Contemporary Engineering Sciences, Vol. 7, 2014, no. 23, 1245 - 1252
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.49154

Detecting Tethering between Devices with Different Operating System: An Experimental Report

Sungcheon Lee

Dept. of Stock and Finance, Sangmyung University
Cheonan, 330-720, South Korea

Hyun-chul Kim

Dept. of Computer Software Eng., Sangmyung University
Cheonan, 330-720, South Korea

Abstract

The increasing growth rate of smartphones, tables, and wireless notebooks as well as wireless cellular bandwidth have all led users to often share a single cellular connection among multiple devices, using a process called tethering. Obviously, sharing a single cellular connection with multiple devices can cause traffic overload on the cellular networks being used, thus it is becoming more and more required to detect such tethered traffic and devices, for more efficient and effective management of the networks. As an extension of our previous workwhere only tcp.window_size values were used for tethering detection, this paper additionally presents the further results obtained (i) when IP.ID values are used, as well as (ii) when the both values were used. Our results suggest that using either or both of IP.ID values and/or tcp.window_size values, we are able to detect tethering when used between devices with different types of Operating Systems

Keywords: tethering detection, OS fingerprinting, IPID, TCP

1. Introduction

The increasing growth rate of smartphones, tables, and wireless notebooks as well as wireless cellular bandwidth have all led users to often share a single cellular connection among multiple devices, using a process called tethering. Obviously, sharing a single cellular connection with multiple devices can cause traffic overload on the cellular networks being used, thus it is becoming more and more required to detect such tethered traffic and devices, for more efficient and effective management of the networks. In this paper, we address the problem of detecting tethered traffic of mobile devices, by proposing a method based on OS fingerprinting and traffic packet feature analysis. In particular, as an extension of our previous work [6] where only tcp.window_size values were used for tethering detection, this paper additionally presents the further results obtained (i) when IP.ID values are used, as well as (ii) when the both values were used. This paper is structured as follows. Section 2 reviews related work and patents recently proposed for the purpose of detecting tethered traffic and devices. In Section 3, we propose our method for detecting tethered traffic and evaluate it. Section 4 concludes this paper.

2. Related Work

In this section, we briefly review recently proposed patents for the purpose of detecting tethered traffic and devices.

2.1. Patent pending no. 10-2010-0105216 "System and method for controlling tethering in mobile communication network"

Proposed by the telecommunication company SK Telecom in Korea, this (more than) four-years-old patent primarily focuses on the characteristics of applications generating traffic of interests. It first identifies the causing application of target traffic using the deep-packet-inspection (DPI) techniques If the identified application is one of common well-known PC applications, such as P2P file sharing (e.g., BitTorrent), Web-hard, FTP, Microsoft Internet Explorer, PC games, etc., this traffic and the generating device is checked with a higher likelihood score of tethering. Once its likelihood score gets over a pre-specified threshold value, it is classified as of tethered one. Being a (more than) four-years-old proposal, this naïve method does not seem effective any more, as nowadays even mobile systems have become to support many P2P or FTP-like file sharing applications as well as PC games ported to mobile OS. Moreover, the proposed method can be neutralized, by simply changing the "user-agent" value of tethered desktop PC's web browsers to that of mobile web browsers.

2.2. Patent no.10-1361-8230000 "Method for deciding tethering service in communication system and apparatus therefor"

Proposed by the telecommunication company Korea Telecom, this method checks TTL (Time To Live) values in IP packet headers sent by mobile devices. The TTL number is decremented by 1 for every network hop that it goes through a network. The theory behind this method is that, if a mobile device has a TTL of 64 and if there are any other packets coming from the device with a different TTL value, then the user is highly likely to be tethering. This method allows us to detect tethering even if both the tethering and tethered devices use the same Operating System. Yet, users can elude this method as well, by manipulating TTL values of the tethered devices so that their packets contain the same, indistinguishable TTL information

3. Using IP.ID and/or TCP window size values for tethering detection

We use the eight devices as depicted in Table 1 for this study of tethering detection. TTL values of mobile device-generated traffic are used to collect the ground truth (of tethering traffic) data, as introduced in the previous section 2.2. We use the Wireshark software [4] to take a look inside the collected packet data.

Table 1. Devices used for our experiments.

Manufacturer	Device	OS
Samsung	Galaxy Nexus	Android 4.3
Samsung	Galaxy S	Android 2.3
Sony Ericson	Xperia Arc	Android 2.3
LG	Optimus G	Android 4.1
Apple	ipad mini	ios 7.0
Apple	Macbook Air	OSX 10.9
Microsoft	PC	Windows 8.1 Embedded
Nokia	Lumia 710	Window Phone 7.8

3.1. Using IP.ID field values from packets in uplink flows

In this paper, we use the definition of a traffic flow based on its 5-tuple (source IP address, destination IP address, protocol, source port, and destination port) with a timeout of 64 seconds. Based on this definition, we first convert the collected set of (uplink) packets into a set of flows, from each of Samsung Galaxy

Nexus (Android 4.3) and Apple ipad mini (ios 7.0), then compared those two flow sets, particularly their recorded IP.ID values. Here, ipad mini is configured to access the Internet through tethering on the Galaxy Nexus phone.

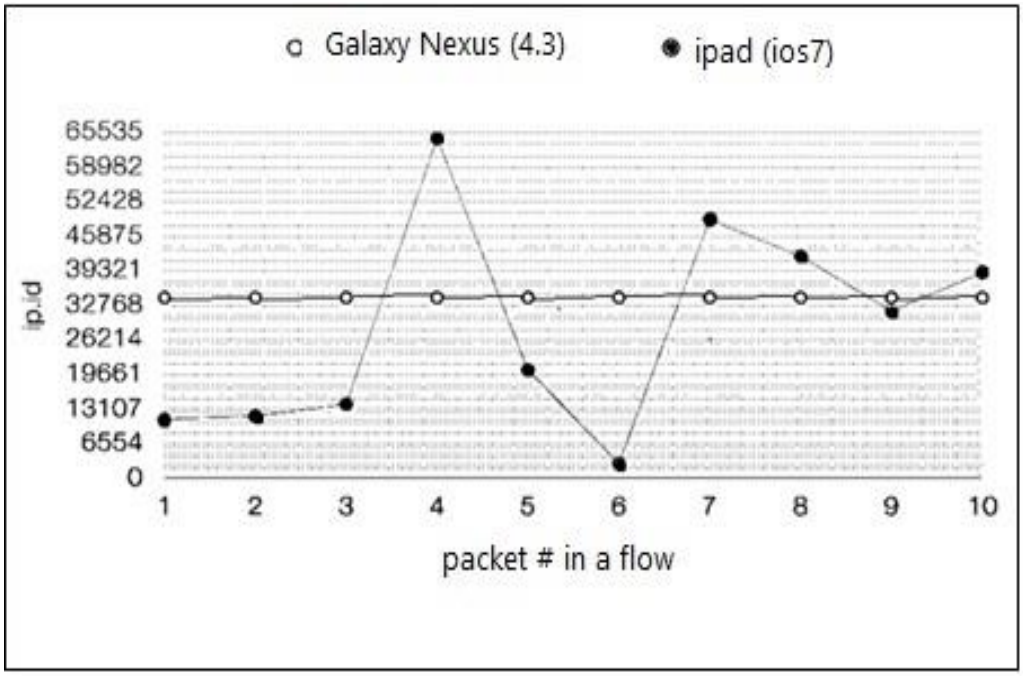

Fig. 1. Observed IP.ID values of packets of two different flows, collected from Galaxy Nexus (4.3) and ipad (ios 7), respectively.

As shown in Figure 1, we observe two different patterns in the IP.ID values obtained from the two different devices. According to our results that are summarized in Table 2, it is observed that different Operating Systems have different changing patterns in the IP.ID values used in a traffic flow. Consequently, when a device with a certain type of OS (e.g., Apple ios 7.0 as in Table 1) is connected through tethering on another device with another type of OS (e.g., Microsoft Windows 8.1 Embedded or Windows Phone 7.8), flows from those two devices should exhibit two different patterns of IP.ID values, which can be used to detect tethering between them.

Table. 2. Observed changing patterns of IP.ID values from different OS

OS	IP.ID pattern
Apple ios 7.0	Random
Apple OSX 10.9	Random
Microsoft Windows 8.1 Embedded	Random
Microsoft Window Phone 7.8	+1
Google Android 2.3	+1
Goolge Android 4.1	+1
Goolge Android 4.3	+1

Table 2 summarizes our experimental results on the changing patterns of IP.ID values, collected from uplink flows generated by different types of Operating Systems. As shown in the table, it is anticipated that we can detect tethering when Android devices use tethering via ios, OSX devices (or vice versa), when ios devices use tethering Windows 8.1 PCs or Android devices (or vice versa), as well as when tethering is used between Windows Phone devices and ios or OSX devices.

3.2. Using tcp.window_size values in tcp.syn packets

Fig. 2 shows that TCP window size written in uplink TCP SYN packets from mobile devices can also be used to detect tethering. For this experiment, we tether a Windows 8.1 PC to a Samsung Galaxy Nexus(Android 4.3) phone then collect uplink traffic from those two devices. Checking TCP window size values of TCP SYN packets contained in the uplink traffic, we observe that there are two distinct values as shown in Fig. 2: 65,535 (from Windows 8.1 PC) and 14,600 (from Galaxy Nexus Phone). Table 3 summarizes the observed TCP window size values from the used eight devices of ours.

Table 3. Observed TCP Window Size values in TCP SYN packets from different OS

OS	window_size in tcp.syn packets
Apple ios 7.0	64k
Apple OSX 10.9	64k
Microsoft Windows 8.1 Embedded	64k
Microsoft Window Phone 7.8	8k
Google Android 2.3	5k
Goolge Android 4.1	12k
Goolge Android 4.3	12k

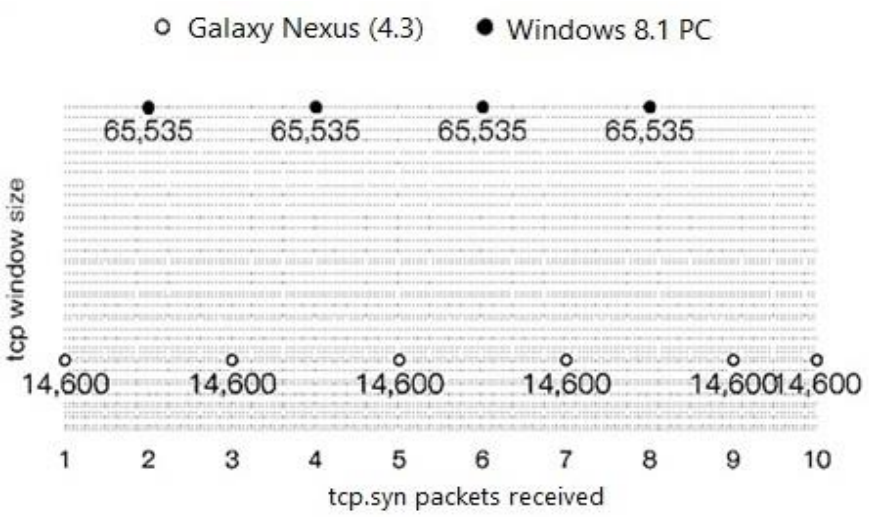

Fig. 2. Observed TCP window size values from uplink TCP SYN packets from Galaxy Nexus(Android 4.3)and Windows 8.1 PC.

From Table 3, we anticipate that it would be feasible to detect tethering when it is between devices with different OS, e.g., between Android 2.3 and Android 4.1 (or 4.3), Apple OSX 10.9 and Android devices, etc.

	Android 2.3	Android 4.1, 4.3	ios 7	Mac OSX10.9	Windows 8.1	window phone 7.8
Android 2.3		2	1, 2	1, 2	2	2
Android 4.1, 4.3	2		1, 2	1, 2	2	2
ios 7	1, 2	1, 2			1	1, 2
Mac OSX10.9	1, 2	1, 2			1	1, 2
Windows 8.1	2	2	1	1		2
window phone 7.8	2	2	1, 2	1, 2	2	

Fig. 3. When IP.ID and/or tcp.window_size features are used: we can detect tethering when used between devices with different types of Operating System (1 and 2 in the figure denotes IP.ID and tcp.window_size values, respectively).

3.3. Using IP.ID values and/or tcp.window_size values

Figure 3 shows that, using either or both of IP.ID values and/or tcp.window_size values, we are able to detect tethering when used between devices with different types of Operating System. We leave the cases where the same OS devices are used as our future work.

4. Conclusions

In this paper, we showed that using uplink IP.ID and/or TCP window size values from mobile devices can be used for tethering detection, particularly when the devices are with different types of Operating System. Yet, these IP.ID and TCP windows size values still can be manipulated or modified by users to evade the detection method. Overcoming this limitation is left as our future work.

Acknowledgements. The authors wish to thank the editor and anonymous referees for their helpful comments for improving this paper. This research was supported by Sangmyung University (2013-A000-0234). Correspondence should be addressed to Prof. Hyun-chul Kim (hkim@smu.ac.kr).

References

[1] Evans, D.: The internet of things: How the next evolution of the internet is changing everything. CISCO white paper (2011)

[2] Cho, B.: System and method for controlling tethering in mobile communication network. Patent, pending no. 10-2010-0105216(2010)

[3] Oh, H: Method for deciding tethering service in communication system and apparatus therefor. Patent no. 10-1361-8230000 (2014)

[4] Wireshark, Go Deep. http://www.wireshark.org

[5] Czajkowsi, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181--184. IEEE Press, New York (2001).

[6] Lee, S., Kim, H.: On Mobile Internet Tethering Detection. *Advanced Science and Technology Letters*, 60 (2014).

[7] Plonka, D., FlowScan: A network traffic flow reporting and visualization tool. USENIX LISA, USENIX, New Orleans (2000).

[8] Douceur, J. and Bolosky, W., A Large-Scale Study of File System Contents. ACM SIGMETRICS, ACM, Atlanta (1999).

[9] Bolosky, W., Douceur, J., Ely, D. and Theimer, M., Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. ACM SIGMETRICS, ACM, Santa Clara (2000).

[10] Lao, D. M. and Shimizu, T., A Method for Discriminating a Signal Peptide and a Putative 1st TM Segment. 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS `01). CSREA Press, Las Vegas (2001).

Received: August 11, 2014