An Application of Pattern Matching for the Adjustment of Quality of Service Metrics

Doug Legge and Atta Badii
IMSS
University of Reading
Reading, England
{d.j.s.legge, atta.badii}@reading.ac.uk

Abstract—**Quality of Service is an important component of Internet Protocol traffic as it allows a prioritisation of designated applications during periods of high utilisation or where there is restricted resource, such that the end-user experience can be optimised. Typically, Quality of Service is defined through manual policies with expert human input required. In this paper, we present initial support for the hypothesis that the definition and on-going change management of manual Quality of Service policies can be replaced through the use of pattern matching techniques, which classify traffic *in real-time*. This paper serves as an introduction to concepts, which may be new to many Internet Protocol-based network engineers, and as a motivation for those in the field of Artificial Intelligence, machine-learning, as to where advanced learning functions could be applied.**

Keywords*-Quality of Service (QoS); self organizing map (SOM); k Nearest Neighbour (kNN); agent*

I. CONCEPT

Quality of Service (QoS) policies provide for the prioritisation of packets on IP networks, partly motivated in the early 2001 by the convergence of voice and data traffic.

Despite the increasing availability of high-speed consumer (e.g., xDSL ≈50Mbps) and corporate (e.g., Ethernet ≈10Mbps) data links, Internet Protocol (IP) engineers now face the conundrum, once seen in the Personal Computer (PC) world, where a faster network resource is rapidly consumed at an increasing rate by bandwidth hogging applications, such as Videoconferencing or Tele-presence. As a result little planned capacity overhead remains and QoS mechanisms continue to be required.

With no end in sight, the dependence of QoS implementation upon expert judgement leaves organisations exposed to high salary costs and a potential loss of critical knowledge resulting from staff churn. In addition, these existing optimisation techniques lead to increasingly complex network operations regards the service mechanisms required to deliver appropriate application performance. Including those supporting activities such as ‘requirements analysis’ and ‘policy change management control’. Verma [1] states “as networks make the transition from *all traffic is equal* to the new model in which some *traffic is more equal than others*”, a way must be found in which to specify differentiate and service traffic types on the network whilst maintaining a simplified abstraction.

Whilst optimisation of traffic-flow is a valid and well researched field, simplification of its engineering and on-going management has, in the first author’s experience as the IT Operations Manager for a UK FTSE 250 company, been long overdue at the coalface of network support.

For some years now machine learning, and in particular ‘neural-networks’, have been used within many industries [2][3][4]. However, it is in the field of data mining that neural networks have been most productive, being used to “extract new information, from existing data, thus providing innovative insights and tactical commercial benefit” [5].

The authors’ previous work [6][7] highlighted those issues corporate organisations face regarding this ‘requirements analysis phase’ necessary to collate that information required to build network traffic services (e.g., QoS policy statements). This work demonstrated how these policies in practice remain sub-optimally implemented through lack of a facility to allow their dynamic adaptation responsive to changing circumstances and thus changing priorities of different business data traffic types.

In this report the authors’ present paper results from initial experimentation regards how varying traffics differ in *sensitivity*, and how that ‘footprint’ within IP packets could be used to characterise data for machine-learning.

This shows how autonomous agents could be devised such that they were capable of traffic categorisation and dynamic differential, reallocation of computer network bandwidth, to various business data streams according to their relative dynamic priorities. Such agents could then reduce the reliance and complexity of current (human) expert QoS policy definition through the deployment of machine-learning techniques for the re-classification of the IP traffic. This paper also introduces the platform on which further experimentation will be completed.

II. APPLICATION SENSITIVITY, DELAY AND PROCESSING

Certain Internet traffic applications are time-critical. The stutter arising from delayed packets often renders videoconferencing unusable. For any System for Intelligent Network Control (SINC) to be adopted it must satisfy the end-user delivery requirements [6]. A summary of sensitivities is given in Table 1.

TABLE I. APPLICATION SENSITIVITIES

Traffic Type	**Sensitivities**			
	Bandwidth	***Loss***	***Delay***	***Jitter***
Voice (set-up)	Very low	Medium	High	High
eCommerce	Low	High	High	Low
Transactions	Low	High	High	Low
Email	Low	High	Low	Low
Telnet	Low	High	Medium	Low
Casual browsing	Low	Medium	Medium	Low
Serious browsing	Medium	High	High	Low
File transfer	High	Medium	Low	Low
ICA	Medium	Medium	High	Medium
Video conferencing	High	Medium	High	High
Multicast	High	High	High	High

Any rule-based system must therefore respect such end-user observable concerns such as: *delay*, defined as *a lapse of time*, which includes *jitter*, often defined as a *packet delay variation* (PDV) used as a measure of the 'variability over time' of the packet latency across a network. In traditional QoS deployments a set of common applications, group of users, or business performance requirements, can be profiled and a template developed for that application and each resulting flow. Thus each flow which fits that profile can be treated the same, reducing the cost of replicating flow information for similar flows. The authors' previous paper [7] drew on an observation-action pair mapping, or "*policy of an agent*" [8], shown in equation 1 below:

$$F \ \theta_t = \alpha_t \tag{1}$$

in which a stateless function F maps its current observation (of network traffic and available resource) to a new action, representing a classification of the data, and $_t$ is the budget (e.g., time) in which the observation is made and the mapping completed. As an example of relevance to this paper, consider the *rule* for an attribute found within IP traffic, such as packet length, shown in equation 2 below:

$$\textit{packet\_length:} \quad \begin{array}{l} \leq y \rightarrow a \\ > y \rightarrow b \end{array} \tag{2}$$

where y, in this instance is packet length in Bytes and a and b exhaust all possible classifications of that packet. An illustration being packets ≤ 80 Bytes are classified a, where a equals a classification of 'Expedite Forward', and all other packets (e. g., > 80 Bytes) are classified b, where b equals a classification of, in this instance, 'AF21'.

Such tests are, however, dependent on a known *typing* of net packets: given variability of the packet structure, simple rules are likely to misclassify traffic, with a resulting incorrect prioritisation. This motivates our research to classify packets based on a "black-box" (unsupervised) approach, by which *a priori* unknown packet structures can be presented to the learning-function for classification based on *learned characteristics* (e.g., voice packets which have a high QoS priority, have these known sensitivities). Where this characterisation is completed using some or all of those attributes available within an IP packet, but where the attribute choice is not fixed, thus allowing for an assessment of belief in a hypothesis to be updated with new data at each *observation epoch*. This set of attributes provides a 'frame of discernment' Θ [9].

III. LEARNING FOR AN INTELLIGENT NETWORK CONTROL

There are many well-known mechanisms for the useful characterisation of data [10][11][12] including:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Evidence from the authors' background research indicated that the use of pattern matching is effective at finding previously unseen patterns within the dataset, and given IP networks have vast sums of data traversing networks, with each packet or frame having an inherent *footprint* resultant from its header(s), this would appear to offer a suitable mechanism for intelligent network control.

There is no lack of data in the typical network [13] making statistical analysis techniques of traffic a relatively easy task. The challenge to the data classification is to accurately classify traffic in real-time. Initial experimentation with the first author's corporate network has focused on the classification of traffic flows using a *k*-Nearest Neighbour (*k*NN) clustering feature. Tarassenko [14] defines the objective of any clustering as:

"Given P patterns in n-dimensional space, find a partition of the patterns into K groups, or clusters, such that the patterns in a cluster are more similar to each other than to patterns in different clusters."

Clustering requires the characterisation of input data within a multidimensional space; in the context of this research we characterise data as the five-tuple:

- source IP address;
- destination IP address;
- source port number;
- destination port number;
- protocol;

- plus additional attributes including length and frame_time_delta.

The authors' have adapted the framework of [15] to identify a classification process that isolates differences within a population of network traffic, each having different a *model* (or description). The process by which this is achieved is defined below, and whilst the final realisation of this research is expected to be deployed using Application Specific Interface Card (ADIC) or Programmable Logic modules, integrated within internetwork devices in much the same way as the WAN Interface Cards (WIC) seen in routers, Fig. 3 shows the current system in development:

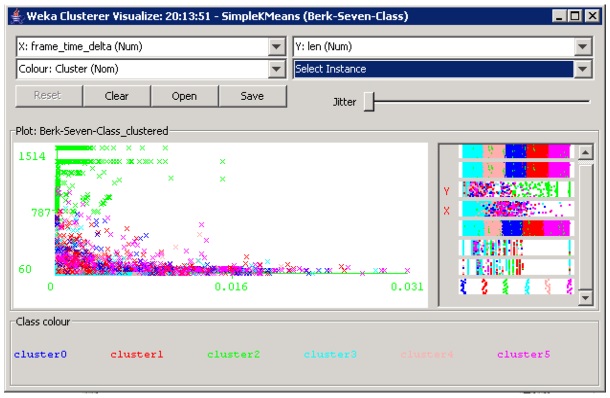
1. **Sensing:** input to the system, the packets arriving on the ingress interface
2. **Pre-process:** signals are pre-processed such that they can be transposed for subsequent operations without loosing relevant processing information. This may use a *segmentation* function to isolate the *features* of the data from each other or from background noise. One such segmentation would be to separate UDP from TCP traffic as each has a differing underlying network requirement.
3. **Feature Extraction:** whose purpose is to reduce the data by measuring certain features or properties, which in turn are passed to a
4. **Classifier:** which evaluates the evidence and makes a decision as to the queue in which the traffic will be transmitted
5. **Post-processing:** Are those processes engaged after the classifier required to return the newly classified traffic to a network egress interface

Of course, the use of *k*NN is not new, however, with networks and the data they handle within a QoS setting highly time critical, we cannot allow the real-time classification of data detrimentally to slow it. In particular, there are industry standards for node processing: the optimal decision boundary which accepts a level of error within the classification to ensure any process keeps within any *overall budget* defined (e.g., RTD) should be less than 150ms according to ITU-T G.114 [16]. The novelty in our research includes, therefore, the engineering of a classification algorithm which will prevent the modelling of extremely complex dimensional dependencies. It was this requirement of visualisation that led the authors to the thought of Pattern Matching for QoS, and which the authors' can now model using that framework of [15] where:

1. **Sensing:**
   - **a.** Traffic is generated from a production network or simulated on a laboratory environment, using client>server transactions, or application simulation software
   - **b.** That traffic generated is captured using network protocol analysis software (e.g., Wireshark; Etherpeek) and saved as a .cap file for analysis

2. **Pre-process:**
   - **a.** The traffic is exported as an .XML compliant .pdlm file such that it can be imported into spread-sheet applications for statistical analysis
   - **b.** The resultant .xml file is loaded into a text reader (e.g., textpad) and searched for non ASCII characters, which would prevent import to those spread-sheet applications
   - **c.** At this stage the .xml data requires to be transposed such that each *attribute* (e.g., *frame_length*) is a column header, and each instance of that attribute (per packet or frame) is a row of known variable type (e.g., nominal, string etc.). This is completed by a SQL transpose routine, the output of which is a reordered .xml file.
   - **d.** That .xml file is opened within Microsoft Excel to ensure consistency of data. This includes missing or errored cells, inconsistent format or corrupt file.
   - **e.** This data file is presented to SPSS PASW, Weka, and Matlab software which enables a number of statistical analysis visualisations to be completed, such as a scatterplot of Frame_length over Time, shown at Fig. 1, "Visualisation of Captured Data in WEKA" below.
   - **f.** A suitable file (e.g., Weka .arf; Matlab .mat) can now be built such that varying learning processes can be explored and evaluated.

3. **Feature Extraction:**
   - **a.** Feature exaction then is the reduction of data to its principle features. In the case of Internetwork traffic this has previously been defined as the five tuple with additional attributes, including but not exclusive to:
     - ip_src
     - ip_dst
     - ip_srcport (tcp or udp)
     - ip_dstport (tcp or udp)
     - prot
     - frame_pkt_length
     - frame_time_delta
     - frame_time_relative



X: frame_time_delta (Num)
Y: len (Num)
Colour: Cluster (Nom)
Select Instance

Figure 1. Visualisation of Captured Data in WEKA.

Equation 3 illustrates how three 'characterisation features', say packet_length (x_a), time_frame_delta (x_b) and ip_dstport (x_c), could be used to characterise a packet as distinct from any other packet not within the same cluster. This would present a feature vector in a two-dimensional feature space [14], where:

$$\begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix} \qquad (3)$$

This use of feature extraction can be illustrated with an adaptation of [14] shown in Fig. 2, where these three attributes are presented to the clustering function (*F*) which has then defined the packets in two distinct clusters with an individual mean of *X*, where:

X_n = the packets captured within time t

(where time t is from the start of capture to the end of the capture processed) with features

$X = \{x_a, x_b, x_c\}$

C_n = the clusters of the set $K = \{C_1, C_2, \ldots\ldots, C_n\}$

$\otimes$= the cluster centre characterised in this instance using the mean vector m_k

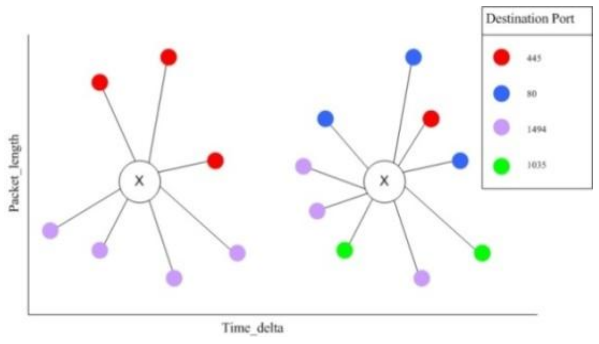


Figure 2. Adaption of Features of a Cluster [14]

This difference between clusters typically measured by the *distance between them* (such as Euclidean distance or Pearson correlation) to define a *closeness* or interrelationship of the traffic. The known issue of Euclidean distance, which assigns more *weight* (i.e., preference) to features with large range could be further optimised through the use of *Manhattan* (or block) distance, such that higher powers increase the influence of large (neighbour) differences at the expense of small ones. An example being, if presented with two features: ip_dstport and frame_pkt_length, with Euclidean distance there would likely be a preference to the former. This is due to the legal range of IP ports being in the range is 0 to 65,535, compared to a standard Ethernet frame *Maximum Transmission Unit* (MTU) of only 1500 (bytes).

That traffic, which is clustered based on its perceived characteristic, as demonstrated in Fig. 2, can then be marked using coding such as Differentiated Services Code Point (DSCP). That cluster of traffic which is defined as being the most *sensitive* (such as voice traffic) or of having the *highest business importance* would then be marked with an appropriate classification, such as Expedite Forward (EF). Each of the six clusters (0-5) would then be forwarded to one of six pre-configured virtual hardware queues within the network devices egress interface.

IV. SUMMARY AND CURRENT RESEARCH

This paper built on an early "feasibility study" to investigate how the agent models, as conceived in the authors' previous papers [6][7], would be implemented to influence internetwork traffic management. Whilst the technique is promising, the current manual transposition of data for the machine-learning application means that only off-line processing is possible. However a realistic implementation requires a platform with the ability to perform online, real-time automated transposition of QoS targets for various data steams intended for presentation to the machine-learning mechanism.

Such a system, initially online if not real-time, has been implemented on a Linux based server running the Snort Intrusion Detection System (IDS) [17]. Ingress network traffic is *sensed* (captured) and updated within a MySQL database table. From the database, rather than using the traditional Snort rule-base, the instances (packets) within the various tables are JOINED and a Python language script presents the data to an Orange k-means clustering algorithm [18]. This algorithm performs the function *F* described, completing the *pre-processing*, *feature extraction, and classification* including *visualisation*. Further research investigates those activities related to *post-processing*, such as the *marking* of that traffic based on Differentiated Services Code Point (DSCP), and *dispatch* routines which, at present, is the re-population of the database with the newly reordered traffic ready for network transmission on the appropriate egress interface.

In addition, further investigation into other Python-based pattern classification applications, such as pyMVPA [19], PyBrain [20], and OpenElectrophy [21] will be progressed to see whether they can perform the work more efficiently.

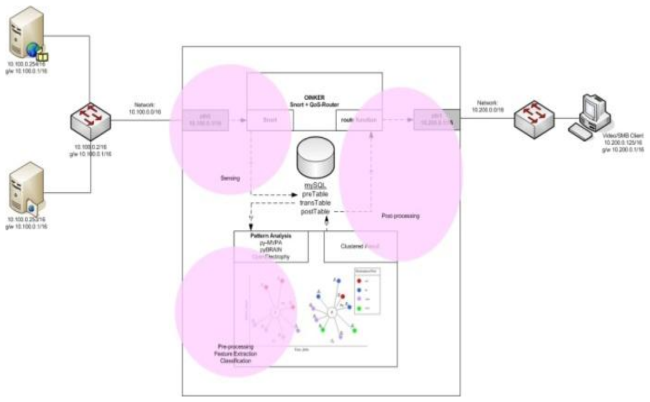


Figure 3. Oinker: Snort-based Systen for QoS Route Forwarding using Pattern Classification

ACKNOWLEDGMENT

The authors' would like to express their sincere thanks to Dr. Jon Hall, Open University, for his assistance in the completion of this paper.

REFERENCES

[1] D. C., Verma, "Policy-based networking architecture and algorithms", Technology Series, New Riders, Indianapolis, Indiana, pp. 6, November 2000.

[2] S. De Lurgio, "Predicting micro-loan defaults using probabilistic neural networks", Credit & Financial Management Review, (Internet), 2002, http://www.allbusiness.com/finance/1049916-1.html, (accessed 19th July 2010).

[3] M. Buchanan, "Why complex systems do better without us", New Scientist, Reed Business Information Ltd., Issue 2668, pp. 28-31, August 2008.

[4] C. J., Tebelskis, "Speech recognition using neural networks", an unpublished thesis, (Internet), 1995, http://portal.acm.org/citation.cfm?id=239333&dl=GUIDE&coll=GUIDE&CFID=97551353&CFTOKEN=37801874, (accessed 26th July 2010).

[5] I. H, Whitten and E. Frank, "Data mining: practical machine learning tools & techniques, Second Edition, Elsevier, San Francisco, 2005

[6] D. Legge and A. Badii, "Conceptualisation of an application of adaptive synthetic socioeconomic agents for intelligent network control", 2nd PERADA Workshop on Pervasive Adaptation, Edinburgh, AISB, pp. 14-21, April 2009.

[7] D. Legge and A. Badii, "A primer for an application of adaptive Synthetic Socioeconomic Agents for Intelligent Network Control", in press, 2nd School of Systems Engineering Conference, University of Reading, December 2009.

[8] N. Vlassis, "A concise introduction to multi-agent systems and distributed artificial intelligence, Morgan & Claypool, California, pp. 55-57, 2007.

[9] R. Callan, "Artificial Intelligence", Palgrave Macmillan, Basingstoke, Hampshire, pp. 163, 2003.

[10] J. P, Bigus, "Data mining with neural networks: Solving business problems-from application development to decision support", Computing McGraw-Hill, New York, pp. 6-29, 1996.

[11] J. F, Sowa, "Conceptual structures: InformationpProcessing in mind and machine", The Systems Programming Series, Addison-Wesley Publishing, Reading, Massachusetts, pp. 281-292, 1986.

[12] G. Marshall, "Advanced students' guide to expert systems", Heinemann Newnes, Oxford, pp. 128-142, 1990.

[13] S. Sui, and C. Zhixiong, "Adaptive network flow clustering", IEEE International Conference on Networking, Sensing and Control, pp. 596-601, April 2007.

[14] L. Tarassenko, "A guide to neural computing applications", Arnold, London, pp. 20-23, 1998.

[15] R. O. Duda, P. E. Hart, and D. G., Stork, "Pattern classification", Chichester, Wiley, New York, pp. 3-23, 2000.

[16] ITU-T G.114 "One-way transmission time, series G: Transmission systems and media, digital systems and networks international telephone connections and circuits–general recommendations on the transmission quality for an entire international telephone connection", Telecommunication Standardization Sector of the International Telecommunication Union, pp. 2-12, May 2003.

[17] Snort, "a lightweight open source network intrusion prevention and detection system (IDS/IPS)", (Internet), 2010, http://www.snort.org, (accessed 26th July 2010).

[18] Orange, Laboratory of Artificial Intelligence, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, http://www.ailab.si/orange/doc/modules/orngClustering.htm, (accessed 20th July 2010).

[19] M. Hanke, Y. O. Halchenko, P.B. Sederberg, S.J. Hanson, J.V. Haxby, and S.Pollmann, "PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data", Neuroinformatics, volume 7, pp. 37-53, 2010.

[20] T. Schaul, J. Bayer, D. Wierstra, and Y. Sun, "PyBrain", Journal of Machine Learning Research 11, pp. 743-746 Submitted 11/09; Published 2/10, (Internet), 2010, http://www.idsia.ch/~tom/publications/pybrain.pdf, ((accessed 26th July 2010).

[21] S. Garcia and N. Fourcaud-Trocmé, "OpenElectrophy: an electrophysiological data- and analysis-sharing framework", Frontiers in Neuroinformatics, Volume 3:14, (Internet), 2010, http://frontiersin.org/neuroinformatics/10.3389/neuro.11.014.2009/full,(accessed 26th July 2010).