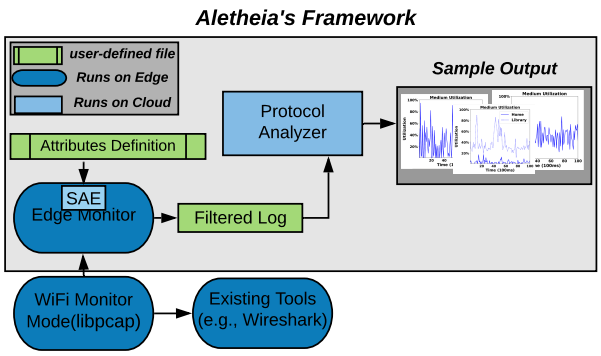


Fig. 1: Structures of Aletheia and existing tools. The Edge Monitor taps the WiFi monitor mode mechanism and uses selective attribute extraction (SAE) to filter and extract per-user attributes of interest while discarding the rest, unlike existing tools that store everything. The protocol analyzer, which runs on the cloud, takes the Filtered Log; it enables the user to analyze and visualize data in whatever fashion they seek with simple configurations.

upfront extraction is necessary to retain only interested data. *ii)* The user requires flexibility to analyze the data based on their *selected* attributes with complete freedom. Therefore, Aletheia has to be flexible to accommodate the user's requirements. *iii)* Low cost commodity WiFi radios have many fidelity and correctness issues (e.g. missing attributes, corrupt information due to CRC or other PHY/driver bugs). The tool has to account for all such defects without hindering performance.

System Overview. Aletheia uses a pre-defined *Attribute Definition File (ADF)* to know what the user is interested in extracting per-frame. The ADF provides instructions to the *Edge Monitor* that runs on low-cost commodity hardware to extract all information of interest to the user; the *Edge Monitor* supports extracting generic fields that exist in every frame (e.g., MAC address, TSFT, etc.), and complex conditional attributes (e.g., protocol version of SMTP packets). Upon the end of logging duration, a *Filtered Log File* is created that contains all attributes extracted from frames received and information necessary for the *Analyzer* to understand how to parse the generated file. The *Analyzer* handles data cleaning of corrupted information obtained due to low-cost hardware. Also, it provides the user with the opportunity to tag frames based on attributes (either defined in the ADF, or created by combination of existing attributes), and extract summary digest, unique values, and graph data based on custom conditions (e.g., select attributes to graph, start and end index for each axis, apply filters).

A. On The Edge

The edge monitor efficiently filters received frames based on the user's *Attribute Definition File (ADF)*; the frames are received through WiFi's monitor mode (using *libpcap* library [14]), and fields are stored into a *Filtered Log*), which will later be processed by the protocol analyzer (Fig. 1). In this section, we detail how we enable the user to easily define their attributes in the ADF and how the edge monitor efficiently extracts attributes per-frame.

Attribute Definition File (ADF) Structure. The ADF is designed for two purposes: *i)* enable the user to define attributes of interest, which can be in every frame or only some frames; *ii)* provide the *Edge Monitor* a clear approach to create its internal data-structures that enable the tool to perform efficient extraction. Thus, the ADF consists of two sections: *Generic Attributes (GA)* and *Conditional Attributes (CA)*. GA are the attributes that exist in every frame received by the interface. For example, let us assume the user is interested in the following attributes: frame size, layer 2 address information, and *radiotap header* [15]. The user defines them by simple instructions per attribute: label of attribute, type of attribute (i.e. string, number), attribute data grouping (e.g., group every 2 bytes for MAC address and 1 byte for IP addresses); and attribute location within frame. CA is where the user defines particular fields that rely on sequential conditional requirements. As an example: if the user wishes to extract from SMTP packets the SMTP version, the user will define IP packet header, UDP header information, and then the SMTP header flag in that sequence. For successful execution, the user must be aware of all conditions where such field may exist and values (e.g., only check for SMTP if the size of frame is larger than or equal to where the SMTP header is. The user is able to define conditions using symbols for conditional linkage (e.g., "->" for the next condition of the attribute).

Edge Monitor. It handles extracting and storing attributes of interest to the user instead of the whole frame. We design the internals of the edge monitor to be efficient while building the data-structures to filter against the indicated ADF appropriately. We face the challenge of repeated computation across multiple CA (e.g., the first CA requests checking if source IP address is 132.168.3.1 to extract destination IP address and the fourth CA requests checking if the source IP address is xx to extract application layer header). *Selective Attribute Filtering* leverages a standard linked-list structure for GA extraction and a multi-level linked-list structure for CA (e.g., Fig 2). If the CA are directly checked on the frame, there can be redundancy (as mentioned previously in the example checking source IP address twice). To prevent this issue, we provide the option to hash the conditional procedure by combining the clause and answer; this leads to filtering a frame once and check for hash validation later. Hashing solution validity greatly saves system performance when CA must check tens or hundreds of bytes (e.g., values of 100 bytes of information) which we show later in the evaluation (section IV). However, this also increases the memory allocation to do the mapping, thus we make this a user-selectable option.

B. On The Cloud

The *Protocol Analyzer* allows the user to visualize and extract data based on their defined attributes with flexibility; it enables creating custom parameters combining different attributes and applying tags. In this section, we describe how to use the analyzer, and how it is designed in four stages:

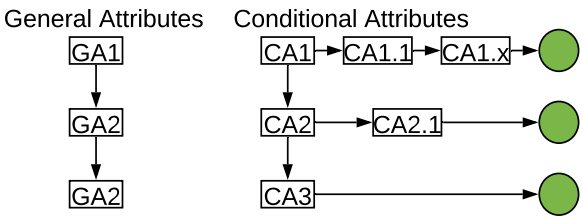


Fig. 2: The internal data-structure the edge monitor uses to extract attributes of interest within each frame. The conditional attributes rely on a multi-level linked list as some attributes may rely on multiple conditions before extraction.

initializing *parameters*, performing *operations*, creating *tags*, and generating *views*.

Parameters and Operations. *Parameters* provide the user the capability to create artificial/custom parameters that can interact with collected logged data. As an example of what can be of interest to tool users: calculating medium utilization will require converting data-rate logged from the *radiotap* header from bits per second (bps) and the size of frame from bytes to common metrics; there will also be the need for a new parameter defined as *duration* that is used to store each frame's duration. For each parameter, users provide an *identification string* for later usage and a *type* (i.e. string or number). *Operations* are where the tool provides the user with the capability to define how the parameter they defined is to be used. Following the example described earlier: the *duration* parameter will require the following equation:

$$duration = \frac{rate * 125000}{size} + preambleDuration$$

The size and rate parameters are obtained from *radiotap* header information on frame basis. (This can sometimes be unavailable depending on WiFi dongle, discussed in section IV.). The last variable must be defined before being used. Thus, the user would have to calculate it per frame by looking at radiotap header information of *preamble* and computing:

$$preambleDuration = (preamble*96)+(1-preamble)*192$$

The preamble variable (also provided by radiotap) is a binary field indicating whether a short preamble (1) (96μs) or long preamble (0) (192μs) is used per frame. The equation allows the system to account for both short and long preamble durations. Similar operations can be obtained with correct attribute definitions per frame to define RTS/CTS backoff and all other medium utilization calculations to ensure accurate measurement.

Tag and View. Many times the user wants to examine a specific subset of frames (e.g., those of a particular type). To do so, the user needs the ability to apply simple and complex tagging policies that can be combined using Boolean operations (and, or, not) along with the flexibility to apply rules to any attribute (either computed during *parameters and operations*, or in the filtered file). For example, consider the task of tagging non-WiFi frames and frames with a specific source MAC address. First, the user will have to create two separate policies: one on the size parameter to detect non-WiFi frames (e.g., less than 70 bytes). Then, they would create another policy and apply it on source MAC address to detect the specific MAC address. In *View*, the user will choose which policies to apply and in which ways. Besides Boolean based policies to get the ideal tag, the user can tag two or more types of frames to have them in different colors to view them (e.g., frames of a given MAC address with color 1, non-WiFi frames with color 2, and all other frames with color 3).

IV. System Fidelity

Accurate Medium Utilization. Our toolset's main focus is an accurate representation of medium utilization in any environment using low-cost commodity WiFi radios. We survey nine different commonly used chipsets spanning 20 different dongles (e.g., ALFA, panda, TP-link, LOTEKOO, etc.). We observe few issues that may hinder the system's accuracy by impacting the frame duration accuracy; we discuss each case in detail based on our observations from the extensive evaluation. *i)* The preamble type header can be invalid; we take the most conservative approach by using the duration of *long preamble* (192 μs). Experiments show that such approach affects medium utilization by 0.5% at most, thus we consider it acceptable. *ii)* The frame rate field can also be invalid in very rare cases (occurring in <0.5% of all received frames). We allow the user to select a replacement; the default rate is set to 6 Mbps empirically based on our experiments. *iii)* Non-WiFi transmissions may appear as frames with very short, invalid sizes (< 100 bytes). We detect them by checking CRC and other header fields. We estimate their airtime using base rate (1Mbps) with no preamble. In all environments we have tested, they occupy 0.1–0.3% of medium even if treated as full frames, thus their impact is negligible. After considering these issues addressed above, we have high confidence in our system's correct and accurate representation of medium utilization using commodity WiFi dongles.

Edge System Efficiency. We evaluate our system's resource utilization on the edge by comparing our edge monitor with wireshark and tshark tools (Table I) running on Raspberry Pi 3B. We let three different Pis (with the exact same hardware, using the same image) run at the same time beside each other for one minute. We find that wireshark and tshark consume more CPU resources (24.1% and 10.3% versus Aletheia's edge monitor (<0.1%). Further, our system consumes less memory (<0.1%) than both wireshark and tshark (15.6% and 8.3%). We also find that when collecting all MAC header information and some network layer attributes, our system is still able to retain 5X smaller file sizes than standard tools used to log medium information. This is critical as Wireshark and tshark logged 11MB in 1 minute in a busy environment. This would scale to 15GB for 24 hours of data collection; meanwhile Aletheia would only require 1.5GB.

Long CA Impact. To evaluate CPU usage, we form a sequence of 12 attributes (strings and numbers) with 15 conditions per attribute. Without hashing the validation and

Tool	*CPU (%)*	*MEM (%)*	*File Size (MB)*
Wireshark	24.1	15.6	11.7
tshark	10.3	8.6	11.7
Aletheia Edge	<0.1	<0.1	2.34

TABLE I: Resource utilization on the edge to capture medium information for the 2 most common tools compared to Aletheia. Aletheia requires less than 200X CPU utilization of wireshark and less than 100X of tshark; it also consumes 5X less memory, which is crucial as the example of logged data sizes are within 1 minute intervals.

repeating the checks, we find that the CPU utilization and memory of Aletheia can spike up to 5% and 7% respectively (slightly less than tshark); with hashing answers to prevent repeated checks, the CPU utilization drops back to 0.3% which we deem acceptable.

V. Use Cases

We demonstrate the value of Aletheia in three cases. 1) *Proper Network Categorization:* we show that a common practice of categorizing test environments as either "homes" or "offices" is questionable, because the intrinsic assumption of low/high traffic at home/offices is invalid. Instead, we recommend using Aletheia to reliably quantify the medium utilization to truly categorize the network condition. 2) *Enterprise Network Troubleshooting:* we demonstrate that Aletheia can troubleshoot networks by identifying misconfigurations and heavy medium consumption by non data traffic. E.g., a single misconfigured access point that keeps sending CTS-to-Self causes other nodes to back off, thus yielding poor network performance. Through Aletheia, we were able to quantify beacon medium utilization of misplaced access points in enterprise buildings to be as high as 30–40%. 3) *Impact of medium utilization on latency and loss:* we study how medium utilization affects transmission latency and packet loss. We observe strong correlation where the latency and loss can increase greatly (e.g., triple or more) when medium utilization shoots up. This further reinforces the importance of knowing the medium utilization when measuring protocol performance. Otherwise it is hard to know whether it is the design or background traffic that caused performance issues.

A. Proper Network Categorization

In both academia and industry, it is common for a networking system to be tested under different environments to evaluate its performance when facing different background traffic. It is also common to test a system repeatedly under the same, stable network condition to exclude the impact of background traffic variations (or average results over time). We show how these tests can benefit from Aletheia.

1) Office networks are not necessarily busier than home. To test a system under different background traffic, researchers tend to repeat the test in different physical locations. A common practice is to classify environments into a few categories (e.g., Home/Office/Hospital/Library) and empirically use homes as representatives for low-utilized environments and offices/cafes for high-utilized ones [16], [17], [18], [19]; we find that such practices are questionable.

Fig. 3 shows 24-hour medium utilization for four locations (two homes, two offices). We observe a home environment (Home2) congruent with the assumption of lower traffic than offices (0–15% vs 15–55%). However, another home (Home1) has higher medium utilization (50–75%) than both offices. Thus, relying on location type to assume its medium utilization is incorrect. Modern home environments contain smart devices and multiple WiFi based devices that stream multimedia continuously, which can utilize more medium than some offices. Without rigorous quantification, researchers may incorrectly attribute observations in performance, good or bad, to their designs instead of background traffic. Using Aletheia, one can precisely measure the medium utilization from their protocol and background traffic, thus obtaining correct insights for improving the design.

2) Low correlation between numbers of devices/networks and medium utilization. We look further into the logs of all four environments, and we do not find high correlation between the number of devices or networks and the medium utilization. We find that going from lowest to highest utilization (Home2, Office2, Office1, and Home1) that minimum–maximum numbers of devices per environment are (6–13, 18–40, 10–30, 19–60) and of networks are (3, 9, 3, 11), respectively. We find that correlation is too low (0–0.2) to draw conclusive results. When more networks available more periodic beacons are transmitted. However, there are also cases with other network modes (e.g., ad-hoc) where every single node sends periodic beacons most of the time. Thus, relying on numbers of devices or networks does not reliably categorize the network.

3) Medium utilization does not remain stable in the same environment within short durations. To understand the impact of design parameters, factors such as dynamic background traffic are usually excluded. Often experiments are repeated in the same environment over hours and/or days, with the hope that the medium is somewhat stable or has some form of pattern, thus allowing results to be normalized over a long duration; Our results show this approach to be problematic.

Fig. 3 shows constant and sometimes large changes in medium utilization for all four environments. Minute to minute, hour to hour, or sudden changes of 20% or more happen multiple times. We further examine the medium utilization in a finer granularity. However, frequent and significant changes happen under different time scales. Multiple access points with different beacons, data rate setups, and many users all contribute to the dynamics. With the help of Aletheia, the user can identify in what durations a network is relatively more stable, and results within such periods may present a more reliable observation of the performance of designs.

4) The dominant causes of medium utilization can vary depending on the environment. Fig. 3 shows that human presence (e.g., Home1) can add up to 25% medium utilization depending on the time in busy environments, while the system's overhead regardless of human impact is above 50% utilization. We believe this is mostly due to the inherent design of existing technologies. Existing WiFi systems

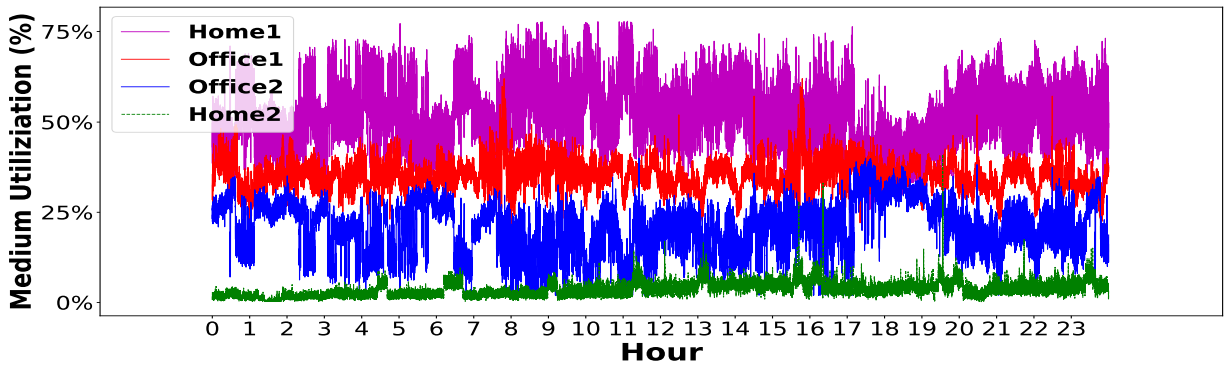
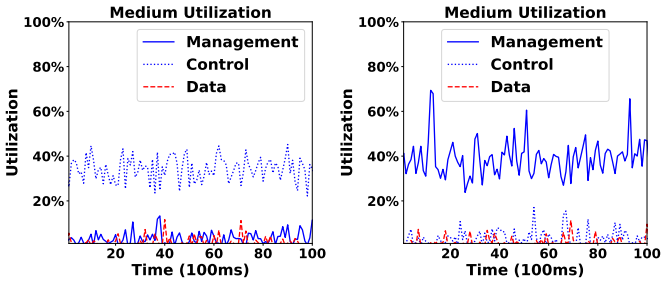


Fig. 3: 24-hour medium utilization of four different environments. Home1 utilization varies between 50–75%, followed by Office 1 (25–50%), Office 2 (5–35%), and Home2 (0–25%). Further, the utilization within the hour can vary by 25-30% in the same environment (e.g., Home1).

in most environments have automatic updates (e.g., smart-TVs, computers), and systems mostly left connected and turned on regardless of human presence. However, "empty" environments (e.g., Home2) are affected by human presence. Thus, these observations depend heavily on the environment analyzed, further demonstrating the need for environmental measurement using Aletheia.

B. Enterprise Medium Troubleshooting

Through Aletheia, we identify a few culprits causing poor network performance, which result from either WiFi device misconfiguration (because normal users are non-experts) or certain WiFi mechanisms with room for improvement.



(a) CTS-to-Self (b) Beacon

Fig. 4: Cases when CTS and beacon frames are the major medium consumer, due to device misconfiguration or WiFi's current imperfect mechanisms.

1) One WiFi misconfiguration can impact many users. We evaluate an area in an office building that is known to have slow WiFi connection (all access points are slow), even though all access points are nearby and have strong signals. We decided to use Aletheia and see if the issue is related to medium utilization before doubting hardware/firmware issues with the access points. In Fig. 4(a), we see the average medium utilization in the area is 30–50% on average. Using Aletheia's flexible filtering and tagging features, we observe that 80% of the frames sniffed are CTS-to-Self broadcast frames. There was a rogue access point sending periodic CTS-to-Self without sending any data, which made all nodes back off thus allowing little data transmission. The CTS frames were not the fundamental cause of latency, but the requests of backoff duration within the CTS frames. Through Aletheia, we were able to find out what the exact cause of the problem was, identify the access point (router) causing the issue, and fix the configuration.

2) Certain WiFi mechanisms need improvement. In another region with average medium utilization, we do not observe any misconfigured systems, but an issue from the 802.11 MAC layer design. Periodic beacons are sent from access points or nodes in ad-hoc mode. Beacon frames are of small sizes (90–500 bytes) by design to avoid consuming excessive medium. However, when many access points exist, which is common in enterprise environments, the accumulation of many beacons over a region can lead to significant overhead. Fig 4(b) shows management frames consuming on average 40% of medium; among those, 95% are beacon frames. Further examination shows there are only 18 users and they do not produce much data traffic. In the current WiFi design, even though little data transmission occupies the medium, 40% of the medium is already taken just for maintaining the networks. We believe this issue warrants further research and improvements.

C. Medium Utilization Impacts Performance.

We look into how quantified medium utilization affects performance of a transmission (i.e. latency and loss rates) by monitoring data sent between two nodes. We setup 2 Raspberry Pis running in ad-hoc mode, one as a receiver and the other transmitter. We keep Aletheia running while the transmitter broadcasts and add a specific *seq* field in broadcast application layer payload to know the order among frames.

We observe that the latency of broadcast can triple (0.3s to 0.9s) due to variations in medium utilization. Higher overall medium utilization leads to longer latency. Higher utilization causes more and longer backoffs, thus more time for transmission. Further, there is a strong correlation of medium utilization stability (more than absolute medium utilization) and loss: the more stable the medium utilization (at 100ms granularity), the less the frame loss. The more unstable (i.e. change in utilization in 10 ms time granularity), the higher the loss. This is due to CSMA/CA being unable to sense changes since the algorithm heavily relies on sensing at the current moment. Thus, if the radio senses the medium is

empty, it will transmit, which leads to high chances of collision in varying medium utilization networks. Less stable medium utilization implies more sudden transmissions, thus yielding higher chances of collisions and therefore losses.

Aletheia can identify three causes of frame losses from logs of scanners, senders and receivers: *i)* hidden terminals where there is transmission from senders but no frame reception at scanners due to collisions preventing the PHY from decoding data, *ii)* wireless interference where there is frame reception at scanners with corrupted CRC, but no reception at receivers because their PHY layer drops such corrupted frames, and *iii)* bugs where scanners receive frames correctly but not receivers.

VI. Discussion

Running Analyzer on Edge. Our *analyzer* code can run on the edge or cloud. Our code can be extended to support parallel processing in the cloud for large scale monitoring, or run on edge devices with limited computing resources to generate graphical results of stored data, albeit generally slower than in the cloud.

Offline SAE. We support a mode of storing all data through *pcap* (e.g., wireshark) and processing the file offline to extract headers of interest then passing it to *analyzer* to get results. This enables users to process stored files in multiple different ways by extracting different fields of interest and re-runningthe *analyzer* to get quick results.

Code Release. We have released Aletheia's source code on GitHub at the following link: https://github.com/SBU-MoCA/Aletheia-WiFi.

VII. Conclusions

In this paper, we introduce Aletheia, a lightweight open-source medium utilization measurement tool that quantifies wireless medium utilization at microsecond granularity using low cost commodity hardware. We describe its selective attribute extraction and its tag and view features. We demonstrate the value of the tool in network categorization, enterprise medium debugging, and understanding latency/loss using 24 hours logged in four different environments. We find that practices commonly employed in wireless experiments are questionable, and identify problems in wireless misconfiguration and current WiFi design. The tool offers medium analysis that is indispensable to distinguish the impact of background traffic from those of design parameters, thus gaining the correct understanding of design choices.

VIII. Acknowledgment

We thank anonymous reviewers for their invaluable feedback. This work is supported in part by NSF grants 1513719 and 1730291.

References

[1] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging wifi-enabled iot," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2017, pp. 1–10.

[2] N. C. Luong, D. T. Hoang, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Data collection and wireless communication in internet of things (iot) using economic analysis and pricing models: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2546–2590, 2016.

[3] H. Yan, Y. Zhang, Z. Pang, and L. Da Xu, "Superframe planning and access latency of slotted mac for industrial wsn in iot environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1242–1251, 2014.

[4] H. S. Jang, H. Jin, B. C. Jung, and T. Q. Quek, "Versatile access control for massive iot: Throughput, latency, and energy efficiency," *IEEE Transactions on Mobile Computing*, 2019.

[5] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel *et al.*, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.

[6] U. Lamping and E. Warnicke, "Wireshark user's guide," *Interface*, vol. 4, no. 6, 2004.

[7] M. Kershaw, "Kismet wireless, http://www. kismetwireless. net," *Retrieved from the Web*, 2007.

[8] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage, *Jigsaw: Solving the puzzle of enterprise 802.11 analysis*. ACM, 2006, vol. 36, no. 4.

[9] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren, "Analysis of a mixed-use urban wifi network: when metropolitan becomes neapolitan," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 2008, pp. 85–98.

[10] A. Bhartia, B. Chen, F. Wang, D. Pallas, R. Musaloiu-E, T. T.-T. Lai, and H. Ma, "Measurement-based, practical techniques to improve 802.11 ac performance," in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 205–219.

[11] K. LaCurts and H. Balakrishnan, "Measurement and analysis of real-world 802.11 mesh networks," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 123–136.

[12] B. Agarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker, "Netprints: Diagnosing home network misconfigurations using shared knowledge." in *NSDI*, vol. 9, 2009, pp. 349–364.

[13] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing wireless packet losses in 802.11: Separating collision from weak signal," in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 2008, pp. 735–743.

[14] V. Jacobson, C. Leres, and S. McCanne, "Tcpdump/libpcap," *Accessed: Jun*, vol. 23, p. 2016, 1987.

[15] R. Header, "Radiotap header for ipw2200, website 2006."

[16] M. Heidari and K. Pahlavan, "Performance evaluation of wifi rfid localization technologies," *RFID Technology and Applications*, p. 74, 2007.

[17] A. Kostuch, K. Gierłowski, and J. Wozniak, "Performance analysis of multicast video streaming in ieee 802.11 b/g/n testbed environment," in *Joint IFIP Wireless and Mobile Networking Conference*. Springer, 2009, pp. 92–105.

[18] D. Turner, S. Savage, and A. C. Snoeren, "On the empirical performance of self-calibrating wifi location systems," in *2011 IEEE 36th Conference on Local Computer Networks*. IEEE, 2011, pp. 76–84.

[19] F. Domínguez, A. Touhafi, J. Tiete, and K. Steenhaut, "Coexistence with wifi for a home automation zigbee product," in *2012 19th IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*. IEEE, 2012, pp. 1–6.