Inference of User-Facing Packets from Network Traffic

Marina Lucena, Francesco Bronzino, Renata Teixeira

Inference of User Facing Packets from Network Traffic

Marina Leao Lucena[1], Francesco Bronzino[1], and Renata Teixeira[1]

[1]Inria, Paris

July 21, 2017

Abstract

This thesis presents a technique to determine whether a user is actively interacting with a device from inside the network. By the observation of user's traffic, network operators can use the detection method to improve traffic management by prioritizing user facing flows.

Understanding whether a user is using a device from inside the network requires separating user generated traffic from the one that is automatically generated by a machine in the background. To accomplish this, we apply three filtering methods, looking at request URLs and at the inter events distance and periodicity. Using real world data from eight devices in the UK collected with a tool capable of recording not only network traffic, but also user input, we show how our method separates the traffic generated by the user.

1 Introduction

The last decade has seen a drastic popularization of access to Internet applications and services. Accessing the Internet from a computer or a smartphone has become a routine task for most people and interactions with these devices can vary from very short bursts of usage to long online sessions. Moreover, the list of connected devices that populate our households is ever growing. While the amount of data that traverses our home networks keeps increasing every day, only a subset of it can be associated with a user directly interacting with online services.

Understanding whether it is possible to isolate and infer user online presence within the network is useful in many fields. From a security standpoint, the detection of user online presence could constitute a serious threat, where malicious attackers might exploit this information to understand when users are at home. On the other end, detecting user presence can be extremely valuable to network administrators in two main fields: performance optimization and network diagnosis and maintenance. If capable of detecting which packets correspond to the current user-facing activity, network administrators could instrument routers in homes to prioritize the subsets of traffic most affecting the user experience. Moreover, if the administrator detects any problem affecting the proper functioning of the network, he can prioritize solving the problems associated with an active user, who is currently online.

This thesis analyzes how to detect which packets in a stream of packets traversing a router originate from user-facing application. We define *user-facing traffic* as network traffic generated by the user, by accessing a website, streaming a movie, or listening to music, for instance. The core challenge of detecting user-facing traffic is to be able to differentiate user online activity from background traffic activity. In fact, many applications run on a user's device but on the background without user intervention, so only a fraction of the packets from an individual's device correspond to data generated by the user herself.

We create a method that takes as input a stream of network packets and identifies the subsets that corresponds to the user-facing traffic. In order to do that, we develop filters that run in real time. The groups of packets towards a given destination will then be filtered out by three different methods in sequence: (i) removal of all packets to/from hosts in a pre-defined blacklist, (ii) elimination of packets with inter-arrival times of less than a second and for last (iii) elimination of packets whose intervals are too constant (meaning the same interval is repeated many times) and so, have a high probability of being generated automatically by a website or software.

We develop and evaluate our method based on two months of data collected from a total of eight participants from the UK in 2015 and 2016. The study monitored participants and stored information related to their network traffic such as network flows, DNS and HTTP requests and location of usage, along with information related to user device interaction, such as mouse and keyboard clicks and information about screen state of applications (fullscreen or idle). More specifically, from the data obtained, we analyze HTTP and DNS requests and filter out any packets that are not produced directly by the user, equivalent to background activity. The choice of using HTTP and DNS requests comes from the fact that this was the data less affected by noise during collection, other datasets had many missing packets. Using data such as applications on fullscreen mode and mouse clicks, we are able to infer user presence, meaning we can assume by this information that the user is currently interacting with the device or generated an action that is currently producing packets (such as watching a movie). From user presence, we extract user absence, which are the periods of time when the user is not interacting with the device. We use user absence as ground truth for our filtering algorithm. Periods of user presence mix user-facing and background packets and so, we cannot get ground truth. For this reason, we utilize user absence periods. By doing this, we are able to evaluate the efficiency of the filtering process.

The main contributions of this thesis are defining the problem scope based on the definition of user-facing and background packets and developing an algorithm to label each network packet based on this definition.

The rest of the thesis is organized as follows: Section 2 describes the problem statement. Section 3 describes the datasets used, along with explanations on the differences between individual devices data available. Section 4 focuses on how we obtained ground truth to evaluate our algorithm. This is followed by section 5, which describes how user absence is detected from the network, by explaining the filtering methods used and the reason for picking them. Section 6 analyses the efficiency of each of the filtering methods for different users. Finally, section 7 explains related work and section 8 concludes presenting future work and final comments.

2 Problem Definition

Our goal is to determine for each packet whether it comes from an user-facing application. By this, we mean to distinguish when the user is directly using an application and the application is generating traffic versus traffic generated automatically on the background. When streaming videos or music, it is not guaranteed that the user will be watching or listening to the music at the time, but we infer that high chances are that he is. This way, we define as user-facing traffic the network packets that have been generated by previous or current user action. We take as input a stream of packets traversing a router and output user-facing packets marked with a flag.

Background activity consists of data resultant from background applications running on devices and generating online traffic (such as a software updates or websites refreshing their pages, for instance), without direct manual request from the user. Making sure these two different types of packets (user generated and background activity) are identified is key to our study, because only the user generated ones indicate real user online presence.

Differentiating user-facing from background packets by solely monitoring network traffic is not an easy task. Many applications may run in parallel and, additionally, there are cases where the same application may run on the background or with the user. For example, Skype can generate user-facing packets if a user is making a call or generate background packets if the software is updating or checking the quality of connection. Another example are news websites, like the New York Times, that refreshes the webpage from time to time, without user request. Identifying these nuances are one of the main challenges of this thesis.

3 Dataset

We take an empirical approach to develop our filtering method. We evaluate packets collected from users' devices. Data collection was conducted with Hostview, a tool developed by Inria. Hostview is an end-host monitoring tool that runs constantly on the background, and collects various system configuration, status and performance metrics. Some of these metrics are event triggered such as network connectivity changes (network interface goes up or down), system power states (screen state, battery charge state), and user presence (user device interaction, such as mouse and keyboard clicks, and foreground application). Other metrics are polled periodically including wireless network statistics, the set of running applications, IO activity and active sockets. In addition, Hostview captures packet traces of all network traffic (first 100 bytes) using libpcap. Hostview captures the TCP/IP headers of all traffic, part of the HTTP header and full DNS packets. The information collected from the HTTP headers includes the request host, method and response status, the request path, the content type and length and, finally, the request referrer. When the user receives or sends a packet Hostview looks at the header and if it is a DNS or HTTP packet it logs it. Using Hostview is appealing because it provides data such as the network packet traces and extra data from the device, such as which applications are on fullscreen mode and user mouse clicks.

Hostview collected different datasets. Our analysis of the data showed some issues on the collection of some of these. For instance, flows and sockets datasets had missing information for many users on many days, meaning that packets that could be seen on the DNS and HTTP datasets didn't have any equivalent packets on the flows and sockets datasets. Moreover, other data as the traffic being generated by the device, was accurate for certain users only in a few days and missing for others. The two datasets related to network traffic that appeared to be the most complete were the ones containing DNS and HTTP requests. For this reason, the rest of this thesis focuses these two datasets, which were used to determine user facing presence. Our methods are valid for packets streams more broadly and our goal is to re-run the analysis after the new deployment of Hostview.

Inria and the University of Nottingham conducted the studies that gathered the data with Hostview. The first part of the study happened in France, with 12 participants. The second phase gathered data from users from the UK, with a total of 14 individuals. Data collected included both PCs and mobile devices. The necessary information to generate ground truth and test our algorithm was only available for a subset of these users, though. For this reason we focus on eight devices, Windows PCs and Windows laptops, of different participants in the United Kingdom. Some devices belonged to more than one person in the same family. For instance, one device was shared by husband and wife.

The amount of data available for analysis from each device differs. The quantity of days with available traffic information varied per device; some having only 4 days available and others having up to 15 days. Some users had packets every day, while others had periods of no usage or were not monitored. As it can be observed in other studies that also deal with collection of user data [1]. Some users provided a lot of data continuously, throughout the two months, while others have long periods of inactivity. Day long measurements mean that many times the device connected to the Internet from different locations, resulting on different network environments. There are devices, for instance, with very few HTTP requests, while others provide more than 200,000 URLs accessed. This type of disparity extends to other data types, like DNS requests. Sockets were not available for seven out of the eight devices used. Moreover, for the one that contained sockets data, on many days it was not accurate or simply didn't exist. The flows and device traffic datasets also presented problems. The analysis in this thesis helped uncover a number of issues with the data which are now being fixed for the new version of Hostview.

4 Ground Truth

User absence are the periods of time when the user is not interacting with the device. It is the opposite of user presence, which includes activities such as web browsing, playing online and offline games or editing documents. User absence represents the ground truth in our study and is used to evaluate our filtering method. Having user absence as ground truth means that we affirm with 100% accuracy that the user was not interacting with the device at a determined time.

In order to obtain user absence, we first had to obtain user presence sessions. A session is defined as an interval of time on which the user has uninterruptedly interacted with the device. Data necessary to determine user absence is provided by Hostview. We get the data in two phases:

1. By extracting the exact time participants manipulated mouse, keyboard, microphone and speakers, and by applications running on full screen mode on the user's devices.
2. By creating user presence sessions with this data.

We had to transform information collected from mouse, keyboard, speaker and microphone usage into sessions, which correspond to an interval of time. Many mouse clicks close to each other, for instance, integrated the same session. The key point was to determine how far apart a

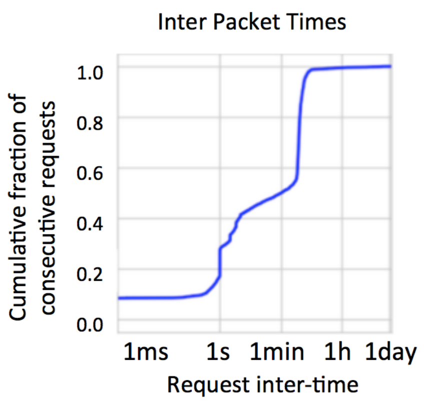


Figure 1: Cumulative fraction of consecutive requests of Avast inter packet times.

click should be to be considered part of a new session. Based on a previous study [2] that evaluated mouse and keyboard inter-event times, we observed that most of these interactions are no more than 150 seconds apart. Utilizing this information, mouse, keyboard, speakers and microphone packets were grouped together when they were less than 150 seconds apart from each other. The information regarding the state of the application screen was already provided into intervals of time and only needed to be connected to the created sessions if they were no more than 150 seconds apart.

5 Detection of user-facing traffic

This section describes the algorithm used to detect user-facing traffic. The algorithm is divided into three main smaller algorithms, each of them responsible for analyzing a given packet and labeling it. The method works as a decision tree, where each of the smaller algorithms is a decision node. If the first node is not able to label the packet, it will send it to the next node. Our algorithm to identify if a packet is user-facing or background is unsupervised. It receives a stream of packets in real time and as output, classifies each of them as user-facing or background.

By analyzing the Cumulative Fraction of Consecutive Requests (CFCR) for inter packet time per URL we observed patterns and inter packet times that repeated at same periodicity. In most of them we noticed many inter packet times of less than a second and also, in some cases, a big number of inter packets of the same length. We assume that packets related to an inter packet time of less than a second are all either background or user-facing depending on the first packet that generated the burst. Background related packets should be filtered out. Also, a packet whose interval to the next one is very periodic should be filtered out. The reason for that we will be further explained when we talk about the short intervals filtering method and the periodic intervals filtering method.

One example to illustrate the behavior described is on Figure 1. It refers to Avast, an anti-virus software, that can be manipulated by an individual, but also runs on the background, when doing software updates or scans. The plot clearly shows that a number of requests happen in less than a second from each other. We can also observe a line that seems almost vertical around the five minutes measure. This line corresponds to a periodic interval.

We also compared the CFCRs of specific URLs to the interval times where users were known to be absent to characterize if the background traffic occurred solely on periods of inactivity. The results, as expected, still showed periodicity when users were present, but were accentuated on intervals where the user was not.

Our algorithm uses three filters, which work as decision nodes on a decision three and will label the packets, like illustrated on Figure 2. In this figure, the gray square represents the input: a packet. Each of the nodes represent one of the filters used: blacklist, short inter packet intervals and periodic intervals. If the first filter, blacklist, is able to identify the packet as background, it labels it that way (represented by the red square), otherwise the packet is analyzed by the next filter. This process persists until the packet is labeled. In the image, blue squares represent the user-facing labeled packets.

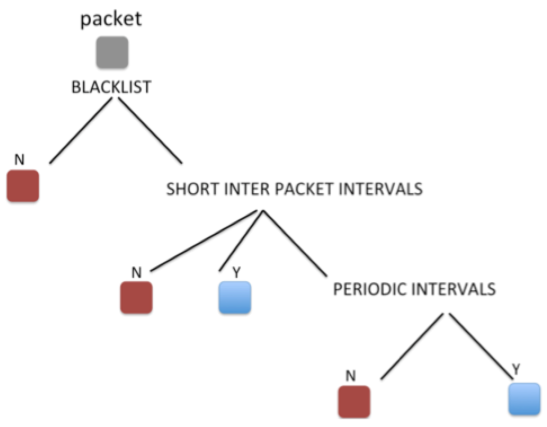


Figure 2: Decision tree with each filter.

5.1 Blacklist filter

By plotting the histogram with all the URLs accessed by the users we observed that many of them were related to background URLs, like the ones associated to software updates. Some softwares might generate a lot of background traffic, like Skype, which is constantly testing whether the user is online and getting which of the user's friends are online. Based on these observations, the first filter consists of a blacklist with URLs related to background traffic. We utilize as blacklist a list obtained online. Even though it is not complete, it is still an important and fast method of filtering, by the fact that is consists of a simple URL association.

5.2 Periodic intervals filter

The observation of CFCRs showed vertical lines, associated with periodic intervals. These periodic intervals are characterized most probably by a software generating updates. Updates tend to happen in constant intervals of time. User behavior is never that precise. For instance, a user will not make requests every 30 seconds, it will vary.

The periodic intervals filter evaluates if a packet is part of background activity or not by determining if such packet is part of a periodic interval. This is determined partially by the theoretic count.

$$\text{theoretic counts} = \frac{\text{session length}}{\text{periodicity}}$$

The theoretic count depends on two variables: the session length and the periodicity. A session is an interval in time made from DNS or HTTP domain packets that are up to a certain time gap apart from each other.

The periodicity is an interval length in seconds. For each session, we calculated the intervals that appeared and the number of times each of them was repeated. For sessions with less than 5 packets, we calculated periodicity if a certain interval length was repeated on more than 50% of the session. For sessions with at least 5 packets, if identified interval lengths equivalent to over 30% of the intervals in a session we calculated theoretic counts. The theoretic count was then compared to the real count - number of times the interval actually appeared in the whole session. If the two values were close, given a small error window margin, the whole session was deleted. Otherwise, just the spike, equivalent to a certain part of that session would be erased. If the analyzed packet is part of the deleted session, it means it it related to background activity.

5.3 Short intervals filter

As explained when analyzing the CFCRs of inter packet times, we noticed intervals smaller than one second. We consider packets associated with these intervals as part of the same group, meaning that they should be labeled the same. The behavior that characterizes such short intervals comes from an action that triggers many events. As previously explained, when the user accesses a website, his first request will generate many others. These generated requests will be very close in time from the first one (less than a second). This behavior might also be generated by background activity. This is why when we identify a burst of short intervals, the classification of the first packet will determine the classification of the other ones.

5.4 Detection algorithm

We classify the filters in two main groups: background filter (blacklist and periodic intervals) and event aggregation (short intervals). Algorithms 1, 2 and 3 present the pseudo code for the detection algorithm, with each filter explained.

Algorithm 1 Detection method

```
function IDENTIFY PACKET(observedPacket, packets, blacklist)
    if $observedPacket$ in $blacklist$ then
        return False
    ▷ packets contain all packets with the same url as the observed packet in the last interval of five minutes from the observed packet
    $previousPacket \leftarrow$ packets[len(packets)-2]
    $interPacketTime \leftarrow observedPacket - previousPacket$
    if $interPacket$ <1 and previousPacket is background then
        return False
    else if $interPacket$ <1 and previousPacket is userFacing then
        return True
    $onlineActivity \leftarrow$ ELIMINATECONSTANTINTERVALS((block1sec[url]))
    if $observedPacket$ not in $onlineActivity$ then
        return False
    return True
```

Algorithm 2 Constant interval elimination method

```
function ELIMINATECONSTANTINTERVALS(packets)
    $invDist \leftarrow$ GETINTERVALDISTRIBUTION((packets))
    for $inv$ in $invDist$ do
        if $(\frac{\text{invDist[inv]}}{\text{invDist[total]}} \geq .5$ and $\text{invDist[total]} \leq 5)$ or $(\frac{\text{invDist[inv]}}{\text{invDist[total]}} \geq .3$ and $\text{invDist[total]} \geq 5)$ then
            $realCount \leftarrow$ invDist[inv]
            $theoCount \leftarrow \frac{sessionLength}{inv}$
            if $theoCount \leq 10$ then
                $errorMargin \leftarrow .2$
            else if $theoCount \geq 10$ and $theoCount \leq 100$ then
                $errorMargin \leftarrow .15$
            else
                $errorMargin \leftarrow .1$
            if $realCount \geq (theoCount - theoCount * errorMargin)$ then
                eliminate whole interval
            else
                eliminate spike
    return filtered packets
```

6 Evaluation

In order to evaluate the algorithm we compared the efficiency of the filtering methods on user absence periods. Considering we had ground truth for the intervals of time the user was not present, we could identify the packets that were generated by background activity: packets generated on user absence periods. We then checked if we could label these packets as background when applying the filtering methods. We plotted the Cumulative Fraction of Intervals (CFI) for four different

Algorithm 3 Interval distribution method

```
function GETINTERVALDISTRIBUTION(packet)
    for packet in packets do
        $interEvent \leftarrow next\ packet - current\ packet$
        invDist[interEvent] ← invDist[interEvent] + 1
        invDist[total] ← invDist[total] + 1
    return invDist
```

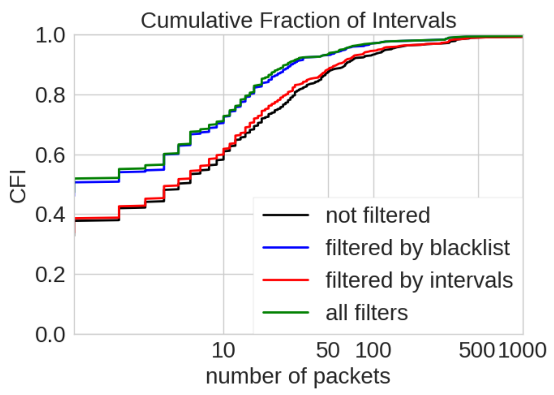


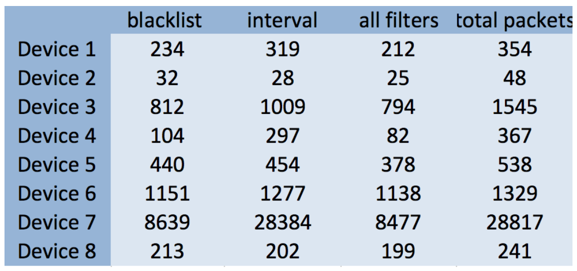
Figure 3: Cumulative fraction of intervals for all users.

combinations: non filtered packets, blacklist filtered packets, interval filtered packets (combination of short and periodic interval filters) and filtered (all methods together). These results are available on Figure 3. They show the distribution of intervals containing a certain number of packets. Perfect results would present all intervals with zero packets.

We can observe that all filtering methods show improvement when compared to non filtered packets. This improvement, though, is still not a good result. When analyzing in more detail packets that were not filtered, they most times corresponded to URLs of short intervals, which didn't give us enough precision to eliminate them on the periodic filtering. Making this filter more aggressive, by allowing a bigger error margin between theoretic counts and real counts, in order to eliminate whole sessions instead of partial intervals could be a solution to improve that.

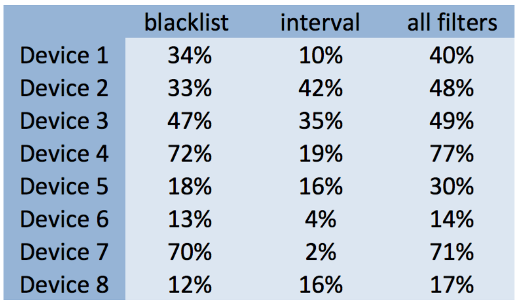
Besides the CFI, which was referent to all devices combined, we analyzed each result separately using a table containing the number of remaining packets after each filtering (Figure 4) along with the percentage of filtered packets it represents (Figure 5).

The blacklist filter was more efficient than the intervals one in most cases. We concluded that using this filter, though, is not a good option. We found the pre-made blacklist online and it is assumed that it is impossible that it contains all possible background related URLs. Moreover, it includes URLs related to advertisements, which not necessarily are background generated. Another issue is that the list is static, meaning is doesn't increase in size by adding new identified URLs and makes it a limited method. The fact that the list doesn't change, also makes the elimination of packets very specific for each user. For instance, if two devices happened to request the same number of URL background packets, but only the ones for the first device are present on the



	blacklist	interval	all filters	total packets
Device 1	234	319	212	354
Device 2	32	28	25	48
Device 3	812	1009	794	1545
Device 4	104	297	82	367
Device 5	440	454	378	538
Device 6	1151	1277	1138	1329
Device 7	8639	28384	8477	28817
Device 8	213	202	199	241

Figure 4: Filtered packets per device on absence periods.



	blacklist	interval	all filters
Device 1	34%	10%	40%
Device 2	33%	42%	48%
Device 3	47%	35%	49%
Device 4	72%	19%	77%
Device 5	18%	16%	30%
Device 6	13%	4%	14%
Device 7	70%	2%	71%
Device 8	12%	16%	17%

Figure 5: Percentage of filtered packets per device on absence periods.

blacklist, the results of this method will vary from 100% to 0% precision. When further analyzing this result, though, we concluded that the blacklist is not an appropriate filtering method. Most websites that generate blacklist, include on their lists URLs related to advertisements as said, which could be user generated. This is a very tricky characterization because these URLs can be generated by the user fetching a web page (user-facing packet) or by a website refreshing the page without user intervention (background packet).

The filtering by intervals showed really poor results, with the best one being 42% of packets filtered on device 2. We think that making the periodicity filter more aggressive would results in a great improvement in these results.

Finally, the method that combined all filters was the best on every device, as expected. In many cases, the performance of the filtered method is only slightly better than the background one, meaning that a combination of the three filtering methods was not as effective as expected.

7 Related work

Analyzing user traffic has been an area of common interest for many researchers and a variety of methods to extract the data and fetch relevant information can be found. These studies focus on traffic at large without distinction between user traffic and background traffic [3] [1]. Many studies on device and user behavior focus on device performance, energy consumption or identifying user daily patterns, such as distribution of usage during the day or routines of app usage, and are based solely on mobile devices [3] [4] [5] [6] [7]. None of the previous studies capture user presence. For energy consumption, for instance, being able to evaluate how user generated and background traffic differently affect battery performance could help on designing algorithms to decrease energy consumption. Different methods could be applied when it is known that the user is not present. Many of these optimizations should also not be a one size fits all solution: users interact differently with their devices, accessing different types of websites and softwares that might be more or less demanding [1] and as so, need to be correctly analyzed in order for the best solution to be provided.

Moreover, even the studies that make analyzes based on background and user facing traffic [8] generate these two types of traffic from controlled experiments and simulations, and not from real traffic. This gives them a notion of these two types of the data without the need to identify each of them, but could not be applied to real users, exactly by the fact that they wouldn't be able to differentiate background from user facing traffic. Even though this type of experiments can be precise, they still don't fully represent data fetch from a real person. Additionally, some other studies, despite using a big group of users, are based only on the observation of mobile phones' usage[9] [4]. This restricts the generalization of results. Mobile phones might be the most popular among users, but tablets and laptops are also part of users daily lives and should be studied as well. This is an issue that is also present in our study but that can be solved with the future data fetched by Hostview, which is still being improved and used.

8 Conclusion and future work

The main difficulty on this work was correctly formalizing the problem to be answered with the available data. Before deciding to evaluate user-facing packets, the main focus was to analyze the periods of time a user was present. Trying to find solutions for the lack of data, such as the flows datasets and trying to create ground truth from the data were hard challenges. The main barrier of not being able to differentiate background online applications from the user-facing ones on the user activities dataset led to a reformulation of the problem, where the focus became user absence periods.

The method provided still has a lot of room for improvement. We didn't have time to evaluate it in depth, which means we applied no ameliorations for the filters. As we mentioned, one possible solution for bad results could be being more strict with the periodic intervals filter. Future work should focus on determining the precise limits of obtaining the best out of the periodic filtering. We could make it more or less aggressive by determining the quantity of period repetitions that should be filtered, or the rate that repetitions should appear in a session in order to filter the whole session out. If it was possible to obtain user presence ground truth, it would be interesting to adapt the aggressiveness of this method. Making it too strong could end up wrongly detecting user-facing packets as background ones.

Another future approach for ameliorating the final method could be done by introducing new types of filtering. A promising example is using a collaborative blacklist. This would be a blacklist created by URLs filtered on the periodic intervals. This is an interesting solution because there is a huge number of URLs that are related to background activity and determining and adding all of them manually to a list is nearly impossible and a very time consuming task. We could further analyze if the URLs detected by the periodic filters were all related to software updates, which would give us a valid blacklist. The collaborative blacklist guarantees that any URL identified as background activity for one user, will be filtered for the other ones as well. It can be adapted depending on the group of users and the results and would fit on cases where the URL intervals are not long enough for identifying periodic intervals.

Moreover, besides the datasets available with user information, each participant on the study was interviewed, commenting on their network habits and routines. Trying to map the results containing their daily usage and trying to further reflect how much can be inferred from their real routines by just analyzing the plots is also interesting.

In order to obtain more general results, analyzing not only windows laptop and PC users, but mobile devices running on different operational systems is also relevant. Results would be richer, even though, they are expected to be as precise as the ones from windows users. We would need to run a new study collecting the necessary type of data from the various devices, but since the Hostview study is on going, plans are that new data will be tested and new results achieved. With these results the filters will be able to be properly optimized. It would be interesting to have users from different countries as well, since after determining user-facing packets, their daily habits and daily usage times could also be analyzed.

Analyzing user traces can provide relevant information on many fields of user analysis, such as energy consumption, user daily routines and application usage. This work only starts exploring the possibilities of extracting user data, but shows that determining user online presence by network traces can be an interesting and promising field of research.

References

[1] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govidan, and D. Estrin, "Diversity in smartphone usage," in *Mobisys '10 Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 179 – 194, June 2010.

[2] H. Lundgren, "Preliminary report on user profiling," September 2014.

[3] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *IMC '10 Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 281–287, November 2010.

[4] Q. Xu, J. Herman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *IMC '11 Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 329 – 344, November 2011.

[5] V. A. Siris and D. Kalyvas, "Enhancing mobile data offloading with mobility prediction and prefetching," in *MobiArch '12 Proceedings of the seventh ACM international workshop on Mobility in the evolving internet architecture*, pp. 17 – 22, August 2012.

[6] D. Ferreira, A. K. Dey, and V. Kostakos, “Understanding human-smartphone concerns: a study of battery life,” in *Pervasive’11 Proceedings of the 9th international conference on Pervasive computing*, pp. 19 – 33, June 2011.

[7] Y. Li, J. Yang, and N. Ansari, “Cellular smartphone traffic and user behavior analysis,” in *Communications (ICC), 2014 IEEE International Conference on*, August 2014.

[8] R. K. Sheshadri, I. Pefkianakis, H. Lundgren, D. Koutsonikolas, A.-K. Pietilainen, A. Soule, and J. Chandrashekar, “Characterizing mobile user habits: The case for energy budgeting,” *IEEE INFOCOM 2015 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2015.

[9] G. Maier, F. Schneider, and A. Feldmann, “A first look at mobile hand-held device traffic,” in *PAM’10 Proceedings of the 11th international conference on Passive and active measurement*, pp. 161 – 170, April 2010.