Title: The problem of large-scale application traffic characterization - can we do better?

Authors: Kirk Jon Khu (kjkhu@i2r.a-star.edu.sg),
Periklis Akritidis (periklis.akritidis@cl.cam.ac.uk),
Angelos Stavrou (angel@cs.columbia.edu),
Kostas Anagnostakis (kostas@i2r.a-star.edu.sg)

Application fingerprinting and identification from their network traffic they generate has always been important for ISPs. Recently, the problem of traffic classification has been attracting a lot of attention due to the popularity of novel P2P applications including content streaming and voice over IP (VoIP). The high network connectivity and traffic imposed by these applications have forced many network operators and ISPs towards application profiling with the intention to control the resource consumption on their networks.

There are at least three factors that make application identification both interesting and challenging. First, as new applications emerge at an ever increasing rate, particularly in the P2P space, coming up with signatures for new types of traffic is increasingly time consuming and becomes a bottleneck as signature development, testing and deployment is usually done manually. Second, applications that are generally regarded as undesirable (and are therefore rate-limited or blocked by operators) are starting to employ encryption and other methods for cloaking their identity. Third, some of these applications often consume huge amounts of bandwidth in unpredictable ways that might interfere with proper congestion-controlled traffic. It is conceivable that a very popular application might even bring the network to congestion collapse regimes.

In the Sybil project, we are exploring ways to help automate the development of application signatures. For this purpose, we have designed an extensible tool where different identification methods can be tested and evaluated on network traffic. Our approach is two pronged: we attempt to improve passive traffic fingerprinting using known traffic patterns and validate these patterns using active probing to reduce false positives and improve classification.

For the passive classification, the tool takes as input a set of candidate packet traces containing traffic generated by the application to analyzed, a set of control traces containing traffic other than the application in question. The tool goes through a list of different plugins trying to come up with parameters for different heuristics, and then tests the quality of these heuristics against the control traces.

An optional, but valuable component is an end-point trace collector that produces augmented packet traces containing not just raw packets but also additional information on the process that generated those packets at the end-points. Application name and version information is essential so that application identification can more easily distinguish test traffic with less guess work involved. This component can be installed by the signature developers on hosts that are used to generate test traffic, or can be installed on customers' hosts, possibly under some kind of service subsidy. Although it is understandable that many users may be reluctant to participate in such a scheme, corporate policy or other incentives are likely to make this approach practical.

To avoid false positives and to improve our classification, we employ binary level protocol analysis for known applications. The results of the protocol analysis can be used to profile application behavior under different network conditions such as packet loss or delay.

For instance, we know the protocol and thus network behavior of known applications including known P2P applications such as Skype: when experiencing packet loss, Skype will attempt to compensate by increasing both the packet rate and the packet size (increased Forward Error Correction).

In general, extracting protocol information from the binary can be extremely valuable for application identification. Protocol information can be effectively used to induce network actions (such as loss, packet injection or delay) that can reveal the actual network application boosting our classification performance.

The use of the host-based component can help us keep up-to-date with popular binaries the users use and quantify the behavior of the applications (for this we need to instrument some communication portion of the applications on the host). Using a host-based component will allow us to model the protocol behavior even if the actual network traffic is encrypted. However, where the communications are encrypted, it is unclear how much of this information can actually be translated to network actions.

Therefore, we would like to go beyond naive pattern matching and rate and size based heuristics. The main idea is to manipulate traffic in the form of artificial packet loss, delay and connection filtering in order to drive the network protocol of applications into exhibiting certain behaviors. The assumption here is that even if applications use fully encrypted payloads and other cloaking techniques, internal policies such as those for fault tolerance or congestion control may reveal the type of application at a modest cost. As some of these techniques are key to good application performance, application developers may find it difficult to cloak those behaviors without hurting their own performance.

In the current realization of our approach we have adapted technology used for zero-day worm generation, and specifically a rabin-fingerprint-based algorithm for identifying frequent substrings within network packets, while also including packet size and rate-based heuristics. For string-matching signatures we also adapt a fast offline search algorithm that uses heavy trace indexing for validating candidate signatures against large traces in very little time (e.g., in the order of less than 1 second for a multi-GB trace). The tool is continuously being extended with additional heuristics for more complex scenarios.