

# Synthetic Sub-flow Pairs for Timely and Stable IP Traffic Identification

Thuy T.T. Nguyen, Grenville Armitage  
Centre for Advanced Internet Architectures  
Swinburne University of Technology, Melbourne, Australia  
{tnguyen, garmitage}@swin.edu.au

**Abstract**—Literature on the use of Machine Learning (ML) algorithms for classifying IP traffic has relied on bi-directional full-flow statistics while assuming that flows have explicit directionality implied by the first packet captured or the Client-to-Server direction. In contrast, many real-world classifiers may miss an arbitrary number of packets from the start of a flow, and be unsure in which direction the flow started. This would lead to degradation in classification performance for application with asymmetric traffic characteristics. We propose a novel approach to train the ML classifier using statistical features calculated over multiple short sub-flows extracted from full-flow generated by the target application and their mirror-imaged replicas as if the flow is in the reverse direction. We demonstrate our optimisation when applied to the Naive Bayes and Decision Tree algorithms. Our approach results in excellent performance even when classification is initiated mid-way through a flow, without prior knowledge of the flow’s direction and using windows as small as 25 packets long.

## I. INTRODUCTION

Real-time traffic classification has potential to solve difficult network management problems for ISPs and their equipment vendors. Traffic classification may be a core part of automated intrusion detection systems, denial of service attacks detection, trigger automated re-allocation of network resources for priority customers, or identify the use of network resources that contravenes the operator’s terms of service.

Commonly deployed IP traffic classification techniques involve direct inspection of each packet’s contents on the network and/or TCP/UDP port numbers. Yet the value of such techniques is diminishing. Regular updates are required to track minor changes in applications’ packet payload formats, customers obfuscate packet contents through encryption, and government privacy regulations may constrain the ability of third parties to lawfully inspect packet payloads.

The research community has responded with traffic classification based on statistical patterns in externally observable attributes of the traffic. Particular efforts have occurred in the application of Machine Learning (ML) techniques to IP traffic classification [1] [2] [3] [4]. Attributes of flows (e.g. max/min packet lengths, flow durations or inter-packet arrival times) are known as ‘features’. Classification involves two stages - training the ML algorithm to associate sets of features with known traffic classes (creating rules), and applying the learnt rules to classify unknown traffic. Each ML algorithm has a different approach to sorting and prioritising sets of features,

which leads to different dynamic behaviours during training and classification.

Our research contributes to the practical application of ML algorithms within the constraints of IP traffic classifiers deployed in operational networks. Most published research has evaluated the effectiveness of different ML algorithms when applied to entire datasets of IP traffic - trained and classified over full flows consisting of thousands of packets and hundreds or thousands of flows. The efficacy of ML classifiers has not been explored when they have access to only a subset of a flow’s packets or they do not see the start of every flow (even the most recent work of [5] assumes the initial packets of every flow are captured and available for classification). In addition, although not always clearly stated, directionality has been an implicit attribute of the features on which ML classifiers were trained and tested. Application flows are assumed to be bi-directional, and the application’s statistical features are calculated separately in the forward and reverse directions [1] [2] [3] [5] [6]. Most work assumes that the forward direction is indicated by the first packet of the flow (on the basis that it is commonly the initial packet from a client to a server) [6] [3]. Subsequent evaluations assume a trained ML classifier sees the first packet of every flow in order to calculate features with the correct sense of forward and reverse direction.

In real IP networks traffic classifiers must reach decisions well before a flow has finished, they may not see the actual start of a flow, and the application’s statistical behaviour may change over the lifetime of each flow. For privacy reasons a classifier may be denied access to packet payloads or even certain header fields, and have no knowledge of where clients or servers are actually located on the network. Thus a real-world classifier cannot be sure whether the first packet it sees (of any new bi-directional flow of packets) is heading in the ‘forward’ or ‘reverse’ direction. This can lead to degraded classification performance.

The preceding considerations gives rise to our proposal that practical real-time traffic classifiers must accurately classify traffic in the face of a number of constraints:

- The classifier should use statistical methods (such as ML algorithms) as TCP/UDP port numbers may be misleading, and packet payloads may be opaque against direct interpretation
- ML classification should be done over a small sliding

window of the last  $N$  packets (to keep memory requirements down and perform classification in a timely manner)

- The classifier must recognise flows already in progress (the flow's beginning may be missed)
- Application's can change their network traffic patterns over time
- The classifier doesn't have to know about the direction the original flow traverses, it can assume the forward direction is the direction of the 1st packet of most recent  $N$  packets it captured, regardless if it is from client to server or server to client.

We have previously demonstrated a novel approach to ML classification that meets the first four requirements [7]. We proposed training the classifier on a combination of short sub-flows extracted from full-flow examples of the target application's traffic. Our approach resulted in excellent performance while allowing the sliding window classifier to properly identify an application regardless of where within a flow the classifier begins capturing packets.

This paper extends our work in [7] to address the last requirement listed above. We propose and demonstrate a novel modification: The ML classifier is trained using statistical features calculated over multiple short sub-flows (as in [7]) and their mirror-imaged replicas (multiple 'synthetic sub-flow pairs'). The sub-flows are picked from regions of the application's full flows that have noticeably different statistical characteristics, and coupling with their mirrored replicas as if they are travelling in the reverse direction.

We demonstrate the benefit of our proposal when applied to two different supervised learning ML algorithms, C4.5 Decision Tree and Naive Bayes. We utilise a hypothetical classification scenario where real-time flows belonging to the online multiplayer game *Wolfenstein Enemy Territory* (ET) [8] must be dynamically detected while mixed in amongst thousands of unrelated, interfering traffic flows. We characterise the classification performance of each ML algorithm as a function of the location of the sliding window relative to the actual beginning of an application flow and the apparent 'direction' of a flow.

Our paper is organised as following. Section II briefly summaries keys components of our proposal. Section III describes our experimental demonstration of our proposal, with our results analysed in section IV. Section V describes some future research directions and concludes our paper.

## II. OUR PROPOSAL

We present an improved technique for training and using ML classifiers such that IP flows can be classified in finite periods of time and without regard to inferred or actual directionality of flow or the number of packets missed from the beginning of a flow. We propose that realistic ML-based traffic classification tools should:

- Operate the ML classifier using a sliding-window over each flow - we presume the classifier can see (or must

use) no more than the most recent  $N$  consecutive packets of a flow (an  $N$ -packet sub-flow) at any given time.

- Train the ML classifier using sets of features calculated from multiple sub-flows. The sets of training sub-flows are created in two steps: (a) Take short sequences of  $N$  consecutive packets from different points within examples of the target application's flows (a sub-flow), and then (b) create mirror-imaged replicas of each packet sequence with the directions reversed (a mirrored sub-flow).

Training the classifier on multiple sub-flows of size  $N$  packets maximises the classifier's ability to recognise an application flow even when exposed to only a small window of traffic from the flow. Each sub-flow is taken from places in the original flow having noticeably different statistical properties (for example, the start and middle of the flow) [7].

Also training on mirrored replicas of each sub-flow is the key contribution of this paper. Doing so enables the classifier to recognise an application's traffic in either direction. This is an important step - as packets flow through the classifier's sliding window the first packet can alternately represent traffic in the Client to Server (C-S) or Server to Client (S-C) direction. To ensure the classifier need not make that distinction, we train the classifier to recognise the application in either direction.

## III. ILLUSTRATING OUR EXPERIMENTAL APPROACH

### A. Machine Learning Algorithms and Terminology

In this paper we use the Naive Bayes [9] and C4.5 Decision Tree [10] implementations in WEKA tools [11]. These are well-understood supervised-learning algorithms with different internal training and classification mechanisms. Testing our proposal with both algorithms reveals benefits in either case, suggesting our proposal is applicable to more than just one particular type of ML algorithm. (Due to space limitations we refer readers to [12] for further details of these algorithms.)

Recall and Precision are two metrics often used to evaluate the performance of ML classification algorithms. If a classifier is trained to identify members of class  $X$ , **Recall** refers to the proportion of class  $X$ 's instances which are correctly classified as belonging to class  $X$  and **Precision** refers to the proportion of the instances which truly have class  $X$  among all those classified as class  $X$ . Both metrics range from 0 (poor) to 100% (optimal). While using both, it is important to note that high Precision only is meaningful when the classifier achieves good Recall.

### B. Flows and Features

**Full-Flows** are bidirectional streams of packets between a given pair of hosts, defined by the source and destination IP addresses, port numbers and protocol. The C-S direction determines the 'forward' direction. Flow timeout is used as specified in [6]. Each **sub-flow** is a fragment of  $N$  consecutive packets (bi-directional) taken from different points within the original application flow's lifetime. Its forward direction is defined the same as full-flow: C-S direction. The **Synthetic Sub-Flow pairs** consists of the sub-flows and their synthetic

pairs, whose statistical properties are the same as the sub-flows except they are swapped for backward and forward directions (as being calculated with the forward direction of the sub-flow is defined as the reverse direction: S-C).

For full-flow, sub-flow and synthetic sub-flow pairs models, we trained and classified the classifier using the following features, calculated separately in the forward and backward directions: Inter-packet arrival interval, Inter-packet length variation and IP packet length; all with minimum, maximum, mean and standard deviation values. These features are calculated based on the framework of Netmate tool [13].

For testing the classifier we assume that the classifier defines the forward direction as the first packet of a flow that it could capture, regardless whether it is from C-S or S-C.

### C. Construction of training and testing datasets

To show the effectiveness of our proposed approach we use completely different datasets for training and testing our classifiers. The game traffic consists of two separate month-long traces collected during May and September 2005 at a public ET server in Australia [14]. Our interfering (non-ET) traffic came from two 24-hour traces collected by the University of Twente, Germany, on February 6th and 7th 2004 [15](named T1 and T2 respectively). They include a large range of common applications (HTTP, HTTPS, DNS, NTP, SMTP, IMAP, POP3, Telnet, SSH, HalfLife, Kazaa, Bittorrent, Gnutella, eDonkey). 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). For each application’s default port(s) we sampled a maximum of 10000 flows per raw tracefile.

For each experiment we trained our classifiers using a mix of ET traffic from May dataset and interfering traffic from T2 (sub-flows are extracted from 8,688 full-flows for ET mixed with 82,957 full-flows in T2). Subsequent testing of each classifier scenario was performed using a mix of ET traffic (N packets extracted from 6,888 full-flows) from September and traffic from T1 (N packets extracted from a mixture of 73,672 full-flows). Further details can be found in [7].

### D. Statistical properties of ET traffic

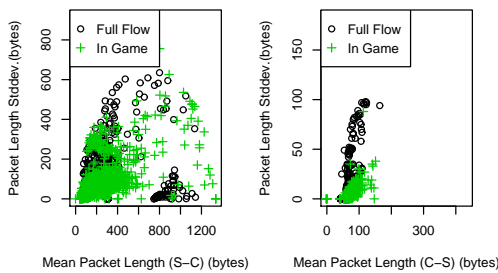


Fig. 1. Packet Length from S-C and C-S

Figure 1 compares Std.Dev vs. Mean packet length calculated with  $N = 25$  taken at the middle of the flows (“In game”), in comparison to the same feature values calculated over full-flows. It is clear that feature values calculated “In game” are

quite distinctive, and differ significantly between the S-C and C-S directions. Clearly there is a significant asymmetric in the feature values in the bi-directional communications.

Measured across all the ET flows, Figure 2 shows the percentage of sub-flows whose first packet is in the C-S direction as a function of how many packets ( $M$ ) are missed from the start of the full flow. This is 100% when  $M = 0$ , and fluctuates significantly for small, non-zero values of  $M$  (the value doesn’t reach 0% for  $M = 1$  because in some finite number of ET flows both the first and second packets seen on the wire are in the C-S direction). In the region  $2000 \leq M \leq 2009$  there appears to be roughly equal chance that the 2001st, 2002nd, ... 2009th packets traverse in the C-S or S-C directions.

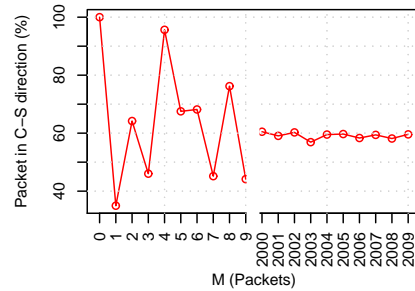


Fig. 2. Flows with the Mth packet captured in the C-S direction

Recall that the game flow’s statistical properties are asymmetric. Thus training the classifier models with features calculated in one direction leads to degraded Recall anytime the first packet captured traverses in the reverse direction. In the next section we see that such a classifier’s overall Recall rate is proportional to the number of sub-flows that actually start in the same direction as used to train the classifier.

## IV. RESULTS AND ANALYSIS

We show how Recall and Precision improve significantly when each ML classifier is trained using multiple synthetic sub-flows pairs instead.

### A. Classifying using a sliding window without training on mirrored sub-flows

First we recap the results in [7], comparing the effectiveness of classification when the classifier has been trained on full-flow features and multiple sub-flow features in a specific direction. We use a window of size  $N=25$  packets and sub-flows based on packets 1-25, 21-45, 41-65 and 2001-2025 [7]. (During ET game-play we see 20 pps from server to client and roughly 28 pps from client to server [16], so this window corresponds to 0.5 second of real time). The classifier has not been trained using mirrored sub-flows, and assumes that the first packet it sees in each sliding window is in the same (C-S) direction as the sub-flows on which it was trained.

Figure 3 shows Recall and Precision for Naive Bayes models as each sliding window moves across the test dataset.  $M$  is the number of packets ‘missed’ from the beginning of

each flow in the test dataset. The graphs cover two periods - early client contact with the game server ( $0 \leq M \leq 9$ ) and during active game-play ( $2000 \leq M \leq 2009$ ).

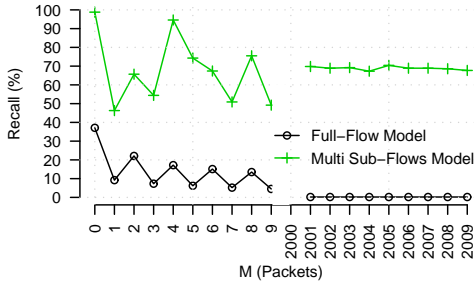


Fig. 3. NB Recall: N=25 trained on Full-Flow and Multiple Sub-Flows

Recall for both training models suffer as  $M$  increases above zero, yet training on multiple sub-flows is significantly better than training on full-flow features. The median Recall when trained on multiple sub-flows fluctuates significantly around 66% ( $0 \leq M \leq 9$ ) and sits relatively stable at 69% when  $2000 \leq M \leq 2009$ . More importantly, for small values of  $M$  we see dramatic shifts in Recall each time the sliding classification window moves by one packet. This is a direct consequence of the classifier assuming (sometimes incorrectly) that the first packet in the sliding window represents the C-S direction when in reality it does not (as shown in Figure 2).

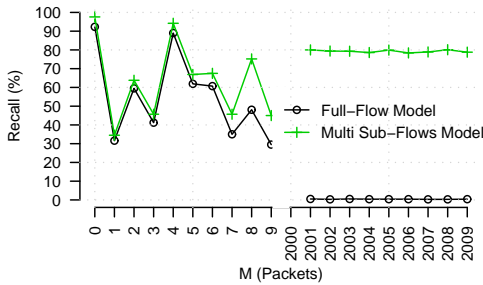


Fig. 4. Recall Rate for Full-Flow, Sub-Flows Decision Tree models

Figure 4 shows similar results with the Decision Tree models. Missing early packets in the flow ( $0 \leq M \leq 9$ ) results in a median Recall of 54% and 65% for full-flows and multiple sub-flows models respectively. Both models degrade noticeably when the first packet in the window is in the S-C rather than C-S direction. When classification begins in the middle of a flow ( $2000 \leq M \leq 2009$ ) the multiple sub-flows model's Recall is stable at 79% (still not particularly good).

### B. Training on multiple synthetic sub-flow pairs, classifying with a sliding window

Now we demonstrate the effectiveness of our new proposal to additionally train the classifier on synthetically mirrored pairs of sub-flows take from different time periods within the original full-flows. The classifier will then recognise new flows in either direction if they have statistical properties similar to any of the sub-flows on which the classifier was trained.

We create the synthetic model for the combination of sub-flows 1-25, 21-45, 41-65 and 2001-2025. Now in the training dataset, we have both instances from the multiple sub-flows with the forward direction defined as C-S direction, and their "synthetic sub-flow pair" instances with feature values calculated as the forward direction is from S-C direction.

Using the Naive Bayes ML algorithm Figure 5 shows Recall as a function of  $M$  for this new classifier ('Multi Syn. Sub-Flows Model') and a classifier trained on multiple sub-flows as in [7] ('Multi Sub-Flows Model', cf. Figure 3).

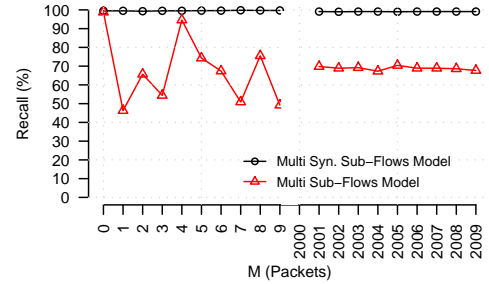


Fig. 5. Recall Rate for Naive Bayes models

Compared to the multiple sub-flows model (with a median Recall of 69%) the multiple synthetic sub-flows curve shows excellent Recall (median of 99%) that is almost unaffected by the alternating of apparent flow direction caused as the classifier misses  $M$  packets.

However, the gain in Recall is accompanied by a slight loss in Precision. Using Naive Bayes Figure 6 shows Precision as a function of  $M$  for the same models shown in Figure 5. The median Precision drops by 1.3% for  $0 \leq M \leq 9$  and 1.9% for  $2000 \leq M \leq 2009$ . Nevertheless our new approach still achieved 96.6%-97.7% Precision for different values of  $M$ .

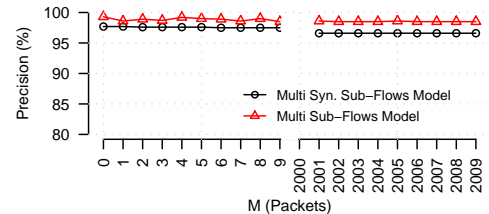


Fig. 6. Precision Rate for Naive Bayes models

Improved Recall is also seen when our new approach is applied to a Decision Tree classifier. Compared to the multiple sub-flows model (with a median Recall of 79%), Figure 7 shows the multiple synthetic sub-flows classifier having excellent and consistent Recall (median of 99%) that is almost unaffected by the alternating of apparent flow direction caused as the classifier misses  $M$  packets.

Unlike the Naive Bayes models, Figure 8 shows the Decision Tree classifier exhibiting a gain in Precision when utilising our new approach. The median Precision increases by 2.7% for different values of  $M$ , staying around 97.3%-98.2% for the Multiple Synthetic Sub-flows model.

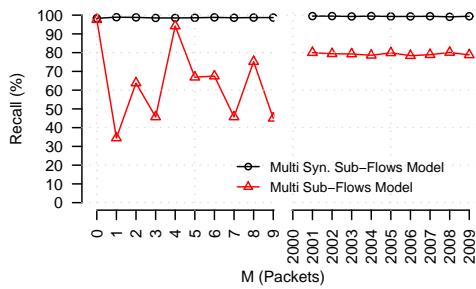


Fig. 7. Recall Rate for Decision Tree models

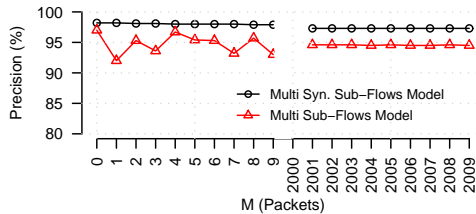


Fig. 8. Precision Rate for Decision Tree models

The change in Precision exhibited by the Naive Bayes and Decision Tree algorithms suggest sensitivities of our approach to the internal construction of each ML algorithm and the statistical properties of the 'interfering' traffic. In the training phase, while using the synthetic sub-flow statistics, we introduce the new range of possible values (of features in the reverse direction) to a single attribute to build the classifier. For Naive Bayes, depending on how different the asymmetric traffic statistics are in different flow direction, this would change the distribution's parameters of the attribute accordingly (the mean and std.dev of the attribute's normal distribution function). Similarly, the values range of a single attribute for the Decision Tree algorithm also might change significantly, introducing new range of test nodes and branches. While the test on each test nodes of the Decision Tree algorithm is based on a particular possible value of the attribute, the decision of Naive Bayes algorithm is based on the distribution of the attribute values that might match better with possible attribute values of the interference traffic. Further explanation of the reduction and increase in Precision for Naive Bayes and Decision Tree algorithms respectively is left to future work.

## V. CONCLUSION

In this paper we address the problem of a practical, ML-based traffic classifier that must use a small sliding window, start at an arbitrary point within a flow's lifetime, and recognise application flows regardless of implied flow direction. We extend previous work on training with sub-flows [7] to include the idea of training with multiple synthetic sub-flow pairs. Each sub-flow extracted from training data is paired with a mirror image sub-flow that appears to go in the opposite direction. We demonstrate significantly improved classification performance by constructing, training and testing

with Naive Bayes and C4.5 Decision Tree classifiers for the detection of Wolfenstein Enemy Territory online game traffic. With a sliding window of only 25 packets we saw excellent results with two quite different ML algorithms. We believe our approach will also show benefits when applied to other ML algorithms.

In the future we plan to characterise optimal values of N for different target applications and memory consumption limits in the classifier itself, explore the impact of packet loss on the achievable Recall and Precision, and test our proposal in the presence of a larger and more diverse collection of interfering traffic. Overall we believe this small proposal significantly improves the utility of ML algorithms inside practical and deployable IP traffic classifiers.

## ACKNOWLEDGMENT

The authors would like to thank the Cisco University Research Program Fund at Community Foundation Silicon Valley for their grant support, S. Zander and N. Williams for their help with the dataset used.

## REFERENCES

- [1] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *PAM2004*, 2004.
- [2] A. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS 2005*, 2005.
- [3] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Proceedings of the IEEE 30th Conference on Local Computer Networks*, 2005.
- [4] S. Zander, N. Williams, and G. Armitage, "Internet archeology: Estimating individual application trends in incomplete historic traffic traces," in *Proceedings of PAM2006*, 2006.
- [5] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, 2006.
- [6] N. Williams, S. Zander, and G. Armitage, "Evaluating machine learning methods for online game traffic identification," Centre for Advanced Internet Architectures, Tech. Rep. CAIA Technical Report 060410C, April 2006.
- [7] T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks," in *Proc. IEEE 31st Conference on Local Computer Networks*, Tampa, Florida, USA, November 2006 (In Press, available at [http://caia.swin.edu.au/pubs/lcn2006-nguyen\\_armitage\\_marked.pdf](http://caia.swin.edu.au/pubs/lcn2006-nguyen_armitage_marked.pdf)).
- [8] (as of 19 December 2005) Wolfenstein. [Online]. Available: <http://games.activision.com/games/wolfenstein>
- [9] G. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 1995.
- [10] R. Kohavi, J. Quinlan, W. Klosgen, and J. Zytkow, "Decision tree discovery," in *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, pp. 267–276.
- [11] (as of February 2006) Weka 3.4.4. [Online]. Available: <http://www.cs.waikato.ac.nz/ml/weka>
- [12] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2000.
- [13] (as of October 2005) Weka 3.4.4. [Online]. Available: <http://sourceforge.net/projects/netmate-meter>
- [14] (as of 27th April 2006) Et server. [Online]. Available: <http://gs.act.grangenet.net>
- [15] (as of 26th March 2006) University of twente - traffic measurement data repository. [Online]. Available: <http://m2c-a.cs.utwente.nl/repository>
- [16] J. Bussiere and S. Zander, "Enemy territory traffic analysis," Centre for Advanced Internet Architectures, Tech. Rep. CAIA Technical Report 060203A, February 2006.