netAI - Network Traffic based Application Identifier

Nigel Williams
Centre for Advanced Internet Architectures (CAIA). Technical Report 060410E
Swinburne University of Technology
Melbourne, Australia
niwilliams@swin.edu.au

***Abstract*— netAI is a tool designed to classify and display network traffic flows in real-time using Machine Leaning (ML) techniques. This technical report provides and overview of the netAI software package and demonstration testbed.**

Keywords- **netAI, Traffic Classification, Machine Learning.**

I. INTRODUCTION

netAI is a tool designed to classify and display network traffic flows in real-time using Machine Leaning (ML) techniques. It has been developed as part of the DSTC [1] project at CAIA.

netAI uses flow-level statistics collected by NetMate [2] or a similar networking meter to predict the application responsible for a given network flow, and provides an alternative to port-based and payload-based classification. Flow predictions appear on screen with additional information such as source/destination hosts, duration and bytes transferred. This information is updated at a user specified time interval.

This paper briefly outlines the netAI software and describes the testbed that was used to generate test data. Further up-to-date information, including man-pages and a getting started guide (covering installation, dataset creation, ML basics) can be found at the netAI homepage [3].

The following are the main features of netAI. As netAI is still under development it is likely additional features will be added.

- Read packet data from live network interfaces or tracefile (tcpdump or Endance format)
- Classify flows on completion or use interim statistics to monitor active flows.
- No restriction on type and number of features used
- Feature extraction and ML based flow classification can be run on different machines - feature extractor supports data export via UDP or TCP
- Flexible packet classification and filtering thanks to NetMate
- A large number of machine learning algorithms can be used thanks to WEKA [4].

II. NETAI DESIGN

netAI consists of a separate GUI and command line tool. The command line tool is useful for constructing classification models and for offline classification of packet traces (for purposes such as trend analysis). The GUI version allows real-time monitoring and prediction of network flows. ML classification is performed using the WEKA library.

A. *netAI componants*

The main componants of the netAI system are the Classifier host and the Network Meter host. While it is possible for the network meter and classifier to be integrated onto a single host, the remainder of the document will consider them as seperate entities.

The role of the Network Meter host is to monitor flows on the network and produce statistics that are then passed on to the classifier host. The classifier host evaluates these statistics against a stored ML classification model to predict the application generating the flow. The relationship between the componants is shown in Figure 1.

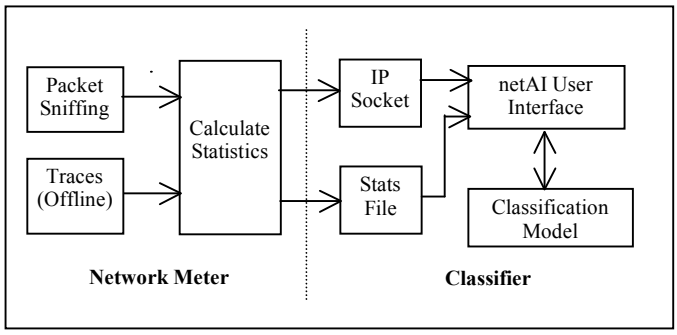

Figure 1: netAI component architecture

Flow statistics are passed between the Network Meter and the Classifier via TCP or UDP. Statistics can also be written directly to file (stats file) to be read by netAI at a later stage.

B. *Software*

netAI has been written in Object Oriented Java and has been tested on Linux and Windows. The modification or additional of functionality should require minimal effort.

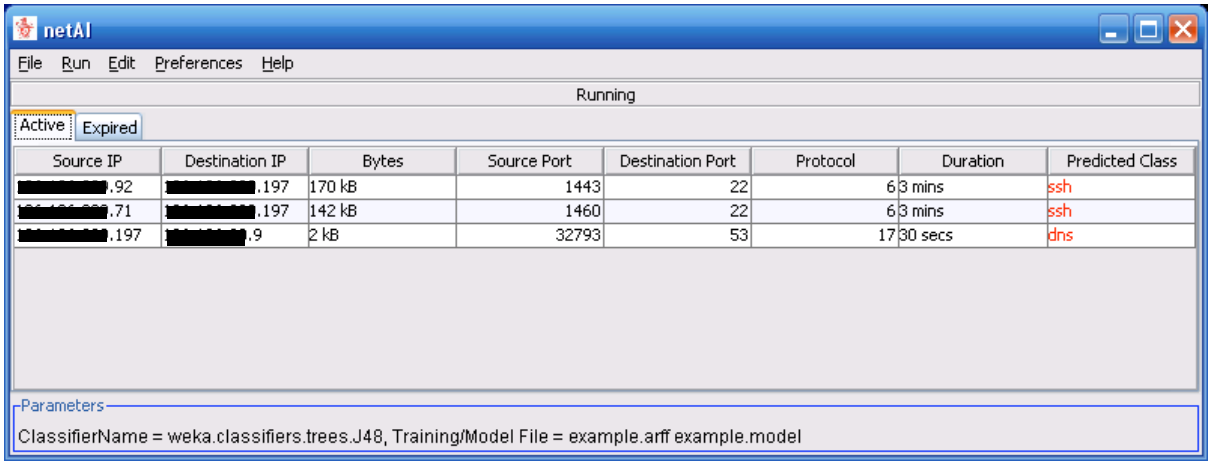

Figure 2: netAI GUI

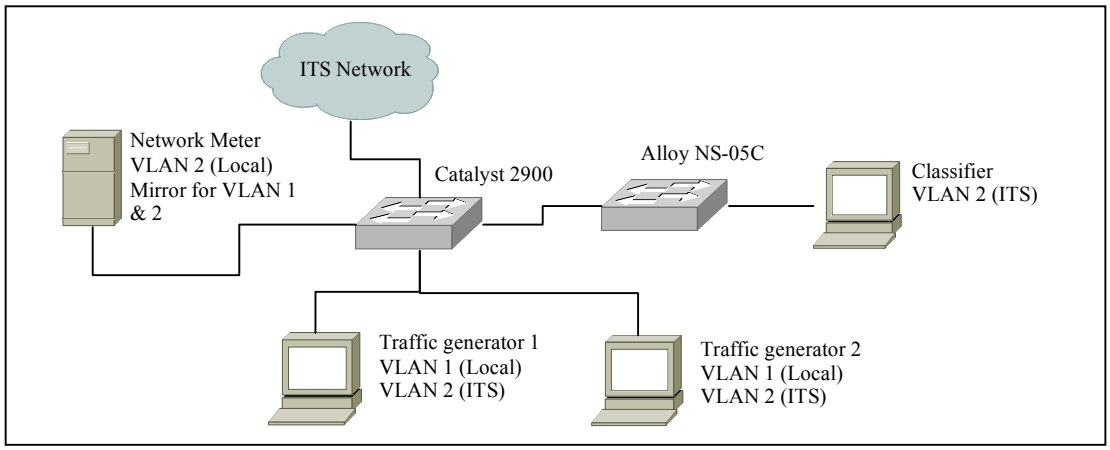

Figure 3: netAI development testbed

The netAI interface (Figure 2) essentially operates as an intermediary between a network meter and a ML algorithm library. As such there is no limit to the number of statistical features or algorithms that can be used to classify flows, as these are determined by the meter and algorithm library.

III. Testbed Hardware and Topology

This section describes the testbed used to test and generate data during netAI development.

A. *Hardware*

Four Intel workstations formed the basis of the testbed network, each assigned a different role - Network Meter, Classifier and traffic generator (2). The hardware configuration of each system is shown below:

Network Meter

- CPU: Pentium 4 3.4Ghz
- NIC: On-board Intel 82562EZ Fast Ethernet, Intel 82541PI Gigabit Ethernet PCI (2)
- RAM: 4GB
- Operating System: SuSE Linux 9.3 32-bit Edition

Classifier

- CPU: Pentium 4 3Ghz
- NIC: On-board Marvell 8001 Gigabit Ethernet
- RAM: 512MB
- Operating System: Windows XP Home SP2

Network Clients (2)

- CPU: Celeron 2.8Ghz
- NIC: On-board Intel 82562EZ Fast Ethernet, Intel 82541PI Gigabit Ethernet PCI
- RAM: 512MB
- Operating System: SuSE Linux 9.3 32-bit Edition

The hosts were connected via two 100Mbps switches, a Cisco Catalyst 2900 and an Alloy N505C 5 Port Nway Mini Switch.

B. *Topology and configuration*

Figure 3 shows the topology of the testbed. Two VLANs were implemented on the Catalyst 2900 using the steps outlined at [6]. Two port monitors (port mirrors) were also configured on the switch, one for each VLAN.

VLAN 101 connected network clients using private 192.168.0.0 addresses (Local), while VLAN 102 connected all hosts to the Swinburne University LAN

(ITS). Spanning Tree Protocol (STP) was disabled for each of the VLANs using the command (in terminal configuration mode):

```
# no spanning-tree vlan <VLAN ID>
```

Two ports were assigned to VLAN 101, while four were assigned to VLAN 102, using the following command (from interface configuration):

```
# switchport access vlan <VLAN ID>
```

Cisco Discovery protocol was disabled on the switch using the `no cdp run` command.

The Network Meter host was connected to the ITS network (VLAN 102) using the on-board Intel Ethernet port. The PCI Ethernet cards were each assigned `192.168.0.X` addresses and connected to each of the VLAN mirror ports on the Catalyst 2900 switch.

The Classifier host was connected to VLAN 102 via the Alloy switch.

The on-board Intel Ethernet port of each traffic generator was connected to VLAN 102. The PCI Ethernet cards were assigned the addresses `192.186.0.4` and `192.186.0.5` and connected to VLAN 101.

C. *Software Configuration*

VLAN 101 was used to generate additional traffic for applications that could not be run on the ITS network. These were the peer-to-peer programs BitTorrent (BT) and Direct Connect (DC). Using these applications required that one traffic generator host act as server and the other as client.

The Azureus BT client contains built-in tools for generating and hosting torrent files for sharing. This can be done using a wizard within the client and the instructions are located at [7].

OpenDCd [8] was used to create a DC 'hub' to which both traffic generators could connect and transfer files. The server is automatically configured on execution and the default settings were used. The client and server programs could be run on the same machine. The Valknut [9] Linux DC client was used to connect to the DC server hub.

The remaining applications used to generate traffic such as SSH and HTTP were run over the ITS connection and did not require additional configuration.

IV. CONCLUSIONS

This paper provides a brief overview of the netAI software and the testbed used to generate traffic during development. In depth and up-to-date documentation regarding the preparation of classification models, an overview of popular ML algorithms and step-by-step instructions can be found at the netAI homepage [3].

ACKNOWLEDGEMENTS

This paper has been made possible in part by a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley.

REFERENCES

[1] Dynamic Self-learning Traffic Classification based on Flow Characteristics (DSTC), http://caia.swin.edu.au/urp/dstc/ (as of April 2006)
[2] NetMate, http://sourceforge.net/projects/netmate-meter/ (as of April 2006).
[3] netAI Homepage, http://caia.swin.edu.au/urp/dstc/netai/ (as of April 2006)
[4] WEKA 3.4.4, http://www.cs.waikato.ac.nz/ml/weka/ (as of April 2006).
[5] ARFF, http://www.cs.waikato.ac.nz/~ml/weka/arff.html (as of April, 2006)
[6] Creating and Maintaining VLANs, http://www.cisco.com/univercd/cc/td/doc/product/lan/c2900xl/29_35xu/scg/kivlan.htm (as of April 2006).
[7] Host and share your torrents, http://azureus.aelitis.com/wiki/index.php/HostAndShareYourTorrents (as of April 2006)
[8] OpenDcd homepage, http://opendcd.sourceforge.net/ (as of April 2006).
[9] Valknut Homepage, http://dcgui.berlios.de/ (as of April 2006).