Evaluating The Performance of ANGEL Flow Classifier

Thuy T.T. Nguyen
Centre for Advanced Internet Architectures, Technical Report 070228H
Swinburne University of Technology
Melbourne, Australia
tnguyen@swin.edu.au

***Abstract*—This technical report gives details on the results obtained from evaluating the ANGEL Flow Classifier (FC) performance. Evaluation metrics comprise of accuracy, timeliness, stability, processing speed, and the efficiency of hardware resources usage.**

I. Introduction

The ANGEL Flow Classifier (FC) is one of the three main components of the ANGEL architecture. The primary tasks of the Flow Classifier are:

- Classify flows based on the packet statistics provided by any Flow Meters (FM) within the system
- If a new flow is classified as requiring priotisation or changes its priority level, signal the Client Manager (CM) with this information
- Purge stored flow information when that flow terminates

With these responsibilities, requirements for the FC include:

- *Accuracy*
  This performance factor measures the accuracy of the FC in detecting game flows based on the information provided by the FM.
  Accuracy is measured in terms of Recall and Precision rates. These metrics are often used to evaluate the performance of Machine Learning (ML) classification algorithms.
  If a classifier is trained to identify members of class X then these metrics are defined as:
  - **Recall:** the proportion of class X's instances which are correctly classified as belonging to class X.
  - **Precision:** the proportion of the instances that truly belong to class X amongst all those classified as class X.

  Both metrics range from 0 (bad) to 100% (optimal). These metrics can be related to the traditional networking/security metrics of False Negatives and False Positives. If we define False Negative as the percentage of class X's instances which are incorrectly classified as not belonging to class X, and False Positive as the percentage of the instances which are not belonging to class X among all those classified as class X. In this case Recall is (1 - False Negatives) and Precision is (1 - False Positives).
  For deployment of the classifier within ANGEL, it is desired to have high Recall and Precision rates (of $>= 95\%$) i.e. low False Positive and False Negative rates ($<= 5\%$) of all the flows being classified.
- *Timeliness*
  ANGEL is a real-time system. The FC therefore must reach the decision quickly enough so that classification results can be sent promptly to the CM for appropriate QoS actions at the CPE.
  The timeliness of flow classification is affected by a number of factors:
  - FM processing time: Time taken by the FM to extract packet information, and to bundle packets for delivery to the FC.
  - Packet inter-arrival time: The FC needs to wait for a full window of packets to calculate statistical parameters and perform classification. This waiting time is affected by the packet arrival rate of the application. For example, with ET traffic during actual game play, the packet rate for a bi-directional flow (client to server and server to client) is approximately 50PPS. In the current FC implementation, we

From February 2007 and July 2010 this report was a confidential deliverable to the Smart Internet Technologies CRC. With permission it has now been released to the wider community.

use a classification window of 25 packets. This would result in 0.5 seconds for the FC to fill the window prior to making a classification decision.
  - FC processing time: Time taken by the FC to do statistical computation for the window of packets, and to apply the ML classification model to make a classification decision.

  For the purpose of evaluating the performance of the FC we primarily focus on the FC processing time component. It is acceptable if the classifier can identify traffic flow within 1 second of receiving a flow's packets at the FC.
- *Stability*
  As the flow classifier will monitor the flow classification for its lifetime (re-classify at every checkpoint interval) we would like to minimise the flapping in flow's classification as much as possible during its lifetime, so that QoS implementation at the CPE will be more stable, and to minimise the data transfers from the FC to the CM and subsequently to the customer CPE. The ideal case is the flow state is maintained stably for the flow's duration.
  This metric will be measured by the number of flow state changes per single game flow for all tested game flows for the duration of the flow.
- *Scalability*
  This metric measures the FC's performance within the constraints of physical resources. We need to see how quickly the FC deals with a large number of concurrent flows and Packet Per Second (PPS) rates.
- *Efficiency of hardware resources usage*
  This measures CPU and Memory usage when the FM and FC capture and classify live traffic in real time, especially with high packet and flow rates.

In this report, we outline the results obtained from testing the performance of ANGEL FC with the above evaluation metrics. As the project's scope is limited to a prototype, the classification model was built as a proof-of-concept only. Tuning methods and parameters are included in the report for future work on the classifier model optimisation if desired.

II. Traffic Classification Approach

Traditional techniques for the identification of Internet applications are typically based either on the use of well known registered port number, or on payload-based protocol reconstruction. However, applications can use un-registered ports or encryption to obfuscate packet contents. Further governments may impose privacy regulations constraining the ability of third parties to lawfully inspect packet payloads. State-of-the-art approaches classify traffic by learning and recognising statistical patterns in externally observable attributes of the traffic (such as packet lengths and inter-packet arrival times).

The ANGEL FC employs the technique of IP traffic classification using Machine Learning (ML). An introduction to ML concepts and the deployment of supervised ML algorithms in the ANGEL FC are addressed in [1] and in [2].

The ANGEL FC is implemented using a simple Naive Bayes algorithm [3]. The algorithm provides an approach for classification based on probabilistic knowledge. It is designed for use in supervised classification, in which the goal is to accurately predict the class of unseen data using a classification model built on sample training instances.

The algorithm can be summarised briefly as following. Let C be the random variable denoting the class of an instance and let $X = X_1, X_2, X_n$ be a vector of random variables denoting the observed attribute values. If c is a particular class and x is an instance to be classified. The algorithm makes statistical conclusions about the probability of instance x belonging to a class c based on:

- the conditional probability of observing the occurrence of each class's instance in the training set (so-called the posterior probability and is denoted by P(C=c))
- and the probability for the instance x given c.

The calculation follows the Bayes' rule:

$$p(C = c|X = x) = \frac{P(C = c)P(X = x|C = c)}{p(X = x)} \quad (1)$$

Based on the outcome the classifier predicts the most probable class.

The algorithm relies on two important simplifying assumptions:

- the instance's attributes are independent given the class and within each class
- the values of numeric attributes are normally (or Gaussian) distributed.

For detail implementation information of the model, the readers are referred to [1] and [4].

III. Testing and Evaluation Method

There are a number of component tests before the overall performance of the FC is evaluated. Figure 2

illustrates the overall interaction between the ANGEL FC and other ANGEL components. They include the flow statistical properties calculations (feature calculations) and the classification model.

Fig. 1. Overall Architecture

Figure 2 illustrates the primary tasks of the ANGEL FC. The Flow Classifier was originally designed to use multiple processes to potentially take advantage of a distributed platform implementation in a scalable implementation. However, initial testing indicated problems with the underlying IPC (Interprocess Communication) messaging scheme. The problem description and solution are detailed in [4]. For testing purposes the FC was re-coded to use a single thread for all flow classification purposes rather than multiple individual processes. In doing so, the FC implementation is under heavier load than in the original implementation. As such, our test results indicate the lower bound of the FC's performance.

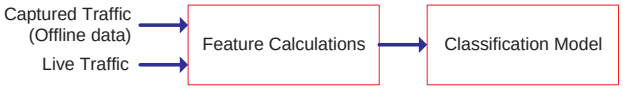

Fig. 2. Components of ANGEL FC

We start the testing process with a performance evaluation of the Naive Bayes model built by the FC implementation. To test the data processing component, we test the FC while the FM reads off-line traffic from pre-recorded tcpdump files. And finally, we test the FC while the FM captures live traffic from the network interface.

The testbed setup for each test is illustrated in Figure 3, Figure 4 and 5.

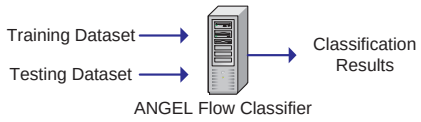

Fig. 3. Test ANGEL classification model only

Test components specifications and configurations can be found in Table I. PCAP buffer sizes (default and maximum) are set to 512KB and 1MB respectively as suggested in the FM Performance Test Report.

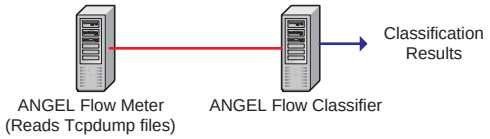

Fig. 4. Test FC with offline traffic analysis

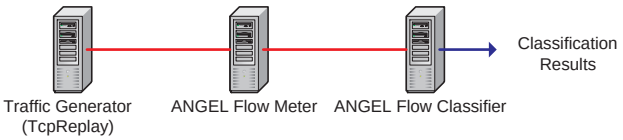

Fig. 5. Test FC with live traffic capture

A. *Training and Testing Dataset*

The classification model is built using the Synthetic Multiple Sub-Flow Pairs approach [5] [6]. We train the ML classifier using statistical features calculated over multiple short sub-flows and its synthetic pair extracted from full-flow generated by the target game application and a number of common interference traffic. The knowledge gained from this training process is classification rules so that the classifier can differentiate game traffic and other traffic in real-time and operating networks.

For testing purpose, we utilise a hypothetical classification scenario where real-time flows belonging to online multiplayer games must be detected and trigger the priotising process. The classifier should also recognise other traffic from a number of common Internet applications and differentiate them from the game traffic.

The applications we chose to test include:

- Game Traffic: Wolfenstein Enemy Territory (ET) [7] and Half-Life 2 Death Match (HL2DM)
- Other Traffic: Web (HTTP), DNS, P2P (Kazza and Bittorent), SMTP

The training dataset consists of game traffic of a month-long trace collected during May 2005 at a public server in Australia [8]. Our interfering traffic came from a 24-hour trace collected by the University of Twente, Germany, on February 6th 2004 [9] at an aggregated 1Gbps link.

The testing dataset is specifically constructed for each different test as described in the next section.

B. *Testing the Classification Model*

To test the functionality of ANGEL classification model, we choose to compare the model built by ANGEL implementation to the model built with the WEKA implementation [10] [11]. WEKA is a free, open-source ML and data mining toolkit in Java, implemented

TABLE I
TEST COMPONENT SPECIFICATIONS/CONFIGURATIONS

Test Component	Detail Specifications/ Configurations
Flow Meter	Intel Pentium 4 3.00GHz with Hyper-Threading, 1GB (2 x 512MB) DDR2 533 RAM, Seagate ST380817AS 80GB SATA HDD, Asus P5LD2-VM motherboard, Running FreeBSD 6.1
PCAP	lipcap release version 0.9.5
PCAP Buffer Setting	sysctl net.bpf.bufsize=524288, sysctl net.bpf.maxbufsize=1048576
Angel_FM	Angel_FM as of 26 Oct. 2006
Traffic Generator	Intel Pentium 4 3.00GHz with Hyper-Threading, 1GB (2 x 512MB) DDR2 533 RAM, Seagate ST380817AS 80GB SATA HDD, Asus P5LD2-VM motherboard, Running FreeBSD 6.1
TcpReplay	version 2.0
FLow Classifier	Intel Celeron 2.8GHz, 1GB (2 x 512MB) DDR2 533 RAM, Seagate ST380817AS 80GB SATA HDD, Asus P5LD2-VM motherboard, Running FreeBSD 6.1

and made available by the university of Waikato, New Zealand.

With the same training dataset, we calculate flow features by modifying the framework of the Netmate tool [12] and build classification models using both ANGEL and WEKA architectures. We compare the two models in terms of model build time, CPU usage and memory usage.

We use the same dataset to test the two models. The dataset consists of ET traffic of a month-long trace collected during September 2005 at a public server in Australia [8]. Our interfering traffic came from a 24-hour trace collected by the University of Twente, Germany, on February 7th 2004 [9] at an aggregated 1Gbps link. (Flow features are calculated by modifying the framework of the Netmate tool [12])

We compare the models performance in terms of accuracy, processing speed, CPU and memory usage.

C. Testing the Performance of the FC with Offline Traffic Analysis

To evaluate the performance of the ANGEL classification model (built in the preceded test) with offline data classification, we modify the FM to read directly from pre-recorded Tcpdump files and pass the captured packet information to the FC for traffic classification. The FM reads the tcpdump files as fast as it can, however, packet's original properties e.g. packet time-stampings are preserved.

The tcpdump files consists of ET traffic collected over a LAN network by the SONG project [13] and Web, DNS, P2P (Kazza and Bittorent), SMTP extracted from a 6-hour trace collected over an aggregated 1Gbps link by the University of Twente (loc4-20040207-2001) [9]. As payloads were missing we inferred application type from the port numbers (judged an acceptable approach because our primary criteria for interfering traffic is that it was not ET).

We evaluate the performance of the FC in terms of Accuracy, Timeliness, Stability, Processing Speed and Robustness. These evaluation metrics are defined in section I above.

D. Testing the Performance of the FC with Live Traffic Capture

To evaluate the performance of the ANGEL classification model (built in the preceded test) with real-time data capture, we use TcpReplay tool [14] to re-play pre-recorded tcpdump files at the original capture rate. The FM is used to capture the traffic for classification by the FC.

For game traffic, we use a number of short traces of ET and HL2DM collected over a LAN network by the SONG project [13]. We also emulate the live capture of Kazaa, SSH, Email and DNS traffic from the same trace as in the Offline Traffic Analysis test.

IV. TEST RESULTS AND ANALYSIS

A. The Classification Model

Firstly we compare the model build time between ANGEL implementation and WEKA implementation. We use a training dataset consisting of sample data from 4 selected sub-flows as specified in [5] [6]. Each sub-flow consists of 25 consecutive packets extracted at different points during the original data flow lifetime. They were selected using ML clustering algorithms such that the classifier has the sample of (desirably all) possible traffic characteristics even when the flow statistical properties vary over its lifetime [5]. It enables the classifier to detect and monitor flow classification state in real-time using classification windows as small as 25 packets.

Results are tabulated in table II, ANGEL builds the models more quickly (takes approximately 60% of time needed by the WEKA implementation).

We next compare how accurately the models classify test datasets. Figure 6 shows the Recall for ET traffic as classified by WEKA and ANGEL models. M represents

TABLE II
MODEL BUILD TIME

Sub-Flows	Total Instances	ANGEL (sec)	WEKA (sec)
4	22,320	3	4.55

the number of packets that the classifier might miss from the beginning of the original game flow. As can be seen, at the early stage of game flows, Recall by the WEKA model is higher than the ANGEL model (by approximately 2%). As the classifying window moves beyond the early state (M >= 2000), the models have similar Recall.

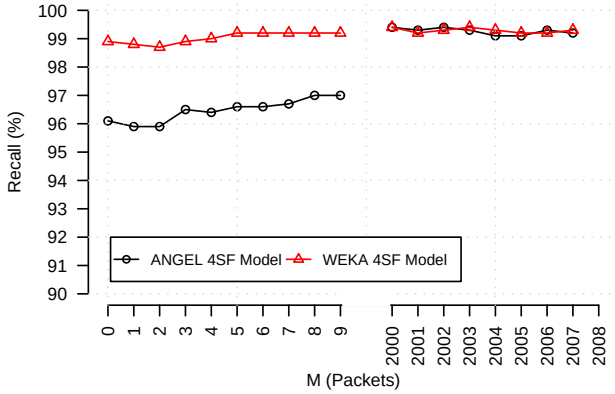

Fig. 6. Recall for WEKA and ANGEL models

Figure 7 shows the Precision for ET traffic while being classified by both ANGEL and WEKA models. Both models demonstrate similar results in Precision of 97.5-98%.

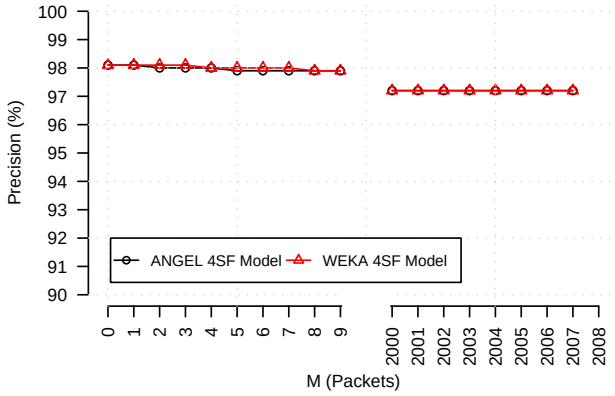

Fig. 7. Precision for WEKA and ANGEL models

In terms of accuracy, though slightly worse than the WEKA model, ANGEL classifier model meets the requirement of accuracy >= 95%.

Next, we compare classification speed of WEKA and ANGEL models while testing the same dataset. As shown in Table III, the ANGEL trained model classifies flows approximately 40% faster than the WEKA model when trained with the same data.

TABLE III
CLASSIFICATION SPEED

Sub-Flows	Total Instances	ANGEL (classifications per sec)	WEKA (classifications per sec)
4	147,249	29,499	21,036

Figures 8 and 9 show CPU and Memory usage when classifying the test dataset by both ANGEL and WEKA classification models. The ANGEL classifier model has similar processing requirements to the WEKA model, at the same time as requiring far fewer memory resources (approximately 12% of memory resource used by the WEKA model).

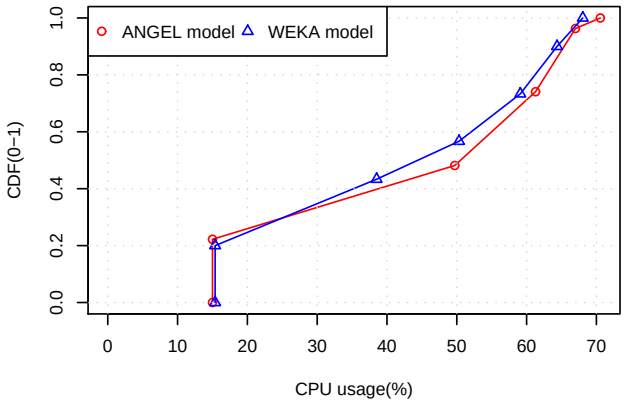

Fig. 8. CPU usage for models trained with 4SF

In summary, the ANGEL classifier implementation is better than WEKA in terms of model build time, classification speed and memory usage. It is slightly worse than the WEKA model in terms of accuracy. CPU usage of both models are comparable.

B. Performance of the FC with Offline Traffic Analysis

This section reports on the performance of the FC when classifying offline traffic. Statistical properties of the testing traces are summarised in Table IV.

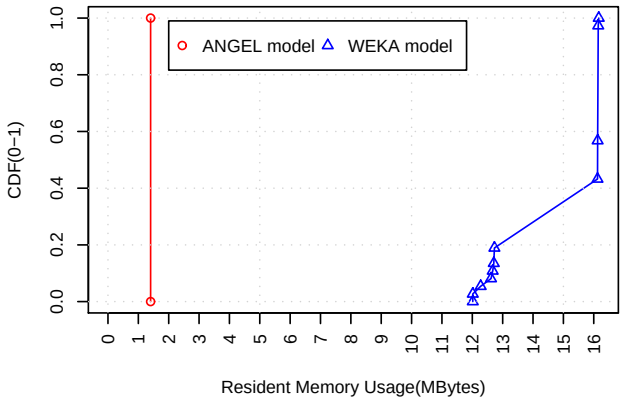

Fig. 9. Resident memory usage for models trained with 4SF

TABLE IV
TEST DATATRACE PROPERTIES

Data Trace	Number of Flows	Number of Packets	Trace Size
ET - 4 players	4	20,000	1.1M
ET - 5 players	5	25,000	1.4M
ET - 6 players	6	30,000	1.7M
ET - 7 players	7	35,000	1.9M
ET - 8 players	8	40,000	2.2M
ET - 9 players	9	45,000	2.5M
Kazaa	7,887	279,334	19M
Bittorrent	1,925	63,243	4.3M
Web	32,215	600,545	41M
SMTP	133	1,283	93K
DNS	1,196	2,818	193K
SSH	80	27,768	2.1M

1) Game Traffic: **Accuracy**

For ET traffic, we test six traces sourced from the SONG database, with a combined total of 39 game flows. Using the ANGEL model built in the previous test, we classify the traffic flows sent by the FM. The FM reads and process packet properties from the tcpdump files. Alhough the tcpdump files only contain ET traffic, they were collected in real-time, from an operating network. As such the flow statistical properties of inter-leaving flows are preserved. We evaluate the accuracy of the classifier in terms of Recall.

A test script has been written so that it keeps records of the flow hashes which have been identified as game flows, as well as flapping results (changes in classification) during the flow lifetime (as the classifier re-classifies traffic flow every window of 25 packets - the classification result might fluctuate during the flow lifetime).

Figure 10 shows the final classification result for each game flow in the trace. It shows that the classifier correctly identified all 39 game flows (Recall of 100%).

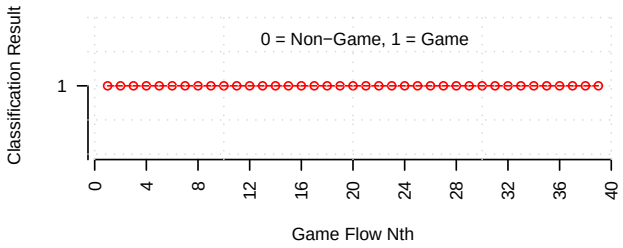

Fig. 10. ET Accuracy

Timeliness

Since we are reading from tcpdump files, the packet information is extracted and sent to the FC faster than the actual data speed in real networks. Figure 11 shows how long it takes the classifier to identify a game flow from the time the FC receives the first packet of each individual flow. For all 39 flows, it takes less than 50msec for the classifier to detect the game traffic. (Please note that in this test, the FM reads packet's information from a tcpdump file, so it excludes the packet inter-arrival component discussed in section I. However, the FM and FC processing components still exist. So the result provides an upper bound on the FC processing time for classification decisions).

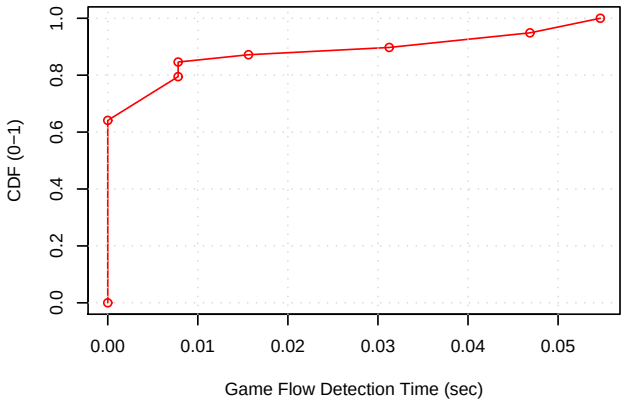

Fig. 11. ET Timeliness

Stability

15 out of 39 flows (38%) flapped state from one to six times during their flow lifetime (see Figure 12).

Figure 13 shows the flapping duration for each of these 15 flows. With high speed of reading from a tcpdump trace, maximum flapping duration is 200msec.

2) Other Traffic: We investigate the performance of the FC with other Non-Game traffic. It is desirable that

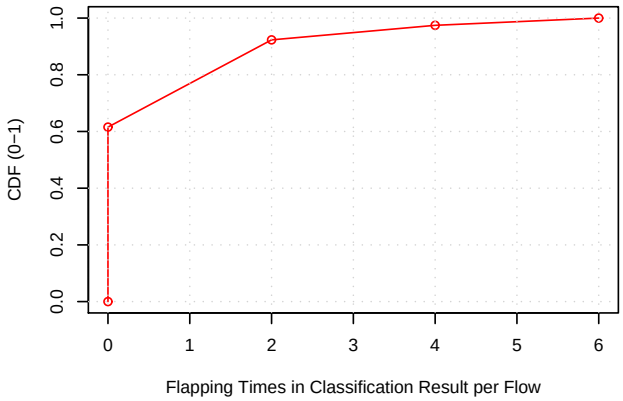

Fig. 12. Changes in classification result (flapping) for ET traffic

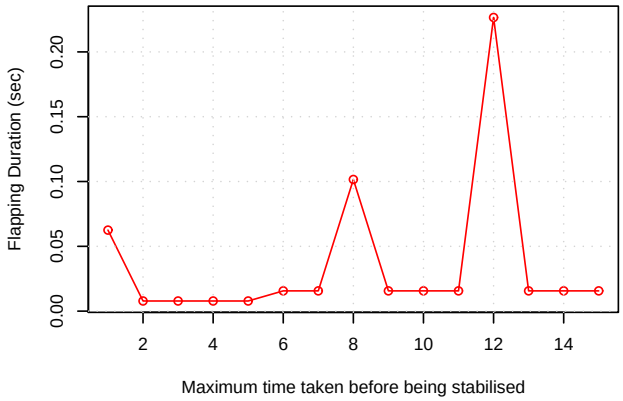

Fig. 13. ET Flapping Duration

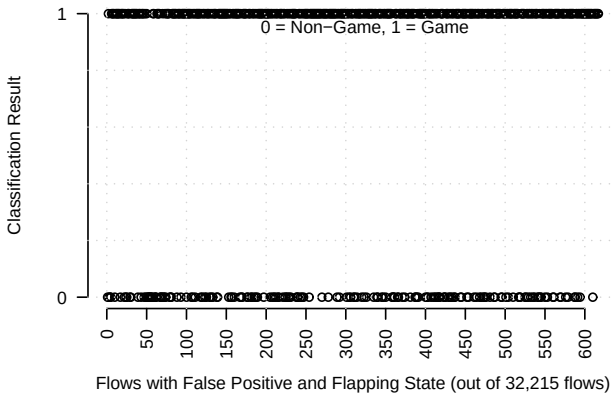

Fig. 14. Web Traffic - False Positive and Flows with Flapping Classifications

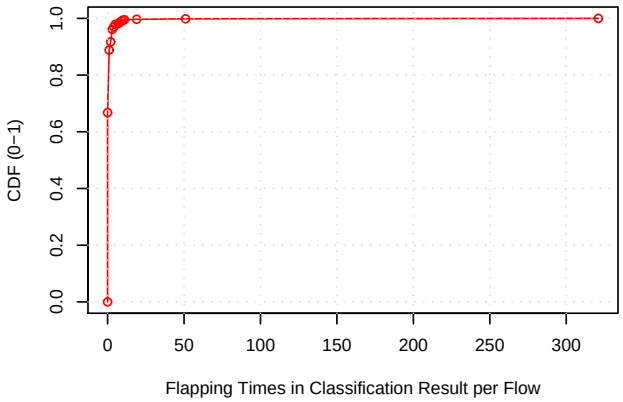

Fig. 15. Web Traffic - Flapping Times

the FC correctly identify Non-Game flows and therefore will not notify the CM about these flows. As for the above test, we let the FM read from Tcpdump traces that contains Web, Kazaa, Bittorent, SSH , DNS and SMTP traffic. We report results for each type of application in the followed sections.

Web Traffic

Figure 14 shows a total of 617 flows (out of 32,215 flows) (1.9%) that were falsely classified as game traffic at their initial classification. However, 439 flows (71%) of these had flapping classification results, before being finally detected as Non-Game traffic. Hence the final false positive rate for web traffic is 0.55% (i.e. Precision rate of 99.5%).

Figure 15 shows the amount of classification flapping per flow. 90% of these flows had $<= 3$ classification flaps during their lifetime. One exceptional flow had flapped 322 times during its lifetime. (It is worth noting here that we consider flows with destination port of 80 to be Web traffic, but actually it can be other types of traffic tunnelled through this port).

Figure 16 shows the duration of classification flapping per flow, for flows where the classification flapped more than once.

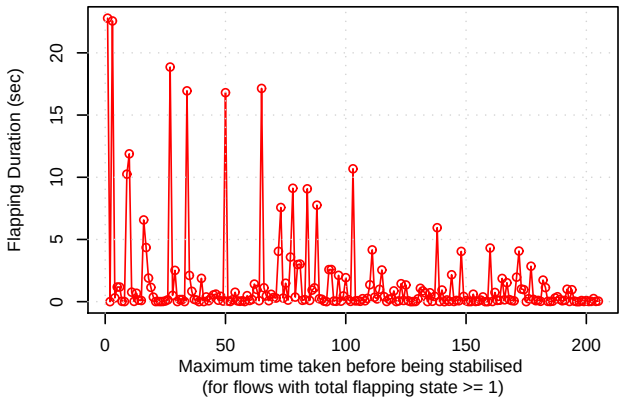

Fig. 16. Web Traffic - Flapping Duration

Kazaa Traffic

The first graph of Figure 17 (from left) shows a total number of 91 flows (out of 7,887 flows) that were falsely

classified as game traffic at their initial classification. However, 66 flows (72.5%) of these flows had flapping classification results before being finally detected as Non-Game traffic. Hence the final false positive rate for Kazaa traffic is 0.32% (i.e. Precision rate of 99.68%).

The middle graph of Figure 17 shows the amount of classification flapping per flow. 90% of these flows had $<= 8$ classification flaps during their lifetime. One flow flapped a total of 51 times during its lifetime.

The last graph (from left) shows the duration of classification flapping per flow where the classification flapped more than once. Again, these results are with the speed of reading data from tcpdump file instead of real traffic capture. In real-time traffic capture, the duration may be longer.

BitTorrent Traffic

The first graph of Figure 18 (from left) shows a total number of 10 flows (out of 1,925 flows) that were falsely classified as game traffic at their initial classifications. However, 7 flows (70%) of these flows had flapping classification results before being finally detected as Non-Game traffic. Hence the final false positive rate for BitTorrent traffic is 0.15% (i.e. Precision rate of 99.85%).

The middle graph of Figure 18 shows the amount of classification flapping per flow. 70% of these flows had only one classification flapping (from Game to Non-Game) during their lifetime.

The last graph (from left) shows the duration of classification flapping per flow, where the classification flapped more than once. Again, these results are with the speed of reading data from tcpdump file instead of real traffic capture. In real-time traffic capture, the duration may be much longer.

SSH Traffic

The first graph of Figure 19 (from left) shows the total number of 28 flows (out of 80 flows) that were falsely classified as game traffic at their initial classifications. However, 8 flows (28.6%) of them had flapping classification results, before being finally detected as Non-Game traffic. Hence the final false positive rate for SSH traffic is 25% (i.e. Precision rate of 75%). The reason for the low accuracy rate for SSH traffic is that we only have a few sample instances of SSH traffic in the training dataset. To improve the performance of the FC toward identifying SSH traffic, we need to collect more SSH traffic sample flows for training.

The middle graph of Figure 19 shows the number of classification flapping per flow. 90% of those flows had up to 33 classifications flapping during their lifetime.

The last graph (from left) shows the duration of classification flapping per flow. It only shows results of flow with flapping times of greater or equal to once. Again, these results are with the speed of reading data from tcpdump file instead of real traffic capture. In real-time traffic capture, the duration may be much longer.

DNS and SMTP Traffic

With DNS traffic, there were only 2 false positive flows out of total 1196 flows (0.17%), hence a Precision of 99.83%. For SMTP traffic, there was no false positive flow out of the total 133 flows tested.

In summary, the accuracy of traffic classification met the requirement of the ANGEL architecture (except for SSH traffic that requires improvement in training the classifier model). However, the flapping rate per flow for all traffic is very high.

To minimise the flapping state and make the classification more stable, we propose a method of confirming the classification result before sending an update to the CM. The FC only updates the flow classification result and signals the CM if it sees two new, identical, consecutive classifications. In effect, the classification result would be held back by at least one classifying window for confirmation. The algorithm is described in greater detail in [1] and [4]. The improvement in term of classification stability by deploying the 'confirmed classification' scheme is evaluated in section IV-C.

3) CPU and Memory Usage: As the FM reads packet information from tcpdump files, the equivalent packet rates as if it sniffs packets in real-time are estimated in Figure 20.

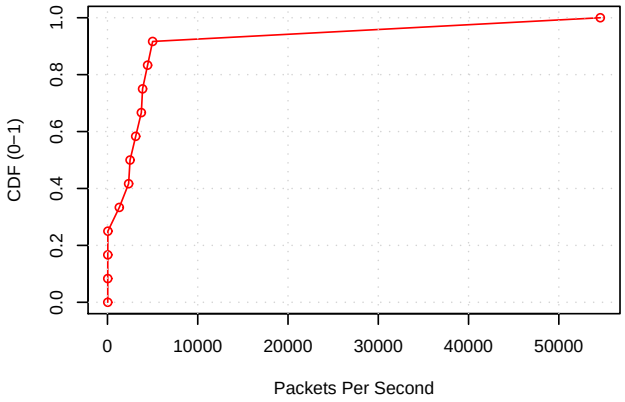

Fig. 20. Reading Tcpdump - PPS

However, since the FM combines roughly 97 packets' information in one single UDP packet to the FC, the incoming packet rate at the FC is much lower (see Figure 21).

With these packet rates, the FC CPU usage for all tests

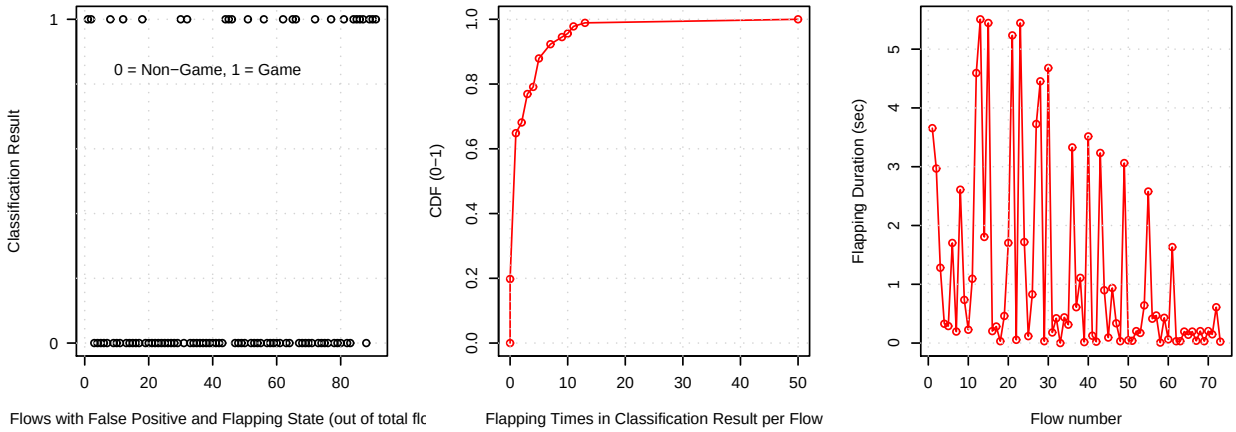

Fig. 17. Kazaa Traffic Classification

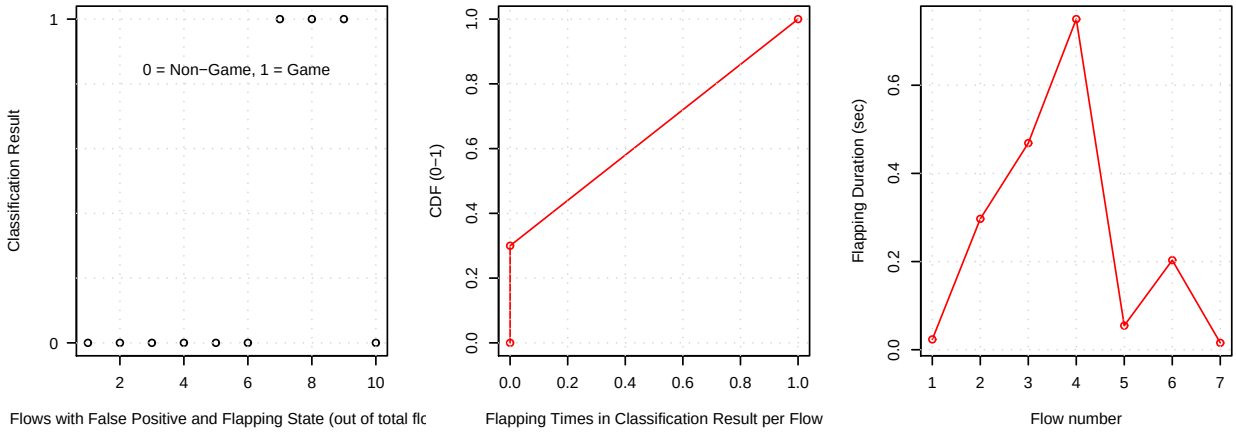

Fig. 18. BitTorrent Traffic Classification

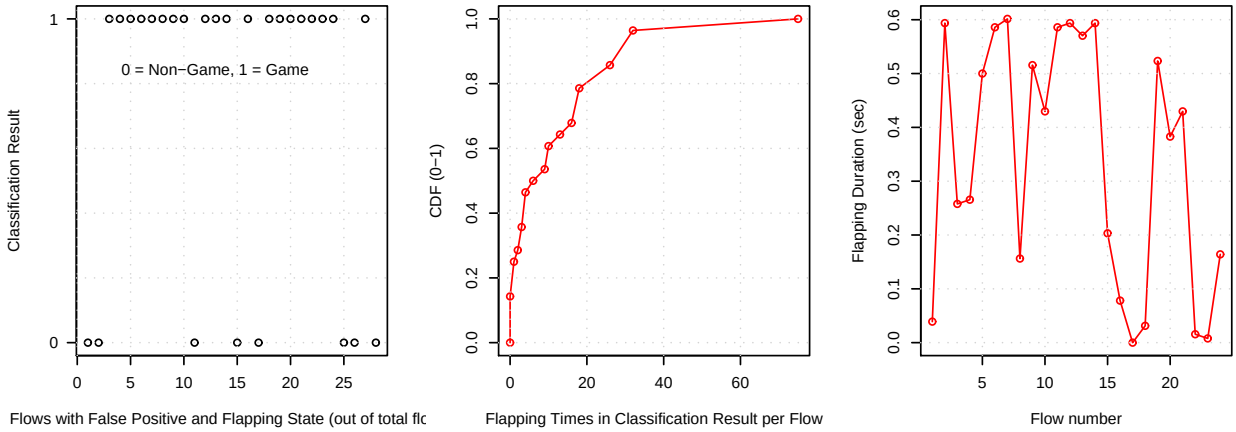

Fig. 19. SSH Traffic Classification

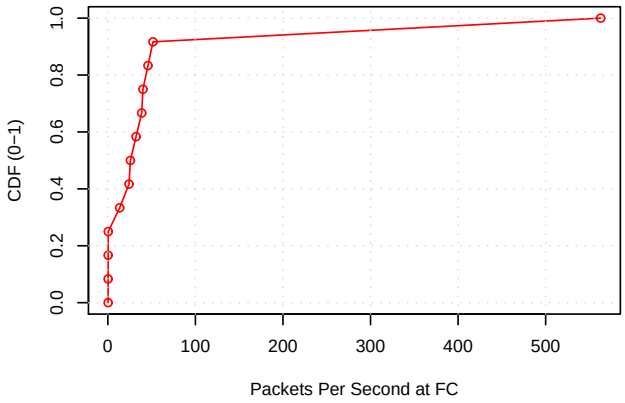

Fig. 21. Reading Tcpdump - PPS at FC

stayed at a negligible value 0%. Figure 22 shows the memory usage for all tests. The total memory usage was less than 5MB for all tests.

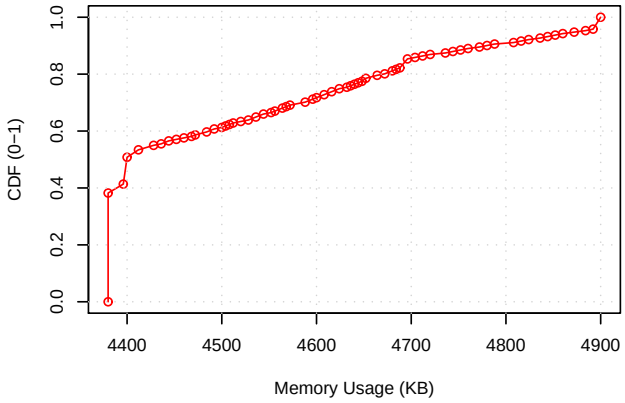

Fig. 22. Memory Usage

C. Improve the stability of the FC by state-confirmation method

This section shows results for the employment of our 'confirmed classification' scheme. We compare the performance of the FC in terms of flapping rate and accuracy between the use /no-use of the scheme for ET, Kazaa, Web and SSH traffic.

Figure 23 shows the results for ET traffic. The first graph (from left) shows that the 'confirmed classification' scheme helped to reduce the total flapping flows by 87% (from 38% to 5%). The last graph (from left) shows that the number of flaps per flow also have been reduced from one to six times per flow to only one flow that flaps twice (Game - Non-Game - Game) out of 39 game flows. The trade-off is the timeliness of the flow detection, which has increased from sub-second flow detection to up to 1.7 seconds.

Figure 24 shows the results for Kazaa traffic. The first graph (from left) shows that the 'confirmed classification' scheme helped to reduce the total flapping flows by ∼87% (from 1.15% to 0.15%). The middle graph shows that the number of flaps per flow during its lifetime have been reduced from maximum of 50 times to maximum of 6 times. As a result, the maximum flapping rate of more than 120 flaps per flow per second has been reduced to ∼ once per flow per second (the last graph from left).

Figure 25 shows the results for Web traffic. The first graph (from left) shows that the 'confirmed classification' scheme helped to reduce the total flapping flows by ∼ 81% (from 1.91% to 0.36%). The middle graph shows that the number of flaps per flow during its lifetime have been reduced from maximum of more than 300 times to maximum of 83 times, 99th percentile of 51 times to 4 times. The maximum flapping rate of more than 380 flapping per flow per second has been reduced to ∼ 21 times per flow per second (the last graph from left).

Figure 26 shows the results for SSH traffic. The first graph (from left) shows that the 'confirmed classification' scheme helped to reduce the total flapping flows by 46.4% (from 35% to 18.75%). The middle graph shows that the number of flaps per flow during its lifetime also have been reduced from maximum of 75 times to maximum of 4 times. Similarly, the flapping rate per flow per second has been reduced significantly (the last graph from left).

D. Performance of the FC in Live Traffic Capture

1) Game Traffic: We use two different game applications for the live traffic capture test: ET and HL2DM.

For ET traffic, test 6 traces of SONG (ET) traffic, with total of 39 game flows.

Figure 27 shows the final classification result for each game flow in the trace. It shows that the classifier correctly identified all 39 game flows (Recall of 100%).

Figure 28 shows how long it takes the classifier to identify the game flow from the time it received the first packet of the traffic flow.

For 37 out of 39 flows (95% flows), game traffic is classified in under 0.75 second. For only two outliers, it takes up to 2.8 seconds. Compared to the results presented in Figure 11 (that shows the time taken to detect ET flows when the FM reads from tcpdump files), these outliers are suspected due to the actual packet inter-arrival time of the particular flows.

Upon examination of the pre-recorded tcpdump files, we found that these two exceptional flows are captured at

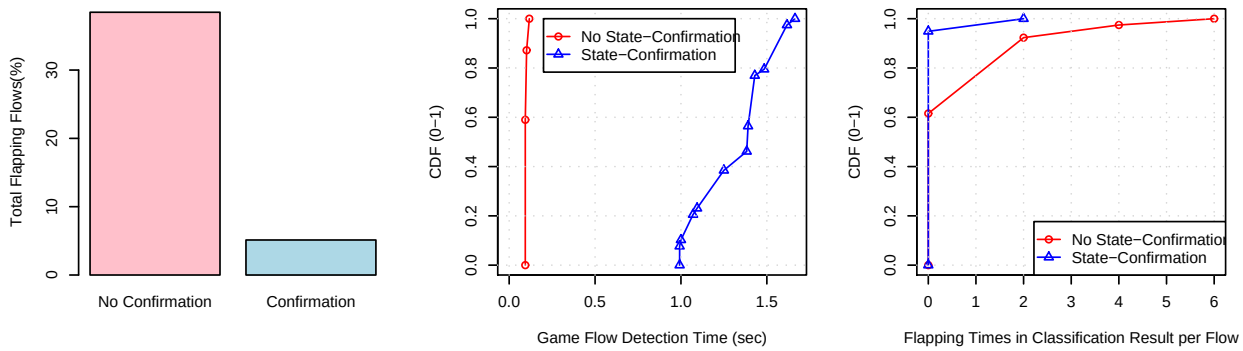

Fig. 23. FC stability with and without classification-state-confirmation approach - ET traffic

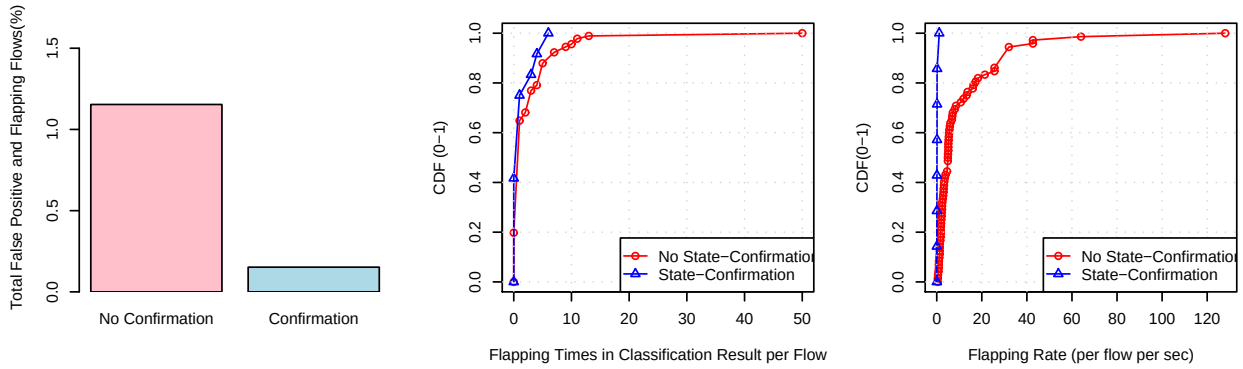

Fig. 24. FC stability with and without classification-state-confirmation approach - Kazaa traffic

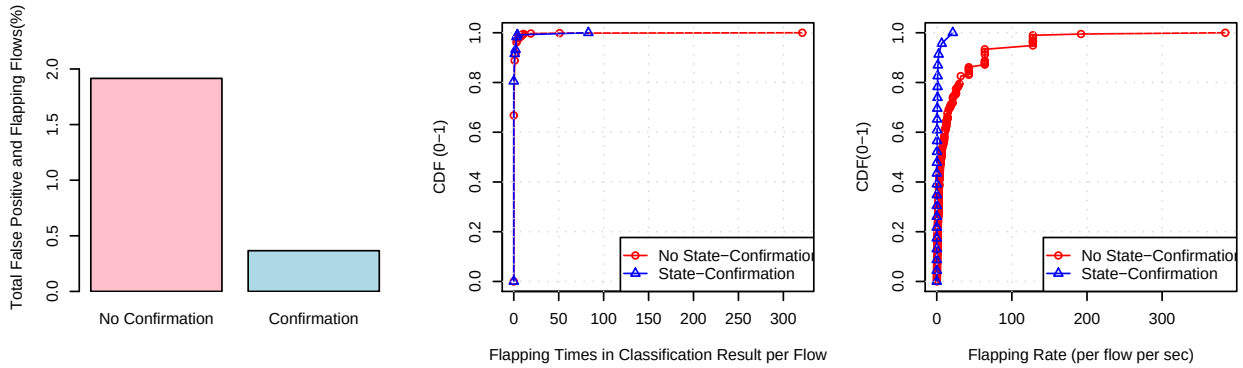

Fig. 25. FC stability with and without classification-state-confirmation approach - Web traffic

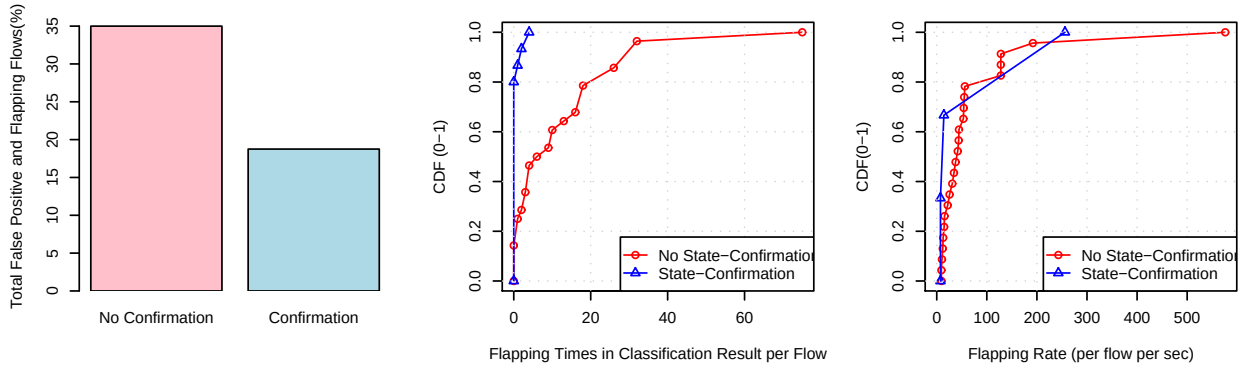

Fig. 26. FC stability with and without classification-state-confirmation approach - SSH traffic

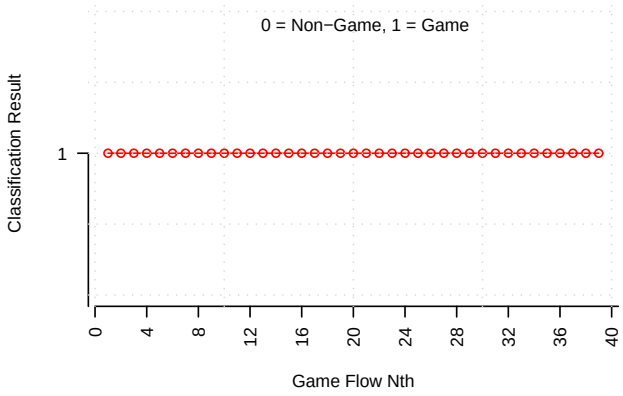

Fig. 27. LiveCapture ET Accuracy

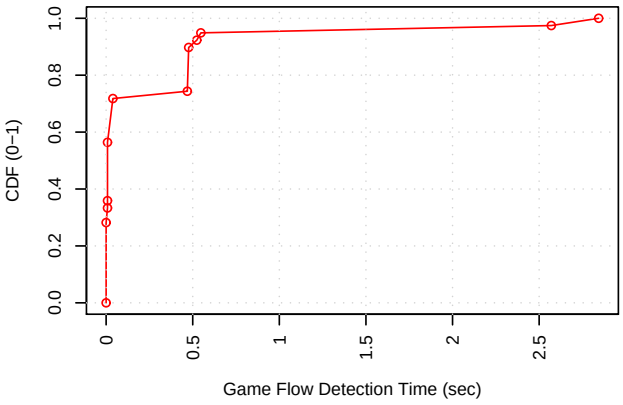

Fig. 28. LiveCapture ET Timeliness

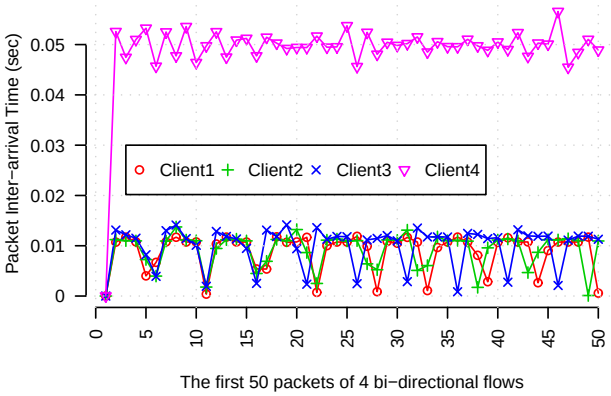

Fig. 29. Live Capture: ET packet inter-arrival time

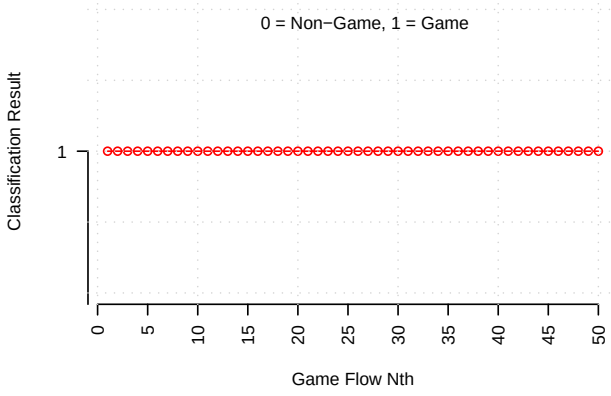

Fig. 30. LiveCapture HL Accuracy

the same client machine in two different traffic capture instances. These flows have longer packet inter-arrival times for the first few packets compared to other flows captured at other clients. Figure 29 shows the inter-arrival times of the first 50 packets of an outlier flow versus three other flows captured at different clients in the same traffic trace.(We need information of 50 packets (2 windows) for a confirmed classification result when using the state-confirmation method). As can be seen in the figure, Client4 (the outlier) flow has a much higher packet inter-arrival time compared with other flows. The mean packet inter-arrival time is 50msec, i.e. it takes approximately 2.5 seconds to collect 50 packets. This agrees with the 2.8 seconds flow detection time found above.

All 39 flows (100%) maintained their classification results stably during their flow lifetimes .

For HL2DM traffic, we test 8 traces of SONG traffic, with total of 50 game flows.

Figure 30 shows the final classification result for each game flow in the trace. It shows that the classifier correctly identified all 50 game flows (Recall of 100%).

Figure 31 shows how long it takes the classifier to identify the game flow from the time it received the first packet information from the FM.

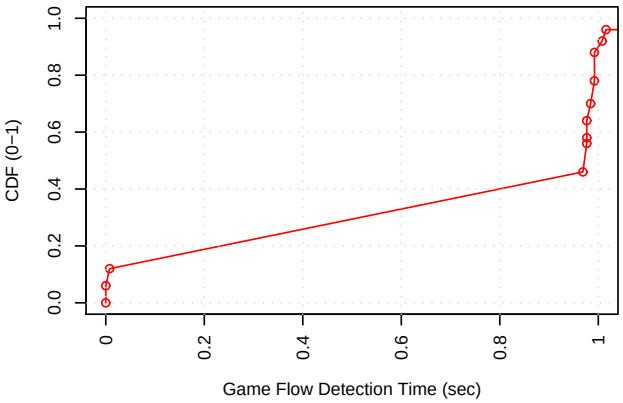

Fig. 31. LiveCapture HL Timeliness

For 48 out of 50 flows (96% flows), game traffic is classified in under 1 second. There were two outliers that take up to 10.8 and 44.2 seconds for flow detection. Reasons for these two outliers are similar to ET traffic

discussed above. They are due to the actual packet inter-arrival time of the particular flows.

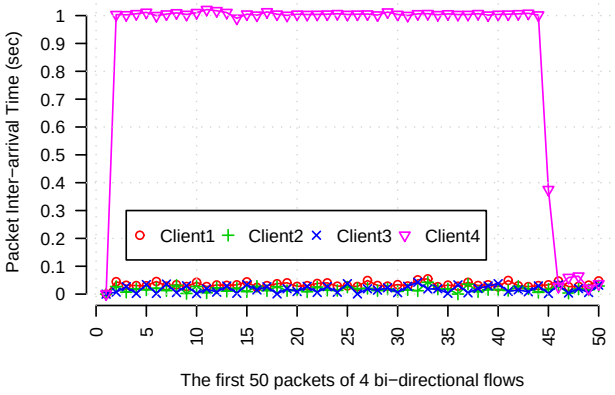

Fig. 32. Live Capture: HL2DM packet inter-arrival time

Upon examination of the pre-recorded tcpdump files, we find that these flows have longer packet inter-arrival time for the first few packets compared to other flows captured at other clients. Figure 32 shows the inter-arrival times of the first 50 packets of the outlier flow with 44.2 seconds flow detection versus three other flows captured at different clients in the same traffic trace. As can be seen in the figure, Client4 (the outlier) flow has much higher packet inter-arrival time compared to other flows. The mean packet inter-arrival time is at 876msec, i.e. it takes approximately 43.8 seconds to collect 50 packets. This agrees with the slow flow detection seen with the flow. Similar results with a lower mean packet inter-arrival time we seen for the other outlier.

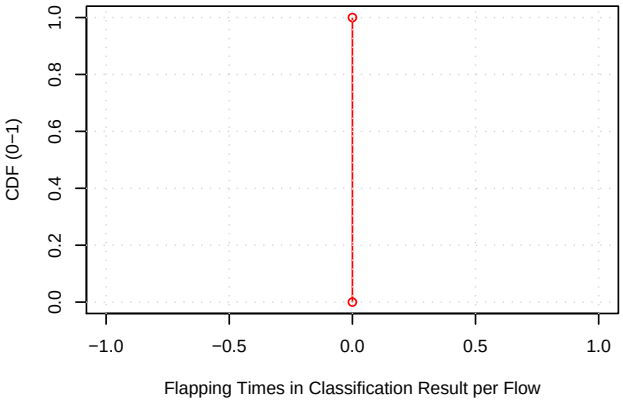

Fig. 33. LiveCapture HL Flapping States

50 out of 50 flows (100%) maintained their classification results stably during their flow lifetimes (see Figure 33).

2) Other Traffic: **Kazaa Traffic** Figure 34 shows the results for Kazaa traffic. The first graph (from left) shows a total number of 12 flows (out of 7,887 flows) that were falsely classified as game traffic at their initial classification. However, 5 flows of these had flapping classification results, and finally were detected as Non-game traffic. Hence the final false positive rate for Kazza traffic is 7/7887 = 0.09% (i.e Precision rate of 99.91%). Among flapping flows there were maximum of 6 flapping times per flow during its lifetime (middle graph). The last graph (from left) shows the duration where the flappings occurs in real-time for 7 flows that flap more than once. The mean duration of flapping times is 222 seconds.

SSH Traffic

Figure 35 shows the results for SSH traffic. The first graph (from left) shows the total number of 15 flows (out of 31 flows) that were falsely classified as game traffic at their initial classification. However, 2 of these had flapping classification results before being finally detected as Non-Game traffic. Hence the final false positive rate for web traffic is 13/31 = 41.9% (i.e. Precision rate of 58.1%). Among flapping flows there were maximum of 2 flapping times per flow during its lifetime (middle graph). The last graph (from left) shows the duration where the flapping occurs in real-time for 4 flows that flap more than once.

Since the trace file is 6-hour long, we do not run live capture test for other traffic. However, the accuracy for these traffic is expected to be the same as off-line traffic capture (when the FM reads from tcpdump files and preserves packet original information).

E. Scalability of FC in Live Traffic Capture

Using the Tcpreplay tool, we create a synthetic ET trace file of 350 unique flows (∼35000PPS) from the original trace of 7 ET flows with a PPS rate of approximately 500PPS by varying the flow's port numbers. We replicate the synthetic 350-flows tracefile once to 50 times. This gives us an approximate PPS arriving at the FM of 2500PPS to 25000PPS.

In terms of accuracy, all the synthetic flows are correctly identified as game traffic (100% Recall) with no flapping classifications for all flows. This suggests that for up to 25000 PPS (no packet loss at the FM (FM Performance Test Report)), the classification result is preserved.

With these packet rates, the maximum FC CPU usage for all tests stayed at a negligible value of less than 0.2% (Figure 36). Figure 37 shows the max memory usage for all tests. Total memory usages are of less than 5MB for all tests.

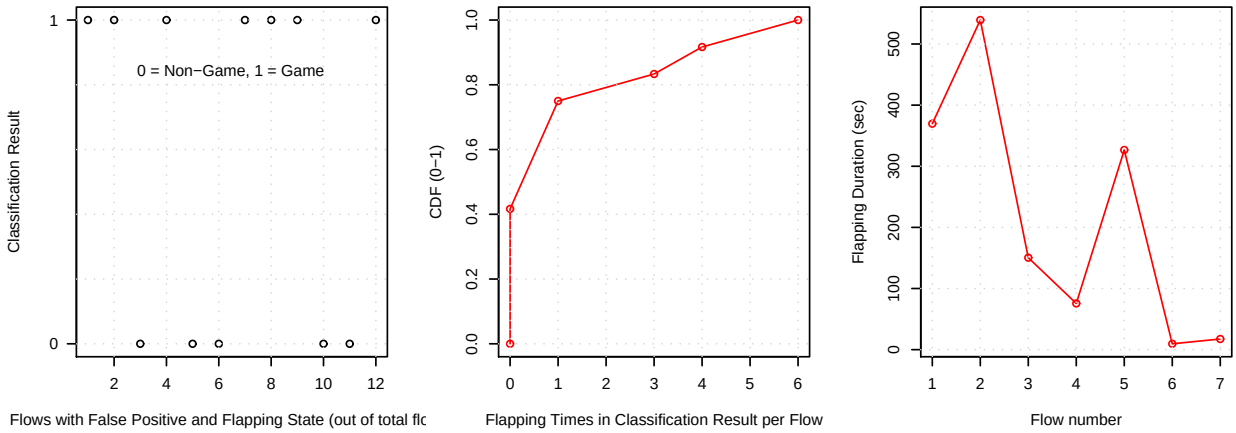

Fig. 34. Live Capture - Kazaa traffic

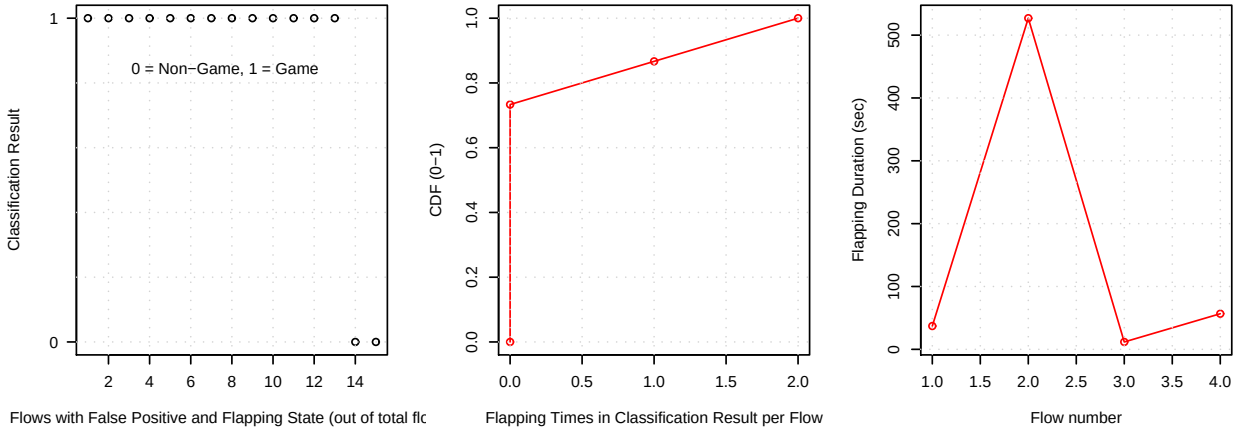

Fig. 35. Live Capture - SSH traffic

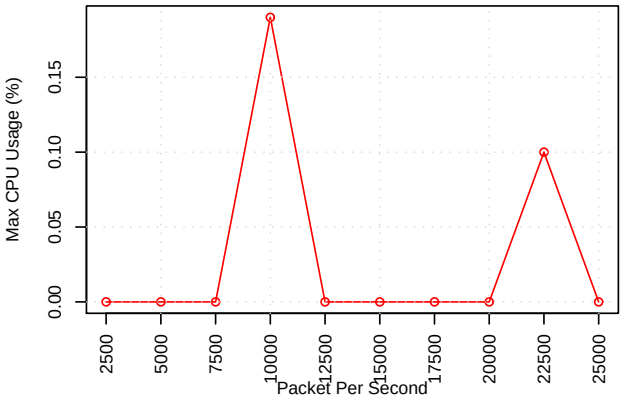

Fig. 36. CPU Usage per Packet Rate

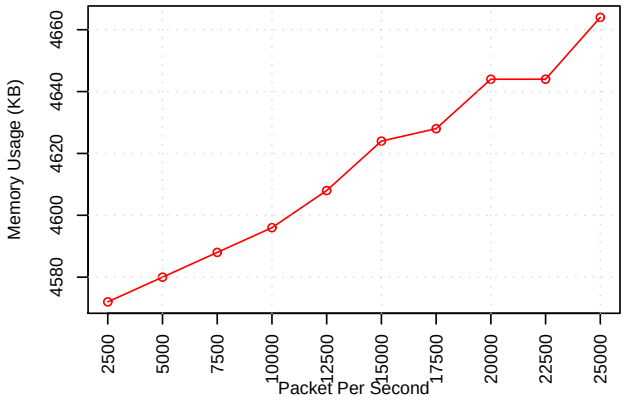

Fig. 37. Memory Usage per Packet Rate

With a typical ET flow of 70PPS (averaged over the 39 ET flows in test dataset), we extracted one single ET flow from the 7-players trace, and use tcpplay and Mergecap [15] tool to replicate it and create a trace containing 50 unique, concurrent flows. We duplicate the 50-flow trace from twice to 10 times, to simulate traffic rate of 100, 150, 200, ..., 500 concurrent flows. The equivalent PPS are approximately 7000, 10050, 14000,17500,..., 35000 PPS.

In terms of accuracy, all the synthetic flows are

correctly identified as game traffic (100% Recall) with no flapping classifications for all flows. This suggests that up to 500 concurrent flows, the classification accuracy is reserved.

With these flow rates, the maximum FC CPU usage for all tests stayed at a negligible value of less than 0.2% (Figure 38). Figure 39 shows the maximum memory usage for all tests. Total memory usages are less than 5MB for all tests.

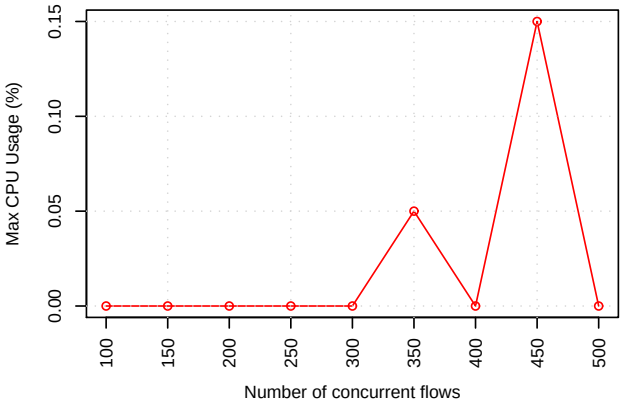

Fig. 38. CPU Usage per Flow Rate

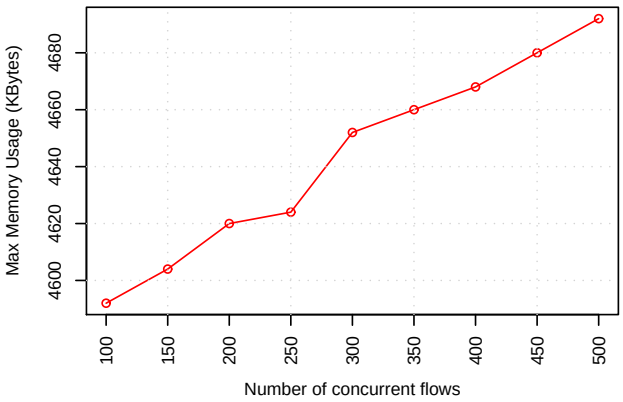

Fig. 39. Memory Usage per Flow Rate

V. Conclusion

The performance of the FC in terms of accuracy, timeliness, stability, scalability and CPU and memory usages when classifying Game and Non-Game traffic has been tested and reported.

The test of ANGEL Naive Bayes Classification model implementation shows that while being comparable to the WEKA model in terms of CPU usage, it is better in terms of model build time (40% faster than the WEKA model), classification speed (faster by 40%) and memory usage. On the other hand, the test results show a slightly worse accuracy rate (of up to 2% for the specific model and test dataset).

The test of the ANGEL FC with offline traffic analysis shows that FC can correctly classified all game flows with Recall rate of 100% for the test dataset used. The timeliness of the FC is achieved by using a small classification window of 25 packets. The FC can detect game traffic within 50 msec. However, as a trade-off, there were the cases when the flow's classification results flap during their lifetime. To overcome the problem and maintain the stability of the FC, we design the FC with 'confirmed classification' method. This has proven to be very effective by reducing the flapping rate by up to 87%. The false positive rate of Web, P2P, Email, DNS traffic are very low (of less than 0.4%), except SSH traffic (of 25%). This suggests the need of training the classification model with more samples of SSH traffic. With the maximum packet rate while reading from tcpdump files of 50,000PPS, the FC maintains its CPU usage of 0% and memory usage of below 5MB for all tests.

The test of the ANGEL FC with live traffic capture shows that with the two types of game applications tested (Enemy Territory and Half-Life2) the FC correctly identified 100% of the game flows mostly within a second with no flapping result. These results are recorded with the FC implementation with 'confirmed classification' scheme.

The scalability test shows that with packet rates of up to 25,000PPS (FM performance test shows that higher packet rates result in packet loss at the FM) and maximum number of 500 concurrent flows per second, the FC maintains its classification accuracy, keeps the memory usage to under 5MB and CPU usage to less than 0.15%. The test trace files used are limited to very short duration of $\sim$1 minute. Other trial runs on longer combination trace files show that the memory usage increases with respect to running time. However the increase is quite small and with slow changes.

Future work include the optimisation of the FC model in terms of the choice of sub-flows to train, greater instances of training data, variance of application types, classification window size, and the combination of the classification window size and the minimum number of delayed windows for the 'confirmed classification' scheme.

Future test might include the robustness test of the FC when classifying traffic with the presence of network delay, jitter and packet loss.

ACKNOWLEDGMENT

This work was supported from 2005 to early 2007 by the Smart Internet Technology Cooperative Research Centre, http://www.smartinternet.com.au.

I would also like to thank Jason But for his valuable suggestions on the approach to stabilise the FC performance, Lawrence Stewart for his help with modifying FM to read tcpdump file and debugging the database, Nigel Williams for his support with debugging the FC and building classification models. I am also grateful for Grenville Armitage's feedbacks to improve the report.

REFERENCES

[1] J. But *et al.*, "ANGEL Architecture Document," CAIA, Tech. Rep. 070228A, February 2006, http://caia.swin.edu.au/reports/050204A/caiaonly/CAIA-TR-070228A.pdf.

[2] J. But, N. Williams, S. Zander, L. Stewart, and G. Armitage, "ANGEL - Automated Network Games Enhancement Layer," in *Proceedings of Netgames 2006*, Singapore, October 2006.

[3] G. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 1995.

[4] J. But, "ANGEL Flow Classifier - Software Architecture Design Document," CAIA, Tech. Rep. 070228D, February 2006, http://caia.swin.edu.au/reports/050204A/caiaonly/CAIA-TR-070228D.pdf.

[5] T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks," in *Proceedings of the IEEE 31st Conference on Local Computer Networks*, Florida, USA.

[6] ——, "Synthetic Sub-flow Pairs for Timely and Stable IP Traffic Identification," in *Proceedings of the Australian Telecommunication Networks and Application Conference*, Melbourne, Australia.

[7] W. E. Territory, http://games.activision.com/games/wolfenstein, as of December 2005.

[8] (as of 27th April 2006) Et server. [Online]. Available: http://gs.act.grangenet.net

[9] (as of 26th March 2006) University of twente - traffic measurement data repository. [Online]. Available: http://m2c-a.cs.utwente.nl/repository

[10] (as of February 2006) Weka 3.4.4. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka

[11] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2000.

[12] (as of October 2005) Netmate. [Online]. Available: http://sourceforge.net/projects/netmate-meter

[13] (as of 21 December 2006) Simulating online networked games. [Online]. Available: http://caia.swin.edu.au/sitcrc/

[14] (as of September 15th 2006) Tcpreplay. [Online]. Available: http://sourceforge.net/projects/tcpreplay/

[15] (as of September 18th 2006) Mergecap. [Online]. Available: http://www.ethereal.com/docs/man-pages/mergecap.1.html