Understanding Network Traffic with NetADHICT

Hajime Inoue, Dana Jansens, Abdulrahman Hijazi, Anil Somayaji
Carleton Computer Security Laboratory (CCSL), Carleton University, Ottawa, Canada

Overview

- The complexity of current Internet applications makes the understanding of network traffic a challenging task. By providing larger-scale aggregates for analysis, unsupervised clustering approaches can greatly aid in the identification of new applications, attacks, and other changes in network usage patterns.
- ADHIC (Approximate Divisive HIerarchical Clustering) is an algorithm that clusters similar network traffic together without prior knowledge of protocol structures. Packet similarity is determined through comparisons of substrings within packets at distinguishing offsets.
- ADHIC is notable in that it
  1. produces a hierarchical decomposition of network traffic in the form of a cluster-identifying decision tree,
  2. needs only a small fraction of packets (about 3% in our traces) to generate a decision tree, and
  3. generates a decision tree that can be used to cluster packets at wire speeds (250 Mbit/sec in an unoptimized software implementation).
- We find that ADHIC appropriately segregates well-known protocols, clusters together traffic of the same protocol running on multiple ports, and segregates traffic from applications, such as p2p, that do not use standard ports.
- NetADHICT, our implementation of ADHIC, is available for download at http://ccsl.carleton.ca/software and is licensed under the GNU GPL license.

Why Hierarchical Clustering?

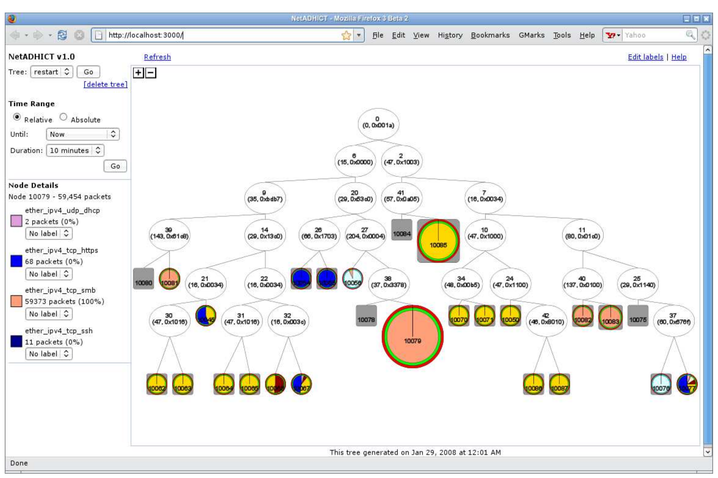
- A hierarchical representation of network traffic has two key advantages:
  - A hierarchy naturally captures the "nesting" of layered protocols, e.g. that UDP and TCP packets both are examples of IP packets
  - Hierarchical representations aggregate low-level concepts into higher level structures, facilitating "high level" views that can be refined as necessary.
- Multiple machine learning approaches have been proposed for classifying packets by application type [2, 4]. Past work identifies traffic as belonging to a small set of pre-defined classes, rather than to arbitrarily large set of hierarchical clusters.
- We cluster packets (rather than classify them) because we wish to capture the commonalities of novel, unknown network protocols and usage patterns. In particular, we want to find *sub-protocol* patterns.
- Others have clustered packets using header information [1]; however, many patterns of traffic, including peer-to-peer networks, self-propagating malware (worms), and flash crowds are more properly distinguished by patterns in payloads, not headers.

Approximate Divisive Hierarchical Clustering

- Clusters are identified by sets of (p,n)-grams.
  - $n-$gram: n consecutive bytes within a packet
  - $(p,n)-$gram: $n-$gram at position p
- ADHIC incrementally learns hierarchical clusters through two operations:
  1. Splitting: If a cluster gets too big, find a (p,n)-gram that matches about half of its packets.
  2. Deletion: If a cluster has too little traffic matching it, delete it.
- ADHIC learns clusters using samples of observed traffic; all traffic is then "classified" using the learned (p,n)-gram tree.
- Leaf clusters are displayed using pie charts that indicate the protocol (port) of classified packets.

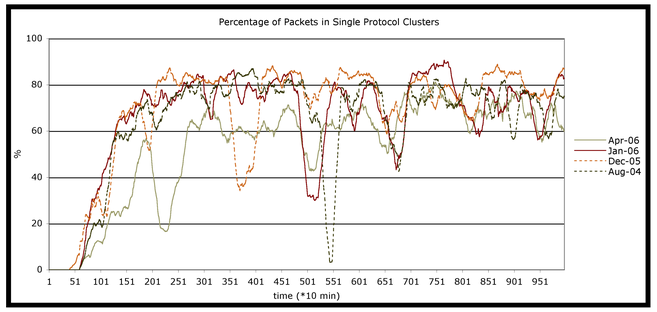
NetADHICT

- NetADHICT allows a user to analyze a live traffic stream and watch the tree evolve over time along with the network's traffic. It also allows for offline analysis of previously viewed traffic, or for the analysis of libpcap-formatted packet captures.
- The NetADHICT interface facilitates interactive exploration of the tree, and allows a user to apply user-defined labels to positions in the tree, giving clearly visible semantic meaning to the clustered traffic.



Results

- Tests were conducted using captures from a small production network, with the classification tree being updated every 10 minutes.
- Observed clusters were highly correlated (∼80%) with well-known ports, even though NetADHICT rarely used port-matching (p,n)-grams.
- When NetADHICT clustered together traffic matching multiple ports, there were often other significant commonalities. For example, a web server on a non-standard port was grouped with standard web traffic.
- In tests with simulated static and polymorphic worm traffic, these packets were segregated from other traffic. NetADHICT was also able to properly segregate BitTorrent, a P2P filesharing protocol.
- Perhaps most remarkably, NetADHICT's performance was not significantly degraded (and sometimes improved) when we *excluded headers* from the packets.



Future Work

- Study bigger datasets from larger networks.
- Study the characteristics of network traffic (e.g., the Zipf-like distribution of (p,n)-grams) and relate them to other measures of network behavior.
- Refine NetADHICT's web interface.
- Study ADHIC in the context of dynamic traffic management, the problem domain that was the original motivation for this work [3].

References

[1] C. Estan, S. Savage, and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of ACM SIGCOMM*, 2003.

[2] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of protocol inference. In *Proceedings of ACM IMC*, 2006.

[3] A. Matrawy, P.C. van Oorschot, and A. Somayaji. Mitigating network denial-of-service through diversity-based traffic management. In *Applied Cryptography and Network Security (ACNS'05)*, pages 104–121. Springer Science+Business Media, 2005.

[4] N. Williams, S. Zander, and G. Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *ACM SIGCOMM Computer Communications Review*, October 2006.