A Discussion of User-Induced and Non-User-Induced Traffic

Benjamin C. Flock

May 3, 2023

Abstract

When analyzing networks, there is rarely a focus on the traffic that is explicitly generated by users as opposed to traffic that is generated without user intervention. As consumer computing has grown more widespread, and the network has grown to accommodate more and more users, there are many more services which communicate with other devices in the network for non-user induced reasons. This paper attempts to take a high level look in a very narrow case to see if there are any feature which distinguish periods of user-induced traffic from periods of non-user-induced traffic. We assess 6 different candidate ”tells”, and find two tells that are heavily related to and work as very good indicators for the machine’s current usage status.

1 Introduction

Networks are, by their nature, interconnected and interdependent systems of high complexity. Devices can talk to each other over networks, and can do so in ways that are not necessarily user generated. In an age of cloud computing and big data, many devices and applications do their computation on the cloud, rather than fully on the device itself. Applications may seek to talk to servers even when they are not in use or collecting data. Applications frequently send information back to the server without the user deliberately triggering that traffic. Messaging applications such as Google Messages, Snapchat, and Instagram all provide Read Receipts, which notify the server that a message has been read, even without the recipient of the message sending anything in response. However, in this paper, we primarily focus on the type of network traffic that is created when applications generate traffic without any user involvement.

This research question is, somewhat surprisingly, rather underdeveloped. This is likely due to three main reasons. First, the research question isn’t asking about a specific phenomenon, but rather investigating the possibility that a phenomenon may exist. Second, reproducibility and generalizability are difficult to establish due to the inherent uniqueness of every consumer system. Finally, consumers are sensitive about their data being potentially compromised. Despite these issues, this paper attempts to take an initial, naive look at how devices create traffic in the background, when a device is not in use. This paper’s study is by no means comprehensive, but rather takes a broad look at 3 systems to analyze traffic that is generated. The majority of the study was done on a consumer Windows 10 machine. The device was left in sleep mode for 10 days, while Wireshark was collecting traffic. The device's traffic was then measured again, while in use, specifically while playing online games with friends. Traffic was also measured on a mostly-unused Raspberry Pi 4 device, and on a Windows Server Virtual Machine, hosted on Azure, with no security policy.

The background traffic that we're looking for can take a number of different shapes depending on the applications on the device. Many applications remain open in the background and send traffic to the server periodically. Discord, for example, notifies the server if a user has come online or if they are idle. It also updates the server if the user is playing any games or listening to music. Many applications check for updates from a remote server as well. Future work should extend this research toward many consumer devices, and see how their behavior differs depending on the applications installed or in use. While there are ethical concerns with this kind of surveillance, the author proposes(but has not implemented) a script that would sufficiently remove sensitive data, by only using network layer and transport layer packet headers.

When discussing this subject, it's important to avoid talking about "user generated" packets. Traffic generation is a spectrum, and describing packets as generated by a user, instead of saying something like "induced by user behavior", doesn't make it clear which category we're discussing. A short list of categories

1. Traffic may be generated by a user fully. This describes traceroute and other network tools that are intended to generate specific types of packets and traffic

2. Traffic may be generated by a user *mostly*. The user intends to communicate to another device on the network, but doesn't specify how exactly to do it. This is where we classify tools like web browsers.

3. Traffic may be generated by an application that the user is interacting with. A user may be playing chess in browser, for example, and the users action makes a call to a backend service.

4. Traffic may be generated by an application without user interaction. This would include a notification about an incoming message from Skype or Discord, for example.

5. Finally, traffic may be generated by an application in a way that is fully insensitive to user state. This would be things like analytics reports, heartbeats, as well as responses to incoming messages.

In this study, our view of "passive" traffic captures items 4 and 5 from the above list, and "active" traffic is intended to refers to items 1, 2, and 3.

2 Data and Methodology

Data was collected with Wireshark[1], which was configured during the capture periods to run in the background at scheduled times.

2.1 Windows Client

A Windows Client machine was used for two of the sets of measurements. The first measurements, heretofore described as the *passive* measurement dataset, encompass a 10-day long period where the computer was not used at all. Unfortunately, it became impossible to work with that much data on the cloud or on the machine available to me; instead, I had to take a slice of that data. Since the measurement was continuous and the machine remained unused, I expect that this approach will still work. The second dataset is the *active* measurement dataset. This is a dataset from a period when the computer was in use. Specifically, this is a time interval when the network connection is in *heavy* use. These measurements were taken during video calls, and while playing online games. These are not representative of *all* use cases; but it would be difficult to create a representative dataset that was representative of most use cases and most machines, especially in the span of 14 weeks and with only one machine actually being used.

2.2 Windows Server

A Windows server machine on Azure was left operating with no firewall, and collected data for 5 hours, the maximum allowed time for packet collection using azure's tool.

Windows Server is a best attempt to get an approximation of a "ground truth" without acquiring another bare metal, newly imaged Windows machine. Windows server is lighter than Windows 10. The expectation is that traffic on the Windows 10 machine, both in periods of use and when the machine is left running idly, would look like a superset of the Windows Server machine's traffic. The traffic on the Windows Server machine should just be doing fairly rudimentary Windows tasks, namely sending analytics data to Microsoft, and polling for potential updates. This collection is slightly complicated by the fact that the machine is on Azure, and so it needs to talk to Microsoft's azure communication channel, 168.63.129.16.

As we will discuss in later sections, there are some oddities with the Windows Server machine due to the connection to Azure.

2.3 Raspberry Pi 4

The last set of data, labelled *Pi*, was taken from a rarely used Raspberry Pi 4 machine running Raspberry Pi OS. This machine was left on its own for 24 hours. It was connected to the same wireless network as the Windows machines.

3 Tells

In order to determine if a period of traffic occurred while the system was in use, it is necessary to identify a number of "Tells". The goal of these Tells is that they should, ideally, look different in an intuitive way if the machine is in use. As such, going off of our initial assumption about the Windows Server machine's behavior being a subset of the Passive behavior and the Active behavior, we expect to see some consistency when comparing the Windows Server dataset, the Passive Dataset and the Active dataset to each other.

3.1 Ports

As shown in Figure 1, the primary difference between the Active and the Passive Dataset is that there are a number of ports in the Active dataset that are transmitting data in very low frequencies. The Source Port seems to serve as a good Tell for if the traffic occurred during a period of use.

Figure 2 shows that Destination ports do not serve nearly as well as one of these Tells. This makes sense. A source machine can typically send traffic from a nonconventional port, but a server listening will typically be listening on a reserved port.

Note: the reason for the high traffic to high ports is due to Windows using high ports for fire-

wall services. [2]

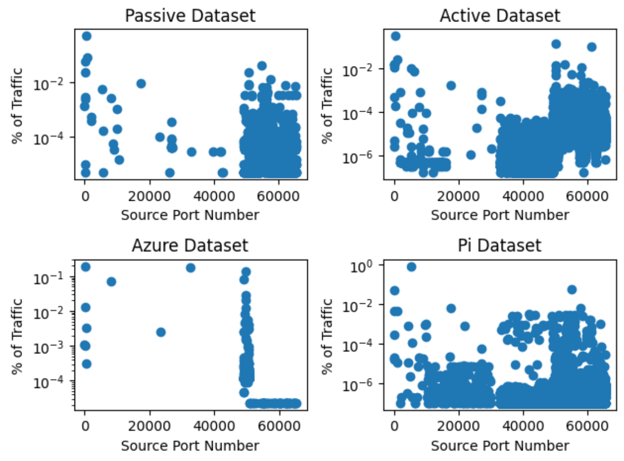

Figure 1: Percentage of Traffic that originated from a given port

3.2 Protocols

(Results can be found in Table ??)

The dataset showed some interesting characteristics with regard to the protocols being used. First, the Pi was talking on MDNS *constantly*, which explains some of the other results in this paper. The Passive Dataset used TLSv1.2 and QUIC as it's two highest, with HTTP not even placing. However, the Active dataset and the Azure dataset both showed HTTP as being in the top 10. This is likely due to the Windows Server machine being on Azure, therefore using some HTTP in order to communicate with the rest of the Azure ecosystem. However, HTTP's presence in the Active dataset's top 10 protocols is noticeable. The absence of HTTP in the Pi dataset also helps us determine that this is likely caused by user interaction. A quick WHOIS lookup helped determine that many outgoing

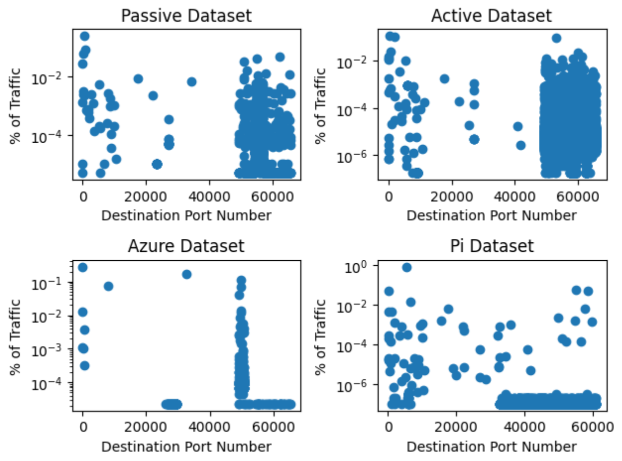

Figure 2: Percentage of Traffic that is set to arrive, or has arrived, on a given port

HTTP packets were going to Akamai-owned IP addresses.

3.3 Sources

Despite the usefulness of Source Ports as a Tell, the dataset did not make source IPs seem to be a very good Tell. Table 2 shows that there's far too much traffic even when the machine is not in use that originates from outside the local area network. Most, if not all, the Azure IPs are owned by Azure. Our finding is that these are simply too sensitive to each specific device's configuration, location, and local area network in order to use source IP as a useful heuristic.

3.4 Destinations

Table 3 shows the top 20 Destination IPs for each of the datasets. There's nothing inherent about these IP patterns that help us determine any real information about the devices, unfortunately.

However, there is some information we may use in both this set and the set of source IPs. If instead of looking at the patterns, we look at the IPs themselves, and do a WHOIS lookup, we can start to paint a picture of what the device is doing. For example, 18.208.4.124 is known to me to by my ec2 instance, and 66.22.212.132 is owned by Discord. Known information can help slightly in identifying whether a user is using the machine when the traffic is captured, although it would be better to have an automated system that can work off patterns, rather than requiring knowledge that may become outdated or difficult to acquire.

3.5 Packet Length

We initially theorized that by plotting Packet Lengths, it would be possible to discern between devices that were in use or not. This was based on the assumption that, when the device is not in use, it is merely sending packets of constant sizes elsewhere. This assumption is based on the fifth type of generated traffic, and might hold true if that were the only type of traffic occuring in passive periods. However, update notifications must have a variable size inherently. Therefore, this is an ineffective Tell for traditional consumer devices, although for certain scenarios where notification type traffic is not a concern, it may still work effectively as a tell.

There is one odd thing about the data, as figure 3 shows us, that sets one of these data points apart from the others. Devices on Azure talk to a special node at 168.63.129.16[3] that connects the virtual device to the Azure platform using frame sizes *far* larger than Ethernet supports. This virtual IP is responsible for managing the connection between the VM and the Azure platform as a whole; for example, the load balancer

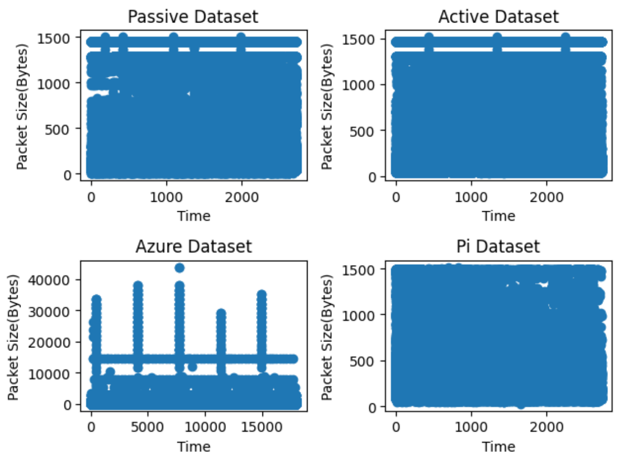

Figure 3: The lengths of packets over a time interval

and heartbeat are dependent on it.

3.6 Volume

The volume measurement was done by reducing the data into buckets, and counting the size of the buckets. Unsurprisingly, the volume of packets increases pretty significantly moving from the Azure dataset to . However, packet traffic doesn't always follow a consistent pattern. To test if the volume is actually higher, we need to use a statistical test, specifically the t-test. Using scipy, we performed a t-test between the Passive dataset and the active dataset. With a p-value of -7.134×10^{-10}, we can say that the packets volume did in fact change, and pretty significantly. The caveat here is that this may depend on machine, and on what's being done on the machine. Additionally, a ttest will need to be run every time; a brief instance of higher volume isn't enough to determine the machine is being used by a human. Figure 4 shows the dramatic difference, by orders of magnitude, between the

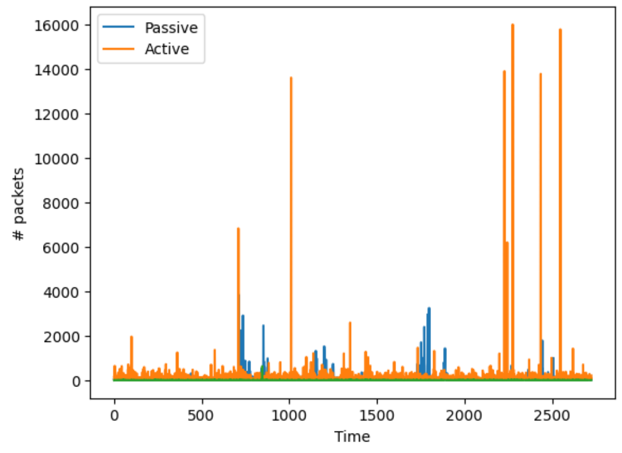

Figure 4: Packets per second for the Azure Instance, the Passive dataset, and the Active dataset

3 windows machines; however, it's impossible to know if this will always hold true without testing more machines.

4 Ethics

No data was anonymized for this study or in this paper. However, given that no payloads are in the data, there is no risk of any sensitive information being revealed.

More generally speaking, there are surveillance concerns with developing methodologies for determining whether a machine is actively being used or not. In workplaces, it could lead to employees having their privacy invaded, and in other contexts, it could be used to, for example, determine whether the inhabitants of a house have left on vacation, and burglarize them.

That being said, understanding the properties of network traffic can help to prevent cyber crimes. Being able to discern when a machine is behaving in an atypical way can help to determine if that machine is compromised.

5 Future Work

The author of this paper sees a number of ways to go forward from this study. Future work should explore the development of a tool that can be safely deployed and can gather necessary information that people can feel safe having on their devices. This degree of collection will enable a closer look at a more general pattern of usage. There is also, undoubtedly, more information to gather from the packets already collected. An entire paper could likely be dedicated to analyzing how time-related aggregations change, using only the data already collected.

It may be possible to build off this methodology to improve Network Intrusion Detection Systems; by knowing how a machine left passive should behave, it should be easier to identify if a machine has been compromised. The immense volume of packets sent by modern machines can make it difficult to process that data, but by focusing on the Tells that we've identified it could be possible to identify compromised machines.

Finally, this paper is primarily focused on Windows machines. The Raspberry Pi showed findings that were somewhat unexpected, and it could be worthwhile to do a study on Linux machines, server and client, similar to this study.

6 Conclusion

In this paper, we generated and analyzed packet traces to see if there were properties we could use to accurately guess whether the machine generating packets was in use, or was generating packets in the background. We found two features that are most telling in this regard. First, we identified that a machine in use has the presence of traffic on high ports, in a range that falls below the firewall window of ports. Second, we identified that packets are transmitted from a machine in much greater volume if the machine is being used. We also looked at a Windows server machine and a Raspberry Pi machine. The Windows Server machine had the least traffic of all the devices, as expected, but gave us an insight into how traffic patterns accumulate as systems becomes more complex. The Raspberry Pi was used as a bit of a baseline, something outside the Windows ecosystem to compare to.

References

[1] Wireshark Foundation, "Wireshark." [Online]. Available: https://wireshark.org

[2] Microsoft, "Service overview and network port requirements for Windows." [Online]. Available: https://learn.microsoft.com/en-US/troubleshoot/windows-server/networking/service-overview-and-network-port-requirements

[3] "What is ip address 168.63.129.16?" [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-network/what-is-ip-address-168-63-129-16

Appendix

Passive	Active	Azure	Pi
TLSv1.2	TCP	TCP	MDNS
QUIC	UDP	HTTP	UDP
TCP	TLSv1.2	NTP	NBNS
UDP	QUIC	TLSv1.2	DB-LSP-DISC
NBNS	SSDP	DNS	NBDS
DNS	SSHv2	NBNS	LLC
DB-LSP-DISC	NBNS	DHCPv6	TPLINK-SMARTHOME/JSON
MDNS	HTTP		ICMPv6
TLSv1	MDNS		DNS
SMB	IGMPv2		GVCP

Table 1: Top 10 Protocols used in each dataset

Passive	Active	Azure	Pi
172.20.111.188	172.20.111.188	10.0.0.4	172.19.3.193
172.20.14.32	66.22.212.132	168.63.129.16	172.19.12.232
172.217.1.97	185.107.192.26	169.254.169.254	172.19.11.74
142.250.190.46	13.107.238.35	168.61.215.74	172.19.0.6
129.22.4.31	35.244.180.134	40.87.160.0	172.19.14.206
52.85.72.195	172.20.14.32	13.107.4.50	172.19.0.219
172.20.8.233	104.22.24.231	40.87.172.0	172.19.11.198
172.20.11.241	142.250.191.225	20.189.173.9	172.19.5.251
172.217.4.195	66.22.212.131	20.209.1.1	172.19.5.63
172.20.7.207	66.22.231.56	40.119.46.46	172.19.2.240
162.159.130.234	172.20.6.112	fe80::f4c2:19de:d536:b6e2	172.19.12.181
146.75.76.237	18.154.227.57	13.107.12.50	172.19.10.173
162.159.135.234	140.82.112.3	40.80.44.0	172.19.13.220
66.194.187.21	8.251.206.254	51.132.193.104	172.19.15.11
172.217.1.106	8.240.200.126	20.42.73.27	172.19.9.114
146.75.76.238	129.22.4.31	20.189.173.15	172.19.14.81
23.220.140.149	104.26.15.184	40.79.189.58	172.19.3.68
142.250.191.99	162.159.135.232	20.44.10.122	172.19.3.24
142.250.191.206	172.20.46.155	104.46.162.226	172.19.3.64
142.250.190.69	172.20.0.6	104.46.162.224	172.19.8.48

Table 2: Top 20 Source IPs for each dataset

Passive	Active	Azure	Pi
172.20.111.188	172.20.111.188	168.63.129.16	224.0.0.251
172.20.15.255	239.255.255.250	10.0.0.4	172.19.15.255
129.22.4.31	172.20.15.255	169.254.169.254	255.255.255.255
142.250.190.46	185.107.192.26	168.61.215.74	ff02::1
255.255.255.255	66.22.212.132	10.0.0.255	172.19.109.55
172.217.1.97	104.22.24.231	13.107.4.50	208.67.220.220
172.217.4.195	224.0.0.251	20.189.173.9	ff02::fb
172.217.1.106	255.255.255.255	40.119.46.46	ff02::1:ff8d:e7f3
142.250.191.202	35.244.180.134	ff02::1:2	172.19.5.73
162.159.130.234	129.22.4.31	20.209.1.1	172.19.8.84
54.235.180.207	172.20.44.51	13.107.12.50	169.254.255.255
176.34.164.201	18.208.5.124	20.42.65.88	168.61.215.74
162.159.135.234	129.22.104.25	104.46.162.224	208.67.222.222
224.0.0.251	66.22.231.56	104.46.162.226	129.22.106.20
142.250.191.99	66.22.212.131	20.44.10.122	172.19.49.161
142.250.191.142	162.159.136.234	51.132.193.104	129.22.8.20
142.250.190.69	13.107.238.35	40.79.189.58	172.19.60.0
3.211.37.212	224.0.0.1	20.189.173.15	172.19.47.136
100.25.231.167	3.220.159.105	20.42.73.27	ff02::16
52.44.223.164	142.250.191.225	40.119.249.228	ff02::2

Table 3: Top 20 Destination IPs