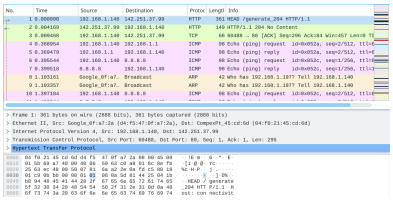
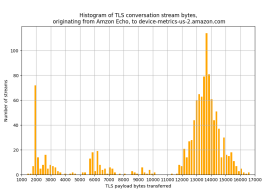
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

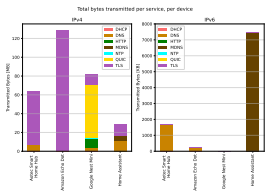
Excel@FIT 2022
http://excel.fit.vutbr.cz

IoT Gateways Network Communication Analysis

Jan Zbořil*







Abstract

Modern IoT gateways are mainly developed by private companies behind closed doors. This results in a closed ecosystem, where only a small amount of information about traffic is available to the public. Therefore, to gain knowledge regarding the operation and communication of such gateways, it is necessary to examine and analyse network traffic flowing to and fro such gateways. This paper's primary goal is to capture and process network traffic data of multiple commercially available gateways intended for home use, analyse their communication behaviours, compare the results to other studies carried out in this area, and discuss possible attacks on used gateways based on gathered data. Communication data were obtained by deploying a controlled environment and analysed using Zeek, together with Wireshark software. Gathered communication data can be further used by researchers in the areas of networking or security.

Keywords: IoT — IoT gateways — Network Traffic Analysis — Attacks on IoT Devices

Supplementary Material: Captured traffic dataset (Google Drive)

*xzbori20@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Since the mid 2010s, there has been a boom in sales and use of Smart Home devices and gateways. IoT Gateways are for interconnecting the IoT devices between each other, and to provide them the connection to the Cloud. Nowadays, many gateways can also serve as a device (smart speakers). Because of this soaring use, it is crucial to make research in this area of IT.

Many major IT companies successfully implemented IoT for home use to be a part of their business model. Amazon started selling their gateways in 2014, while Google followed in 2016 [1].

Since selling these devices generates revenue for their manufacturers, such gateways were made closed sourced, and they lack transparency. This creates a problem. No one, who is not directly participating in the design of such devices, can directly observe how these devices work. The academic sphere can partly solve this issue by being a counter-balance, and it can point out the security vulnerabilities of IoT gateways.

For the purpose of this research, IoT gateways were chosen based on their general consumer availability and popularity. Each gateway was integrated into the environment, and a set of tasks were carried out. Network traffic was captured and stored during the runtime of the tasks. The resulting data were filtered and analysed using Zeek and Wireshark, using a sheet editor for easier manipulation with a large amount of data. The main concern of data analysis lies in processing the DNS queries and data streams flowing to and fro the gateways.

This paper aims to assess the state of security of IoT gateways, what attacks can be directed against them, if the fingerprinting of the gateways can be deducted from the traffic, et cetera. A publicly available

dataset consisting of network traffic of four widely used IoT gateways is one of the results of this work. The other result is an analysis of this dataset, which highlights interesting patterns and traffic behaviour and provides a direct comparison between tested devices. It clearly shows distinct features which can be used for device fingerprinting. This work confirms and disproves some results from the aforementioned studies.

2. Related Work

Researchers have already carried out studies in the field of the traffic and security of IoT gateways. Amar et al. [2] built out an environment consisting of many devices and gateways, and captured the traffic for 22 days. Their work explains the learned device characteristics or other non-expected results, especially in the way the devices were set up and how they operated. They talk about global statistics, including total bytes sent per device, per protocol, etc.

Ivan Cvitić et al. published a research [3], in which they successfully sorted IoT devices into several classes based on the characteristics of their network traffic flows. The paper also include devices communicating via a Zigbee protocol, among those using Ethernet or Wi-Fi. Based on the coefficient of variation of received and sent data (C_u index), and data transformation, in order for the data to behave like a normal distribution, so statistical tests for this particular distribution could be used; they were able to link the device to one of four classes of devices sharing similar behaviour.

In their research [4], Pierre-Marie Junges et al. used captured traffic analysis for inference of user actions, such as turning the smart light on or off. They took the position of an outsider, looking into the traffic between the LAN and the cloud. They noted that identification of the devices, especially from the TLS handshake, can be extracted from such traffic. First, they identified the problems of deducting information from the traffic. They claim they are the following: no individual IoT device signature, gateway abstraction, encryption. They also made several assumptions on which their later actions are based: sending actions to the IoT devices-actions in one command are sent as one; incidence of the actions on the packet size, command size stability and data structures similarity. They then captured network traffic while operating multiple IoT devices, using combinations of different actions on various devices. After measuring the size of datagrams from the start of captured TLS streams and doing computations, they were able to distinguish the correct operation taken with 98.4% accuracy. However, the authors rely heavily on the assumption that the IoT device sends traffic to the cloud after an action inside the network is taken, which is not always true, as is later revealed in this work.

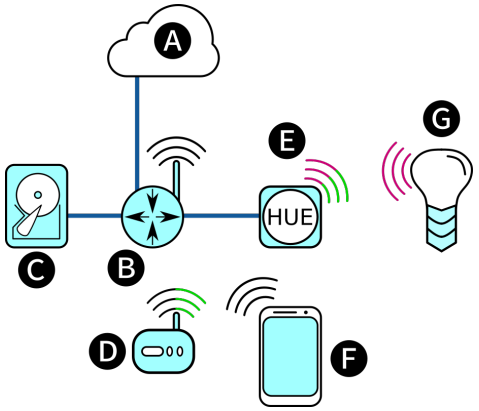


Figure 1. The Figure demonstrates, how the devices, which the LAN was composed of, were connected. A – The cloud; B – Turris router; C – External SSD for storing captured files; D – tested IoT gateway; E – Phillips Hue Bridge; F – Smartphone with control application; G – Phillips Hue light bulb. Wireless communication between the devices is colour coded.

3. Setup and Methodology

This chapter shows the process of selecting the gateways to be tested and the setup of the environment, and it describes the methodology of data capturing and analysis.

3.1 Gateways selection

The gateways to be tested were the following: The Aeotec Smart Home Hub, Amazon Echo Dot 4th gen., Google Nest Mini and Home Assistant (HA) software gateway running on a Raspberry Pi 4. They were chosen based on general consumer availability and popularity. All gateways, except the HA, are closed sourced. The HA was specially chosen for its open-sourced nature to compare, how such different philosophies compete.

3.2 Environment Setup

IoT gateways were tested separately. The LAN consisted of a Turris MOX router, serving as a default gateway. The Samsung external SSD, intended to store the captured pcap files, was connected to the router via a USB port and mounted as an EXT4 volume—Turris router is based on OpenWrt Linux distribution [5]. A Phillips Hue smart light bulb was wirelessly connected to the Phillips Hue Bridge, which was plugged into the Turris router using Ethernet. Each of the tested

IoT gateways was connected to the Turris router via Ethernet, where applicable, or via Wi-Fi. Each gateway was set up using its own Android mobile application. The entire setup is illustrated in Figure 1.

Due to the lack of knowledge of device compatibility and the validity of the used analysis methods, it was decided to test each device separately. BBuilding the environment this way does not represent a real-life network, where various devices are connected simultaneously. However, it isolates possible deviations that might occur only when the tested IoT gateway communicates with a specific device. Another point is that such a complex environment would be challenging to recreate.

3.3 Capturing methodology

Traffic for each IoT gateway was captured in two operation modes-active and passive.

The Phillips light bulb was repeatedly turned on and off ten times in a row during the active capturing. This capturing was done to discover traffic, which could be associated with the actions taken.

Passive capturing mode consisted of leaving the gateway running idle for seven days, in which there was no direct, intentional interaction with it. The gateways were running in the standard university office, where a conversation and other noises happened, which could impact the collected data.

3.4 Methodology of Data Analysis

After the data had been collected into pcap files, records of streams, DNS, and other information were extracted using the Zeek command-line tool [6]-an open-source network security monitoring tool.

Zeek logs were copied into a spreadsheet editor to ease the manipulation and filtration of data. DNS logs, including queries, answers, intervals between queries, and the number of such queries, were documented. Valuable and compelling information was gained from both TCP and UDP streams. These discoveries contain an amount of data transmitted, significant differentiating attributes, encryption information, the suspected purpose of the streams, et cetera. Where applicable, a graph, diagram, or conversation log has been made. The Wireshark software was used for gaining detailed information and fact-checking Zeek logs.

4. Analysis results

Results of the analysis are located in this chapter. They are grouped based on the device, with information learned from the passive mode written first.

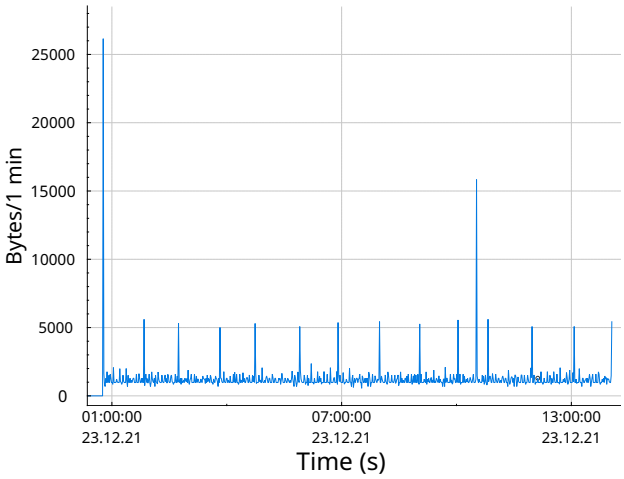


Figure 2. Input/Output graph of the stream from the Aeotec gateway towards the dc-eu01-euwest1.connect.smartthings.com endpoint. X-axis step is 1 minute. All the spikes, seen in traffic, originates from the server side.

4.1 Aeotec Smart Home Hub

Out of 89.9 MB transmitted by Aeotec gateway, 68.7 % were TCP, while 55.1 % were TLS. In terms of packets, TLS contributes to 28.6 % of all packets. UDP makes only 9.8 % of all transmitted bytes and DNS 6.2 % of bytes. Aeotec sent 39 429 DNS requests, but all of them were asking only for three queries. These were: api.smartthings.com, fw-update2.smartthings.com and dc-eu01-euwest1.connect.smartthings.com.

Most connections were held against api.smartthings.com (14 446). This TLS 1.2 encrypted stream ran periodically every minute during the whole week. The connection towards the dc-eu01-euwest1.connect.-smartthings.com endpoint ran twice. The first stream was already running when the capturing began, and it lasted 6.5 days into the capture period. The second stream followed immediately after the first one ended. The streams were idle most of the time, with only TCP keep-alive packets being sent. The I/O graph of the second of these streams is shown in Figure 2.

All endpoints used by Aeotec hub were hosted on Amazon's AWS servers. Aeotec Smart Home Hub did not use NTP for time synchronization purposes.

Light Bulb Control

One continuous TLS stream was detected during the live capturing, which could have been sending commands toward the cloud. Here, the sizes of frames are main determinants. There are two possible patterns of packet streams, which could had been sending the data towards the cloud at the dc-eu01-euwest1.connect.smartthings.com endpoint. Both are demonstrated in Table 1. In the first case, it is assumed that there are six packets sent for one operation and not all the pack-

Scenario 1		Scenario 2	
Direction	Size	Direction	Size
out	113 B	out	113 B
out	490 B	out	490 B
in	113 B	in	113 B
in	318 B	in	318 B
out	113 B		
out	490 B		

Table 1. The Table shows the possible patterns in packets sent from (out) and to (in) the Aeotec Smart Home Gateway when operating the light bulb.

ets were captured (captured pcap ended sooner). The second scenario assumes that not all the packets in the stream are directly connected to the light bulb operations.

4.2 Amazon Echo Dot, 4th gen.

Amazon Echo (AE) queried for 23 different DNS records (the most of all gateways). All of these were for A records, while most of these requests (2014) queried for the d3p8zr0ffa9t17.cloudfront.net domain name. All DNS queries from AE can be seen in Figure 3.

Every 5 minutes, the AE sends bursts of 10 ICMP Echo messages towards the LAN's default gateway. The AE gateway uses an NTP service for its time synchronization. NTP conversations occurred 31 times.

The AE communicated with the local Phillips Hue bridge, d3p8zr0ffa9t17.cloudfront.net, acsechocaptiveportal.com and fireoscaptiveportal.com using a plain HTTP. The status code of all responses from the last-mentioned endpoint always arrived in two packets, with HTTP codes 204 and 400.

AE participated in a total of 17 distinct TCP/TLS streams. One was directed towards the 65.9.90.59 IP address (DNS answer for d1s31zyz7dcc2d.cloudfront.net). It lasted 42 seconds, and it transmitted 105 MB inbound. It can be assumed, with confidence, that the gateway pulled a software update using this stream.

Stream, which repeated mostly—3 047 times, communicated with api.amazonalexa.com. These streams were short, with an average length of 0.37 seconds. They used multiple IP addresses, presumably for load balancing. All conversations are encrypted using TLS 1.3 or TLS 1.2. TLS 1.3 conversations had client hello PDU padded to 583 TLS bytes.

Conversations between the Amazon Echo and softwareupdates.amazon.com happened eight times. Intervals between streams range from 00:15:33 to 1 day and 01:35:01. These streams do not download software updates, as the endpoint's name may suggest (small amount of data transferred).

Connections to the device-metrics-us-2.amazon.com endpoint ran 1 204 times. The endpoint's name suggests the sending of metrics data.

There was no clear conversation between AE and the cloud during the active capturing, which might have transmitted the commands taken to the cloud.

4.3 Google Nest Mini

Google Nest (GN) sent a total of 33 887 DNS queries, 20 for A or AAAA, 1 PTR and 5 MDNS queries. The GN used Google's own DNS (8.8.8.8). Other devices used the address given via DHCP. This one was used by GN only when queries at 8.8.8.8 failed. IP addresses and domain names overlap—a server hosts multiple services on different domain names.

The GN used NTP for the time synchronization. Endpoints for NTP were time[0-4].google.com. A maximum of 20 minutes passed between NTP synchronization.

The gateways checked periodically for connectivity to the default gateway and DNS server. These ICMP echo messages were sent in batches of two requests for each endpoint.

Interesting conversations happened on the UDP port 10101—serving as both source and destination port. 224.0.0.250 and 239.255.255.251 were the endpoint's multicast IP addresses. Time-to-live of packets for these streams was set to four. This is not the default multicast TTL value.

The GN was the only tested gateway that used the QUIC protocol. Six different communication groups utilizing the QUIC protocol were observed. Google developed QUIC (Quick UDP Internet Connections) protocol, and it is meant to reduce the overhead and latency of TCP while keeping its benefits. QUIC is trying to implement TCP, TLS and HTTP on UDP (more in RFC 9000 [7]).

The QUIC streams were to the tools.google.com endpoint—406/407 streams had 216.58.201.78 address as the destination. The typical conversation consisted of client hello → rejection → client hello → connection. It was protected using TLS 1.3. The authentication algorithm used was AES-GCM, and Curve25519 was used for key exchange. TLS handshake packets were padded to the same length. The next stream, which was repeated 92 times, had www.google.com as its endpoint. TLS parameters were the same as for the previous stream. 471/555 frames (84.86 %) carried either 33 B or 1 350 B of payload. Intervals ranged from 00:01:29 to 11:10:03. The rest of the QUIC streams had the same characteristics as the aforementioned did.

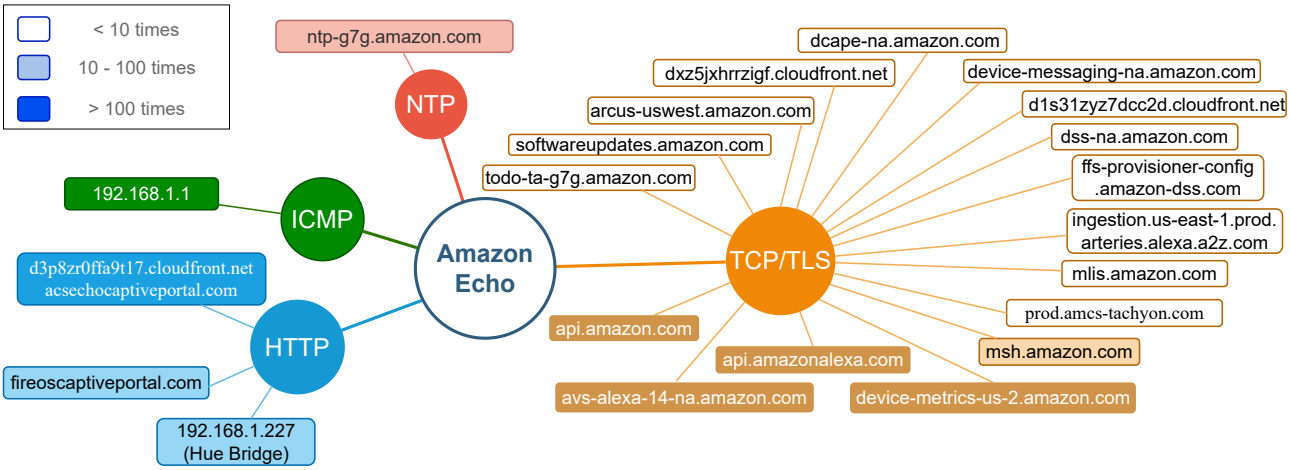


Figure 3. Map of endpoints to which the Amazon Echo established connection. Connection type is differentiated by colour. The Colour shade marks the number of connections to the endpoint.

The GN also used the TCP/TLS protocol. While the average stream duration, when using the QUIC protocol, was less than 1 second, communications via TLS were much longer—2 113.5 seconds on average. Other TLS parameters were the same as when using QUIC.

All streams observed during the live capturing were also found in the passive one. They do not seem to carry any data linked to switching the light on/off.

4.4 Home Assistant

The Home Assistant sent A or AAAA DNS queries for six endpoints. This is the second-lowest number, after Aeotec, in tested gateways.

Querying for an IP record of cognito-idp.us-east-1.amazonaws.com was made to gain a way to authenticate itself towards Amazon's servers. Quote Amazon AWS [8]: "Amazon Cognito lets you easily add user sign-up and authentication to your mobile and web apps." This is presumably used in order to get to cloud.nabucasa.com, which provides cloud solutions for Home Assistant and is served by Amazon AWS.

DNS queries for www.home-assistant.io and analytics-api.home-assistant.io were answered with the same range of IP addresses. The overlap is demonstrated in Figure 4.

An intriguing observation not experienced elsewhere was scanning the local area network using the PTR DNS record. The Home Assistant sent batches of 254 PTR queries for the entire network with a /24 prefix every hour.

Home Assistant used Cloudflare NTP servers to get current time information. The endpoint, to which it connected 295 (interval between streams was always 00:34:08), was time.cloudflare.com.

There were 2 889 HTTP communications towards Cloudflare hosted endpoints without registered domain

Figure 4. Diagram showing the running of multiple services on one endpoint—all three domain names under the *.home-assistant.io wildcard domain share the same IP address. Non overlapped relations are dismissed.

names.

The Home Assistant connected to github.com 112 times, always with two TLS streams starting simultaneously. Conversations with the cloud.nabucasa.com were running during the whole week, divided into two distinct TCP/TLS streams. The first stream started before the capturing had begun, and it ran until February 18th, 20:20:03 CET. The second ensuing conversation began on the same day, at 20:20:37 CET and lasted past the end of packet sniffing. The two streams manifested vastly different behaviour.

The Home Assistant did not communicate with the cloud when operating the lights. However, a conversation with a smartphone was observed. Twenty packets with TCP data and PSH flag, either 79 B or 80 B in length, were sent from the phone. There was no difference between turning the lights on or off (smartphone as a source). Fifty-seven packets with TCP ACK and TCP PSH flags were sent back to the smartphone. Responses were either 2, 3 or 4 packets long. The lengths of these packets were not unified. The lengths were contained in an interval of 13 B and 105 B of TCP payload. The first response packet usually arrived 50-60 ms after the request, with the next packet sent from the endpoint after the next 20 ms. In the case of the

fourth packet being sent, the interval after which it was sent depended on the type of action taken. For the “lights on” operation, the fourth packet was sent approximately 1 second after the third. In the case of ”lights off”, the fourth packet followed immediately after the third one.

4.5 Global Results

In total, the capturing lasted for a total of 28 days. More than 4080 MiB in 6 579 433 datagrams of raw data were captured, later filtered to 516 MiB and 2 278 689 packets of data used for the analysis. Most filtered data was sent and received by Amazon Echo —221.6 MiB or 54 %; with Google Nest Mini coming second with 117.0 MiB of data. The Aeotec gateway transmitted 93.7 MiB of filtered data. Raspberry Pi came last, with 83.5 MiB of data sent or received.

Figures 5 and 6 show the total amount of data transmitted by the transport layer protocol (5) and by application layer protocol (6).

This traffic analysis showed how much **information can be obtained from the traffic**, even when **encrypted using modern ciphers**. Based on this analysis, the devices running inside the network can be **clearly identified from the data stream (device fingerprinting)**, even when observed from behind the local network. The usage and distribution of protocols, the amount of data sent, and the specific patterns in each of the streams or between the streams can serve as a basis for device distinction. This is valuable, especially since that the MAC address of a device—the main and the most accessible identifier—is lost at a network’s border.

4.6 Learned information and Other Studies

The methodology of capturing and analysis process of this work was based on the one of Amar et al. [2]. This was done to achieve the most comparable results. From the data of Amazon Echo, it can be deduced that **traffic in this paper and Amar’s paper differ, yet they share certain similarities in other areas**. The cause for this could be either using a different version of the Amazon Echo device, different settings of the device, or simply the time difference between the two studies—during which Amazon may have changed internal processes. The **conversations of Amazon Echo in this research were mainly composed of TLS** traffic (see Figure 6), while Amar showed that their Echo also communicated via ICMP and other protocols. The Echo in this research did not send as many DNS requests, and especially it did not send any requests for DNS resource records such as www.example.net, compared to Amar’s work. Also, NTP connections happened less often in this work than in the paper of Amar. **Between publishing Amar’s work and this paper, the general use of TLS rose**. This is a positive discovery; however, the low number of tested devices could skew the result, and it may not represent the situation in the entire IoT segment. This work differs from the Amar’s mainly in the type of devices tested—this study is concerned with IoT gateways only. Also, the results of this paper emphasize more detailed characteristics of streams than Amar.

The dataset collected in this work is similar to the one featured in the paper of Ivan Cvitić et al. [3]. It could be further used to validate their research if the same process was carried out on the dataset. The work was done the same way or similarly until the point following the extraction of stream data. Different tools were used for the part of streams data extraction.

Junges et al. [4] used traffic flow data to create a tool which can identify user taken IoT actions inside a network with 98.4% accuracy. The tool relies on several assumptions and solutions to challenges. **Their assumption that gateways send data after command execution is not always correct, as the results of this paper manifest. The Amazon Echo and Google Nest Mini do not indicate that traffic is generated when operating a light bulb. Therefore, the tool created by Junges et al. would be unusable for these gateways**.

5. Conclusions

In this research, an analysis of the network traffic of several IoT gateways intended for home use was conducted. This analysis demonstrated that **valuable information about the devices running inside a local network could be obtained by observing traffic** leaving and entering the network. **The fingerprinting of the devices, based on the traffic flow, is certainly possible.** Most information can be gained from DNS queries and answers. Due to the extensive use of TLS encryption for HTTPS, traffic information can be extracted from parameters such as packets lengths, patterns found in the traffic, destination addresses and their DNS records, et cetera. **However, the exceptionally high use of TLS is an excellent sign in the grand scheme of things**, meaning that manufacturers are not underestimating the importance of data security and confidentiality.

The dataset, consisting of captured traffic, is openly shared and available. Its use for future research can be another contribution of this paper. Future research, consisting of machine learning, neural networks, et cetera, can be trained and used on this dataset, to find

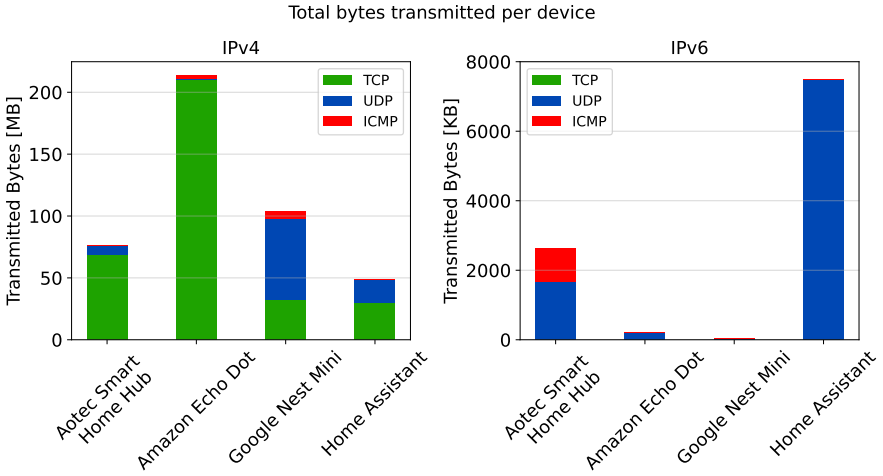


Figure 5. The Figure presents the total amount of transferred data. based on transport layer protocols + ICMP

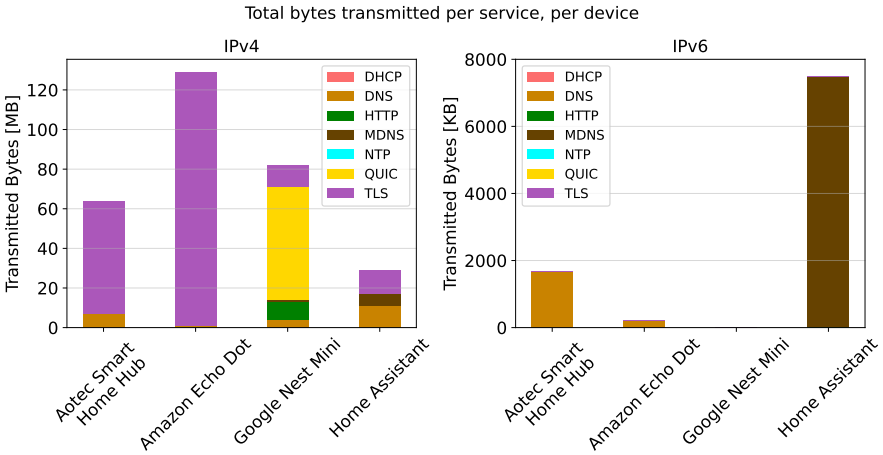


Figure 6. The Figure manifests the total amount of transferred data based on application layer protocols.

or confirm the emerging data patterns presented as the results of this work.

Acknowledgements

My gratitude belongs to Mgr. Kamil Malinka, Ph.D. and Ing. Ondřej Hujňák.

References

[1] Ava Mutchler. A timeline of voice assistant and smart speaker technology from 1961 to today. online, March 2018. https://voicebot.ai/2018/03/28/timeline-voice-assistant-smart-speaker-technology-1961-today/.

[2] Yousef Amar, Hamed Haddadi, Richard Mortier, Tosh Brown, James Colley, and Andy Crabtree. An analysis of home iot network traffic and behaviour, 03 2018. DOI: 10.48550/ARXIV.1803.05368. https://arxiv.org/abs/1803.05368.

[3] Ivan Cvitić, Dragan Perakovic, Marko Periša, and Mate Botica. Definition of the iot device classes based on network traffic flow features. In *4th EAI International Conference on Management of Manufacturing Systems*, pages 1–17, 01 2020. ISBN: 978-3-030-34271-5.

[4] Pierre-Marie Junges, Jérôme François, and Olivier Festor. Passive inference of user actions through iot gateway encrypted traffic analysis. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 7–12, 2019. ISBN: 978-3-903176-15-7.

[5] CZ.NIC, z. s. p. o. Turris mox. online, 2021. https://www.turris.cz/cs/mox/.

[6] The Zeek Project. Zeek documentation. online, April 2022. https://docs.zeek.org/en/master/index.html.

[7] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021. https://www.rfc-editor.org/info/rfc9000.

[8] Amazon Web Services, Inc. Amazon cloudfront. online, 2022. https://aws.amazon.com/cognito/.