

Appmon: An Application for Accurate per Application IP Traffic Classification

Demetres Antoniadis, Michalis Polychronakis, and Evangelos P. Markatos

Institute of Computer Science

Foundation for Research & Technology – Hellas

{danton,mikepo,markatos}@ics.forth.gr

Abstract—Accurate per-application network traffic characterization is becoming increasingly difficult in the face of emerging applications that use dynamically negotiated port numbers or masquerade their traffic using pervasive, firewall-friendly protocols for bypassing firewall restrictions. At the same time, information about the contribution of different network applications and services to the traffic mix is highly demanded by network administrators for facilitating effective network management and traffic engineering. In this paper we present appmon, a passive monitoring application for per-application network traffic classification. Appmon uses deep packet inspection to accurately attribute traffic flows to the applications that generate them, and reports in real time the network traffic breakdown. Appmon is easy to configure and deploy, and is publicly available as an open source application.

I. INTRODUCTION

Both the research community and network administrators lack of publicly available tools able to distinguish network traffic by the application that generates it. Researchers in the traffic classification area need a reference application in order to evaluate new classification approaches. Most researches tend to build customized tools in order to evaluate their approaches, which may result into inconsistent results among different classification methods. On the other hand, network administrators need information for the applications running on their network.

The emergence of peer-to-peer file sharing, multimedia streaming, and conferencing applications has resulted to a substantial increase in the traffic volume, since they transfer a large amount of data. However, monitoring the traffic generated from such applications is becoming increasingly difficult.

Traditionally, traffic attribution to the corresponding applications is performed using the statically assigned port numbers. Widely used network services, like the Web, Telnet, SSH, and many others, are associated with well-known port numbers which can be used for identifying the traffic related with each application. However, many major new applications, including popular, bandwidth-hungry file sharing applications and widely used video and voice conferencing applications, do not use well-known port numbers. Instead, they allocate and use dynamically negotiated ports. Furthermore, some applications masquerade their traffic using pervasive, firewall-friendly protocols, like HTTP, in order to bypass firewall restrictions and make the identification of their traffic harder. Indeed, several widely used applications like BitTorrent [1]

and Skype [2] can be configured to operate through port 80, which is usually left open even in environments with strict firewall configurations. Nowadays, the assumption that port 80 traffic is solely HTTP Web traffic is hardly true.

It is clear from the above that traditional network monitoring methods for determining per-application network usage are not effective anymore for accurate traffic categorization [3]. Having identified this issue, several researchers have conducted significant work towards alternative ways for network traffic classification. Due to the popularity and high bandwidth demands of peer-to-peer file sharing applications, a significant body of work has focused on the identification and categorization of peer-to-peer application traffic. Initial approaches used deep packet inspection and application signatures for attributing traffic flows to the corresponding applications [4], [5]. Recent approaches identify the applications that generate the traffic either by deriving statistical models for certain protocols [6] or by characterizing the behavior of the host generating this traffic [7].

Motivated by the significance of traffic categorization for effective network management and traffic engineering and aiming at gaining a better understanding of Internet traffic, we have developed appmon, a passive network monitoring application for accurate per-application traffic identification and categorization.

II. TRAFFIC CLASSIFICATION APPLICATION

Appmon passively monitors traffic passing through a monitored link and categorizes active network flows (identified by the 5-tuple) according to the application that generated them.

The classification algorithm uses information from both the packet header and payload. The first step is to check if the packet belongs to an already categorized network flow. Information about the network flows seen so far is stored into a hash table, and allows for a “fast path” processing of subsequent packets of an already categorized flow.

Packets that do not have a matching entry in the hash table are passed down to the next processing level, where each packet is sequentially processed by a set of modules called application trackers. Each tracker is responsible for identifying the traffic of a particular application or protocol. There are three different types of application trackers, depending on method used for classifying traffic: packet inspection trackers, protocol decoding trackers, and header filtering trackers.

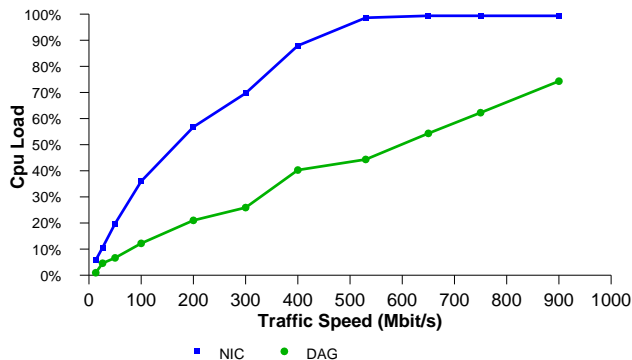


Fig. 1. CPU usage of appmon when tested with various traffic rates

Packet inspection trackers are used for tracking application-level protocols, mainly used in peer-to-peer file sharing applications such as Gnutella [8] and BitTorrent. Each packet inspection tracker searches inside packet payloads for characteristic application messages or binary byte sequences that are used by application protocols. These application messages were selected by extensively reverse-engineering the network traffic of popular file sharing applications, as well as by studying the related work on signature-based traffic classification [4], [5], [9].

Protocol decoding trackers are used for publicly documented application level protocols that use dynamically assigned ports for data exchange; such as FTP.

If none of the above groups of trackers succeeds in identifying a given packet, then the packet is passed to the header filtering trackers. Filtering trackers classify traffic based on packet header information such as identifying predefined registered ports [10] and other protocol information. Filtering trackers are implemented using BPF filters [11].

As we have already discussed, several applications masquerade their traffic using widely used, firewall-friendly protocols, like HTTP, in order to bypass firewall restrictions and make identification of their traffic harder. To avoid potential traffic misclassification due to such tricks, trackers are prioritized, with packet inspection trackers applied first, then the protocol decoding trackers, and finally header filtering trackers.

III. PERFORMANCE

We stressed appmon by sending traffic in various speeds. As Figure 1 implies appmon can process up to 500 Mbit/s without any packet loss when running on a regular NIC interface, while it is able to process all 900 Mbit/s when running on top of the DAG card. The results imply that the application can fully monitor a Gigabit link using a DAG card.

For our second experiment we deployed appmon in a real network environment, aiming at verifying the performance results of the first experiment. Appmon was installed on a sensor at University of Crete, monitoring the incoming and outgoing traffic from the campus to the Internet. The traffic is captured using a DAG 4.2GE passive monitoring card.

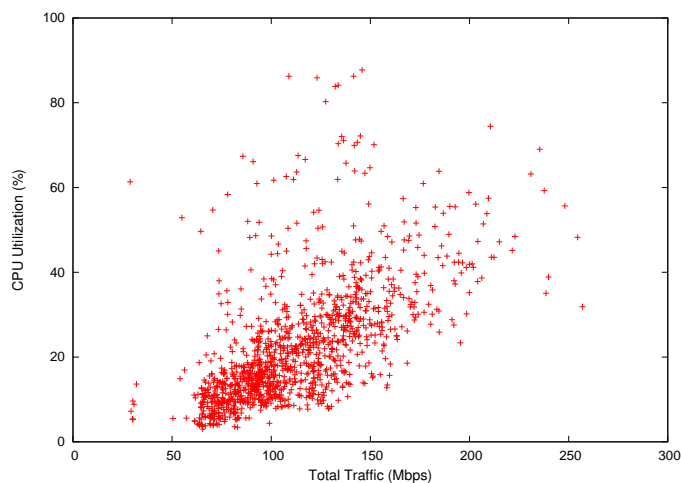


Fig. 2. Appmon CPU Load Vs. Traffic Load while running on a live monitoring sensor at University of Crete for a period of four days. Each point corresponds to a five minute interval.

Figure 2 presents the CPU load (y-axis) of the monitoring sensor as a function of the monitored traffic load (x-axis). Each point corresponds to a five minute interval, computed as the average of the measurements performed every 10 seconds in that interval. The measurement period was four days. Appmon has a steady behavior, since the CPU load increases as the traffic load increases. Some corner cases in which the load is increased significantly while the traffic load is low are probably caused due to the almost simultaneous arrival of many new traffic flows that have not yet been categorized.

IV. AVAILABILITY

Appmon is a free open-source application, available at <http://lobster.ics.forth.gr/~appmon/appmon-sa.tar.gz>. The web interface of some deployed appmon sensors is accessible from http://lobster.ics.forth.gr/~appmon/public_sensors.html.

REFERENCES

- [1] B. Cohen, "BitTorrent protocol specification," *First Workshop on Economics of Peer-to-Peer Systems (P2P03)*.
- [2] S. Skype Technologies, "Skype, from <http://www.skype.com>," 2005.
- [3] A. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," *PAM, March, 2005*.
- [4] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is P2P dying or just hiding," *Proc. IEEE Globecom04*.
- [5] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," *Proceedings of the 13th international conference on World Wide Web*, pp. 512–521, 2004.
- [6] L. Bernaille, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.
- [7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 229–240, 2005.
- [8] G. Kan, "Gnutella," *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, 2001.
- [9] [Online]. Available: <http://protocolinfo.org/>
- [10] "Internet assigned numbers authority." [Online]. Available: <http://www.iana.org/>
- [11] S. McCanne and V. Jacobson, "The BSD packet filter: A new architecture for user-level packet capture," *Proc. Winter93 USENIX Conference*, 1993.