Performance Analysis of the ANGEL System for Automated Control of Game Traffic Prioritisation

Jason But, Thuy Nguyen, Lawrence Stewart, Nigel Williams, Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{jbut,tnguyen,lastewart,niwilliams,garmitage}@swin.edu.au

ABSTRACT

The Automated Network Games Enhancement Layer (ANGEL) [6] is a novel architecture for meeting Quality of Service (QoS) requirements of real-time network game traffic across consumer broadband links. ANGEL utilises detection of game traffic in the ISP network via the use of Machine Learning techniques and then uses this information to inform network routers - in particular the home access modem where bandwidth is limited - of these flows such that the traffic may be prioritised. In this paper we present the performance characteristics of the fully built ANGEL system. In particular we show that ANGEL is able to detect game traffic with better than 96% accuracy and effect prioritisation within a second of game flow detection. We also demonstrate the processing performance of key ANGEL components under typical hardware scenarios.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: [network architecture and design, network operations, distributed systems, internetworking]

General Terms

Performance, Management

Keywords

Traffic Classification; Quality-of-Service; Machine Learning; Online Gaming

1. INTRODUCTION

Highly interactive network games, such as the very popular first-person shooter (FPS) genre, place different demands on the network compared to traditional non-interactive applications (e.g. web, email). For these games the user's ability to play successfully greatly depends on the Quality of Service (QoS) present in the network. Recent studies found that with degrading network conditions (increasing delay, jitter, packet loss) users play less effectively (e.g. they achieve a lower score) and their perceived quality of the game play significantly decreases (see [9, 13, 16]).

With wider adoption of broadband access technologies such as Asymmetric Digital Subscriber Loop (ADSL), Cable Internet and 802.11x, it has become increasingly popular to share Internet connections between a number of computers, or run multiple network applications concurrently. This type of heterogeneous network traffic environment places increased demand on the network infrastructure, potentially creating a bottleneck at the access link. This impacts a game and file transfer differently - game playability will be significantly degraded while the file transfer will experience some reduction in throughput. A solution is to separate the traffic into two groups: interactive (real-time) and non-interactive (non real-time). Interactive traffic can then be prioritised to ensure QoS requirements are met.

We have previously proposed ANGEL [6], a novel architecture for moving QoS signaling from the application to the network. ANGEL places intelligence within the ISP network to identify game traffic flows that would benefit from QoS, transparently providing QoS to applications. We also proposed a novel traffic classification method based on machine learning (ML) [21]. Preliminary results show that our method accurately and effectively separates game from non-game traffic, and that this classification can be made upon receiving as little as 50 packets from the game application.

The paper is structured as follows. Section 2 summarises the ANGEL architecture. Section 3 presents the ANGEL system functionality while Section 4 discusses the performance characteristics of the ANGEL system.

2. ANGEL ARCHITECTURE

We have previously outlined the need and basic architecture for the ANGEL system [6]. We noted that networked games are particularly sensitive to variations in network quality [9, 13, 16, 18, 19]. We also presented a variety of possible techniques for improving the QoS provided to network games before deciding upon simple priority queuing. This form of QoS support is already enabled in consumer grade hardware [7, 22] - albeit in a statically configurable form - and can be demonstrated to substantially improve the performance of networked games in the presence of other network traffic despite only operating on one link rather than

NetGames'07, September 19-20, 2007, Melbourne, Australia

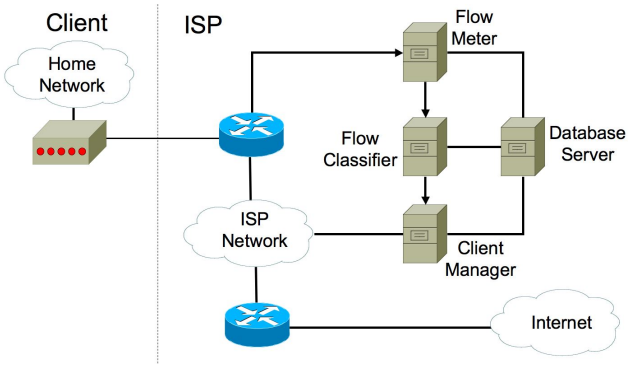

Figure 1: ANGEL System Block Diagram

across the entire network path [6], [11].

2.1 Basic System Structure

The basic ANGEL Architecture is shown in Figure 1. The basic premise is that the ISP side components of ANGEL receive a copy of all network traffic which is classified in order to detect network game flows. This information is then used in a feedback loop to individual CPE devices to request prioritisation for the selected flows.

The ISP side components consist of three modules: a Flow Meter, a Flow Classifier and a Client Manager. These modules have the following key features:

- **Flow Meter -** Multiple Flow Meters, the number and configuration dependent on the ISP network configuration. The Flow Meter is responsible for monitoring a copy of all network traffic at a particular location, filtering this traffic into individual flows, and forwarding packet (timestamp, packet size) and flow (IP/Port number tuples) information to the Flow Classifier. The Flow Classifier does not capture or inspect any packet payloads.
- **Flow Classifier -** Single Flow Classifier, The Flow Classifier is responsible for using the information forwarded by Flow Meters to classify the flows in real-time. Once a flow has been classified as being - or has been re-classified to - a network game flow, this information is forwarded to the Client Manager for communication to the respective CPE devices. The method employed to classify flows is flexible.
- **Client Manager -** Maintains a database of connected CPE devices and forwards priority flow information to the relevant CPE.
- **ANGEL Protocol -** Responsible for all communications between the ANGEL ISP side components and ANGEL enabled CPE devices. The protocol is designed to cover the full range of potential system failure modes and concurrent connection possibilities.

The primary workload placed on ANGEL rests on the Flow Meters and the Classifier. The ANGEL architecture allows for multiple Flow Meters to increase the overall capture rate. As classification is made on a per-flow basis, the single Flow Classifier can be implemented as a distributed system.

2.2 Classification

As discussed in [23], traditional traffic classification techniques can be problematic. Port-based classification systems, while requiring minimal processing effort, have low accuracy rates. Meanwhile, stateful reconstruction of session and application information impose significant complexity and processing load on the traffic identification device. Further, governments may impose privacy regulations constraining the ability of third parties to lawfully inspect packet payloads.

ANGEL makes use of Machine Learning (ML) [21] techniques to perform traffic classification. ML is a powerful technique for data mining and knowledge discovery, which searches for and describes useful structural patterns in data. An ML-based classifier classifies traffic by learning and recognising statistical patterns in externally observable attributes (such as packet lengths and inter-packet arrival times).

Use of Machine Learning techniques for traffic classification is not new. However most previous attempts use these techniques for post analysis of traffic [4, 5, 17]. ANGEL is unique in that is uses a real-time Machine Learning architecture first proposed by Nguyen and Armitage in [14] and [15].

ANGEL deploys a form of supervised learning where the classifier is built using input consisting of traffic flows with known classifications. After being constructed (trained), the classifier is able to use its classification model to classify new traffic flows.

It is worth noting that although we have adopted a Machine Learning approach as the classification method within ANGEL, our modular design means that alternate classification methods can be used should one choose to do so.

ANGEL deliberately separates flow classification from prioritisation. Just because a flow has been classified as a game does not mean it should be prioritised above all other flows. While highly interactive games such as First Person Shooters (FPS) can benefit from prioritisation, other game types such as role-playing or turn-based games do not require this treatment.

This division allows ANGEL to be extended to support new applications in the future by specifying how different flow classifications should be treated. In this way we allow for a future implementation where other application types (eg. streaming multimedia, VoIP) might also be classified and prioritised whilst P2P applications (eg. BitTorrent) may be given lower prioritisation.

3. SYSTEM FUNCTIONALITY

An entire ANGEL system (Figure 2) was built to test system functionality. Since consumer grade equipment does not implement the ANGEL protocol, the ANGEL enabled CPE device is run as software installed on the Client computer.

In all test scenarios, traffic was properly captured and classified by ANGEL. The Client Manager then communicated this information to the appropriate CPE device handling that flow **only** if that CPE device was ANGEL enabled **and** registered with the ANGEL system. An appropriate traffic prioritisation rule was installed at the CPE device.

When running two concurrent flows from the same client computer (game and file upload) the difference in performance when using ANGEL is visibly obvious. When ANGEL is not enabled, game performance is jerky and slow, resulting in difficulty in controlling player motions and aim-

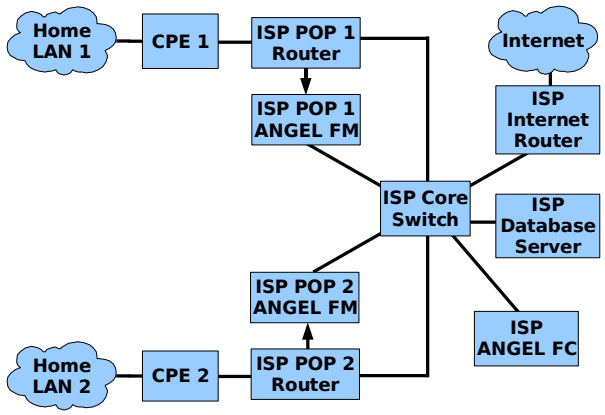

Figure 2: System Functionality Testbed

ing at a moving target. Once ANGEL was enabled on the underlying CPE device, the prioritisation rule was employed within 1 second and game play returned to normal.

Quantitavely, the effect of network delay and delay variation of FPS-type games has been shown by Zander et. al [16] while the benefits of traffic prioritisation to game-like traffic in terms of delay variation was shown by But et. al [11].

An interesting side-effect of continuously updating a flows classification over its lifetime is what happens during different phases of a game. In the case of ET, flow classification during downloading of game maps resulted in the flow classification changing to non-game, temporarily disabling prioritisation until actual gameplay resumed.

4. ANGEL PERFORMANCE

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

Our performance metrics are **accuracy**, **stability**, **timeliness** and **processing performance**. Accuracy is measured in terms of Recall and Precision rates. These metrics are often used to evaluate 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 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 as Recall is (1 - False Negatives) and Precision is (1 - False Positives).

While both Recall and Precision metrics are important, we treat Recall with higher priority. Precision indicates the proportion of flows classified as games that are not game flows - as long as this is not too low, the number of extra flows given priority will be minimal (as the flows are only competing with other flows from the same customer). We are less tolerant of falsely classifying game traffic - as the game will experience network conditions equal to (or worse with low Precision) a situation where ANGEL is not operating.

Unlike traditional classification schemes where traffic is classified once for the flow, ANGEL performs continuous, real-time classification. This is done using a sliding window technique where a traffic flow is segmented into windows of N packets (ANGEL uses N=25) and the flow is re-classified every time a full window of packets is seen. This has the potential to result in the classification for a flow changing over time. Another key measure of performance is classification stability - how stable is the classification of a flow over time.

Timeliness refers to how long it takes once a game flow's traffic has been captured before it has been classified by ANGEL, the notification sent to the CPE device, and a prioritisation rule put into effect. This measure is important as prioritisation should occur quickly enough that it is not noticeable to game players using ANGEL. The timeliness of the ANGEL system in classification is primarily measured in the Flow Meter and Flow Classifier.

The ANGEL Flow Meter and Flow Classifier are responsible for handling the majority of the processing load of ANGEL. In this respect, it is important to gauge the processing performance of these two modules in order to consider how ANGEL may function under a large number of users/flows. We consider the processing performance of the Flow Meter in terms of its packets per second (pps) and flows per second (fps) capture rate. Flow Classifier performance is considered in terms of its classification rate.

The classification model was implemented using a simple Naive Bayes Algorithm. The algorithm provides a classification system based on probabalistic knowledge. It is designed for use in supervised classification, where the goal is to accurately predict the class of unseen data using a model built on sample training data [12, 14, 15].

Our classification model is constructed with Wolfenstein Enemy Territory (ET) [20] traffic of a month-long trace collected during May 2005 at a public server in Australia [3]. Our non-game traffic consists of common Internet applications (HTTP, DNS, SSH, P2P (Kazza and Bittorent) and SMTP) from a 24-hour trace collected by the University of Twente, Germany, on February 6th 2004 [2] at an aggregated 1Gbps link. The model is constructed as stated in [14] and [15].

The ANGEL system and testing components' hardware specifications are presented in Table 1.

4.1 Accuracy

Figure 3 shows the Recall and Precision for ET traffic as classified by ANGEL models. M represents the number of packets that the classifier might miss from the beginning of the original game flow. The model achieved better than 96% Recall and 97.5% Precision for ET traffic.

These results confirm previous results in [15] and allow the classifier to correctly classify flows where the flow has already started and in the case of real-time, ongoing classification to ensure an accurate classification over time.

We also tested the classification model with ET traffic captured over a LAN (SONG dataset) [1], consisting of 39 game flows with total of 195,000 packets, the classifier correctly identified all 39 flows.

Table 1: Component Configurations

Component	Configuration
Flow Meter Traffic Generator	Intel Pentium4 3.00GHz with HyperThreading 1GB (2 x 512MB) DDR2 533 RAM Seagate ST380817AS 80GB SATA HDD Asus P5LD2-VM motherboard Running FreeBSD 6.1
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

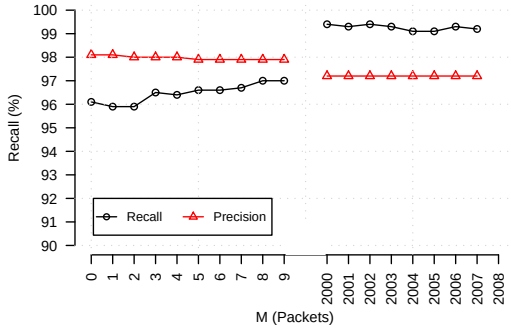

Figure 3: Recall and Precision for ANGEL classification model

4.2 Stability

While the ANGEL Classifier correctly identified all game flows in the previous test, in nearly 40% (15 of 39) of these flows, the classification changed state from one to six times during the flow lifetime. This result can occur because ANGEL continuously re-classifies flows in realtime.

In an effort to improve the classification stability, we developed the *"confirmed classification"* algorithm. Essentially the algorithm applies a low-pass filter to the classifier output, removing high frequency changes in classification. The algorithm works by changing the flow classification only when two consecutive, non-overlapping windows of packets generate the new classification.

The effectiveness of our algorithm is shown in Figures 4 and 5. Figure 4 shows that for ET traffic, the number of flows exhibiting changing classification results has dropped from 15 to 1 and that this flow classification changed state only twice (game - non-game - game). The key result here is that the stability of true positive classifications is improved, limiting the times in which prioritisation rules are removed for game flows.

We are also interested in the stability of classification for true negative classifications (non-game flows). While not as important as the classification of game flows, it is still beneficial - more stable classification reduce ANGEL signalling to the CPE devices and minimise the traffic competing with game flows for prioritisation.

We tested the stability of classification for a number of

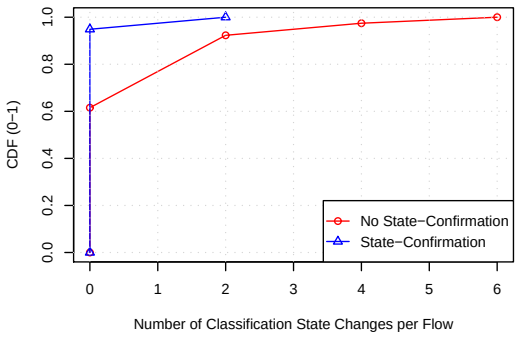

Figure 4: Classification stability for ET traffic

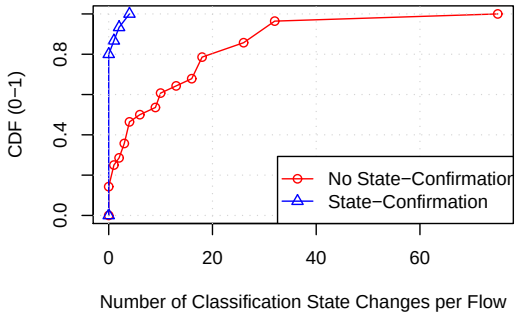

Figure 5: Classification stability for non-game (Kazaa) traffic

different applications, Figure 5 shows the results for Kazaa traffic. *"Confirmed classification"* has significantly increased the stability of the classification in terms of the percentage of flows with a single classification. Further, the number of classification changes for the flows has dropped from a maximum of 50 to 6 changes. Similar improvements have been seen for other non-game traffic flows.

4.3 Timeliness

A drawback to the *"confirmed classification"* algorithm is that to classify a game flow, at least two windows of packets must be classified - the classification result is delayed. ANGEL uses a classification window of 25 packets. For ET traffic, the packet rate for a bi-directional flow (client to server and server to client) is approximately 50PPS [8]. The use of *"confirmed classification"* would result in a doubling of the minimum classification time from approximately 0.5 seconds to approximately 1 second.

To evaluate the performance of the ANGEL classification model with real-time traffic classification, we use the TcpReplay tool[1] to replay pre-recorded tcpdump files at the original capture rate. The tcpdump files consists of ET traffic from the SONG dataset.

Figure 6 shows the CDF of the time taken by the classifier to identify game traffic from an ET game trace. In almost all cases (37 of 39 flows) the classification is made in under 0.75 seconds. In two outlier cases the classification takes up to 2.8 seconds. Further analysis of the packet trace for these two flows indicate a mean packet interarrival time of 50ms as opposed to 10ms for the other 37 cases. It is expected to

[1]Available at `http://www.sourceforge.net/projects/tcpreplay`

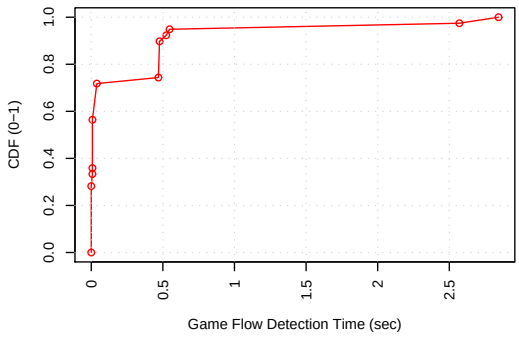

Figure 6: ET Game Flow Classification Timeliness

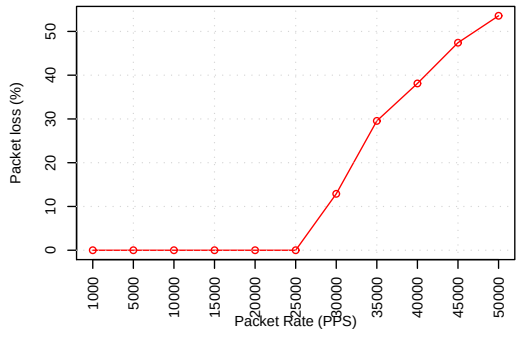

Figure 7: Flow Meter Packet Loss Rate vs PPS

take five times longer to build up the packet window such that a classification can be made.

4.4 Processing Performance

4.4.1 Flow Meter

The Flow Meter was tested in terms of packet and flow rates that could be captured and processed with negligible packet loss at the meter. These figures are important to help determine the number and location of Flow Meters that must be deployed within an ISP network. Of key importance are the maximum rates at which traffic can be reliably captured and processed.

The Flow Meter was initially tested for capture performance at high data rates - measuring the capture performance at different packet-per-second (PPS) rates. We are particularly interested in the number of lost packets - packets that are not captured and processed for delivery to the Flow Classifier. In our evaluation, we consider different PPS rates between 1,000 and 50,000pps in increments of 5,000pps.

Figure 7 shows packet loss rates within the Flow Meter (the difference between the number of packets offered to, and number of packets parsed by, the Flow Meter) as a function of offered load. Packet loss is negligible until the offered load exceeds 25,000pps.

Previous work [10] indicates that the packet loss rate can be affected by the configured PCAP buffer size. To minimise this, we have set the PCAP buffer size on the Flow Meter to 512MB. Our achieved packet loss rate compares favourably with the PLR performance of the underlying PCAP capture library. This indicates that the performance of the Flow Meter is limited by the underlying OS packet capture and PCAP implementation. This is also supported by other results that indicate that for all input packet rates, memory usage remained relatively constant at 5MB while (median) CPU usage was also under 30%.

4.4.2 Flow Classifier

The Flow Classifier implementation was tested under a worst case scenario - a single process classifying all flows. A multi-threaded/process implementation would enable the classifier to take advantage of idle CPU time classifying alternate flows while waiting for hardware IO to conclude. This may enable better use of multi-processor and multi-core hardware, potentially increasing the maximum processing rate.

To evaluate the performance of the classifier in terms of packet and flow rates, we generated a synthetic trace file based on the capture of one Enemy Territory session from the SONG dataset. This trace was duplicated to create multiple traces with different numbers of concurrent game sessions (and PPS rates). These trace files were then replayed to the Flow Meter which subsequently passed the data to the Flow Classifier for classification.

Given our performance measurements on the Flow Meter, we limited the generated traffic rate to 25,000PPS (500 concurrent game flows). At this input rate, the Flow Meter successfully captured all packets which were then passed to the Flow Classifier. At the Classifier, all flows were correctly classified as game traffic (100% accuracy).

Significantly, the classifier ran with a total CPU load of under 0.2% and memory footprint of under 5MB. This suggests that the bottleneck in ANGEL lies at the Flow Meter. Our system test verifies the functionality of running ANGEL with multiple Flow Meters to overcome this bottleneck.

5. CONCLUSION

ANGEL is an architecture to be deployed within the ISP to detect and classify network flows. Upon classification of network game flows, ANGEL would communicate this information to any registered ANGEL enabled CPE devices that handles the flow. The CPE device then installs a traffic prioritisation rule to provide class-based QoS for these flows.

We have built a working ANGEL system that can be deployed and provided code to implement the CPE side of the ANGEL protocol. ANGEL continuously re-classifies a flow for every consecutive, non-overlapping window of captured packets. Our results indicate that the ANGEL system is able to correctly classifying game traffic with greater than 96% accuracy. The use of our *"confirmed classification"* algorithm significantly improves the stability of flow classification, reducing the number of classification changes during a flows lifetime. This has the effect of improving accuracy by ignoring one-off changes in flow classification.

Performance tests indicate that the bottleneck in system performance is likely to be the Flow Meter, where performance was limited by the underlying packet capture facilities rather than the Flow Meter implementation. Using our test platform, we were able to capture packets at a rate of 25,000PPS on a single Flow Meter. It is however possible to scale ANGEL by deploying more Flow Meters in the network as network topology and traffic flows dictate.

Performance tests of the Flow Classifier indicate that classifying traffic using a Machine Learning approach can scale to large numbers of flows. Our tests in classifying traffic

captured by a single Flow Meter at 25,000PPS show that this can be performed using only 0.2% CPU load with a small memory footprint.

In terms of user-perceived performance, game traffic is typically classified and prioritisation rules established within 1 second of the first packet being captured. ANGEL is also successful in classifying flows when traffic is captured after the flow has commenced.

As ANGEL is a modular system based on Machine Learning techniques to classify traffic in real-time, it is envisaged that future development could extend ANGEL to support prioritisation for other real-time traffic classes (eg. VoIP).

6. ACKNOWLEDGMENTS

This work was supported by the Smart Internet Technology Cooperative Research Centre. http://www.smartinternet.com.au

7. REFERENCES

[1] SONG - Simulating Online Networked Games Database, Smart Internet CRC, May 2007. `http://caia.swin.edu.au/sitcrc/staticpages/index.php?page=song`.

[2] University of twente - traffic measurement data repository, as of 26th March 2006. http://m2c-a.cs.utwente.nl/repository.

[3] Et server, as of 27th April 2006. http://gs.act.grangenet.net.

[4] A. McGregor, M. Hall, P. Lorier, J. Brunskill. Flow Clustering Using Machine Learning Techniques. In *Passive & Active Measurement Workshop*, April 2004.

[5] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *ACM SIGMETRICS*, June 2005.

[6] J. But, N. Williams, S. Zander, L. Stewart, and G. Armitage. Automated Network Games Enhancement Layer - a proposed architecture. In *Proceedings of the 5th Workshop on Network and System Support for Games (NetGames2006)*, October 2006.

[7] D-Link. 108G Gaming Router. http://games.dlink.com/products/?pid=370.

[8] G. Armitage, M. Claypool, P. Branch. *Networking and Online Games - Understanding and Engineering Multiplayer Internet Games*. John Wiley & Sons (ISBN: 0470018577), April 2006.

[9] G. J. Armitage. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake3. In *Proceedings 11th IEEE International Conference on Networks (ICON)*, September 2003.

[10] J. But and J. Bussiere. Improving NetSniff Capture Performance on FreeBSD by Increasing the PCAP Capture Buffer Size. Technical Report 051027A, CAIA, October 2005. `http://caia.swin.edu.au/reports/051027A/CAIA-TR-051027A.pdf`.

[11] J. But, S. Burriss, G. Armitage. Priority Queuing of Network Game Traffic over a DOCSIS Cable Modem Link. In *Proceedings of Australian Telecommunications and Network Applications Conference (ATNAC)*, December 2006.

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

[13] M. Dick, O. Wellnitz, L. Wolf. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of ACM Network and System Support for Games (NetGames) Workshop*, October 2005.

[14] T. Nguyen and G. Armitage. Synthetic Sub-flow Pairs for Timely and Stable IP Traffic Identification. In *Proceedings of the Australian Telecommunication Networks and Application Conference*, Melbourne, Australia, 2006.

[15] 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, 2006.

[16] S. Zander, G. Armitage. Empirically Measuring the QoS Sensitivity of Interactive Online Game Players. In *Proceedings of Australian Telecommunications and Network Applications Conference (ATNAC)*, December 2004.

[17] S. Zander, T.T.T. Nguyen, G. Armitage. Automated Traffic Classification and Application Identification using Machine Learning. In *IEEE 30th Conference on Local Computer Networks (LCN 2005)*, November 2005.

[18] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games (NetGames) Workshop*, August 2004.

[19] T. Henderson, S. Bhati. Networked games - a QoS-sensitive application for QoS-insensitive users? In *Proccedings of SIGCOMM RIPQoS Workshop*, August 2003.

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

[21] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), December 1997.

[22] Ubicom Inc. Solving Performance Problems with Interactive Applications in a Broadband Environment using StreamEngine Technology, October 2004.

[23] 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 Communication Review*, October 2006.