Autonomic Diagnosis of Anomalous Network Traffic

Angelos K. Marnerides, David Hutchison
Computing Department, Infolab21
Lancaster University
Lancaster, UK
{a.marnerides,dh}@comp.lancs.ac.uk

Dimitrios P. Pezaros
Department of Computing Science
University of Glasgow
Glasgow, UK
dp@dcs.gla.ac.uk

Abstract— **Network traffic abnormalities pose one of the greatest threats for networked environments. Autonomic communications offer a solution: it should be possible to design network mechanisms that behave adaptively and respond to any anomalous phenomenon that threatens normal network behaviour. In this paper we present the design of an adaptive anomaly detection component that has been built as part of an autonomic network system. We have implemented an entropy estimator to predict the onset of anomalous traffic behaviour within an autonomic resilience framework, and a Supervised Naïve Bayesian classifier which synergistically empower the core properties of self-adaptation, self-learning and self-protection for next generation networks. Being part of an always-on, automated measurement and control infrastructure, such mechanism enforces the adaptive system reaction to suboptimal network operation and its subsequent restoration, while requiring minimal static (re)configuration and operator intervention.**

Keywords- Autonomic Networks; Resilience; Anomaly detection; Ttraffic classifier

I. INTRODUCTION

An increasingly important requirement for next generation networks is that they exhibit autonomic behaviour in order to minimise the need for static configuration and operator intervention. Autonomic architectures are being designed to employ self-* mechanisms such as self-adaptation, self-configuration, self-awareness and self-protection, based on always-on automated measurement, monitoring and control of their networked and system components. The diagnosis of anomalous traffic in autonomic networked environments in particular, poses great challenges that are hard to be confronted from both a system and a network perspective. By definition, autonomic systems are required to behave intelligently and to adapt their operation at the onset of sudden changes in the network traffic, so that they can mitigate the effects of malicious or legitimate processes that threaten the system with resource starvation.

However, due to the diversity and the dynamic behaviour of sources of suboptimal network operation, the entirety of anomalous traffic cannot be classified by statically using a single detection methodology. It is therefore required for a system to be supported by mechanisms that adapt to the operational traffic dynamics and provide reasonably accurate conclusions regarding traffic behaviour.

This work was supported in part by the EU FP6-IST Autonomic Network Architecture (ANA) Project (FP6-IST-27489).

Work done in the area of measurement-based anomaly detection has provided valuable results presenting numerous detection/classification techniques such as those reported in [2][3][4]. Most of these methodologies can be instrumented within an autonomic context employing principles of self-learning and self-adaptation without the need of external operator guidance.

We argue that our proposed design and implementation is a contribution towards the practical and theoretical instrumentation of the aforementioned principles. Our anomaly diagnosis engine provides for adaptive prediction and further categorization of network anomalies. In this way, abnormalities caused by legitimate or malicious intentions may be initially predicted, detected and at the same time be classified on a real-time scenario. These abilities are achieved due to the collaborative behaviour that our architecture enforces on the selected algorithms we have implemented.

In addition, our diagnosis framework operates within an overall resilience architecture that we also present in this document. Within the context of resilient networking, a system or network is required to keep its operation under an acceptable service level at the onset of various threats. Our proposed diagnosis component acts as the threat detection engine where it will be the unit in charge for notifying a mitigation engine in order to confront the threat in real-time. Hence, our framework implicitly serves one extra autonomic capability that of self-optimization. At the same time, it exhibits a pluggable behaviour which facilitates autonomic network nodes running our resilience architecture to dynamically employ a desired algorithm either for prediction or for classification, in order to customise the process of network behaviour diagnosis. We have used the Autonomic Network Architecture (ANA) node infrastructure to deploy the measurement-based diagnosis component, since it fulfils the core autonomic requirements and at the same time offers flexible design capabilities [1].

The remainder of this paper is structured as follows: section II briefly describes our resilience architecture that hosts the diagnosis mechanism, and introduces the ANA node infrastructure. Section III shows the engineering and internal algorithmic design of our detection framework where Section IV is purely dedicated on presenting the implementation of our prototype. Section V discusses the achievements and potential

of the proposed system, discusses future work and concludes the paper.

II. RESILIENCE ARCHITECTURE

Autonomic network environments are required to be resilient. Resilience is defined as the ability for a network to provide and maintain an acceptable level of service in the face of various challenges to normal operation [7].

Our resilience architecture adopts a modular design that uses information from a distinct monitoring facility [8]. It is composed by two core Functional Blocks (FBs), the Detection Engine (DE) and the Remediation Engine (RE). In ANA, any protocol entity generating, consuming, processing and forwarding information is abstracted as a FB, which may reside locally on an ANA *node*-host or it may be distributed across many hosts on a same or different *compartments*. A compartment is a primitive abstraction within the ANA terminology and is responsible for determining how FBs cooperate in order to provide particular functionality (or a service) to the data, control, and management planes [5]. In general, a compartment refers to the most absolute network entity which is autonomous and implements all the operational and administrative rules for a given communication context.

In simple terms, an ANA *node* is a local means for message exchange between FBs [5]. At the same time, one or more ANA *nodes* can map directly onto a physical networked host, or a single ANA *node* may span across multiple physical nodes. In reality, the ANA *node* is considered as the collection of mandatory and optional software components that in practice allow a physical device (e.g. router, switch, computer, sensor) to "run ANA" [6]. It is mainly composed of three core elements: the Minimal Infrastructure for Maximal Extensibility (MINMEX), the ANA playground and the ANA hardware abstraction layer. The ANA MINMEX provides the basic low-level functionalities which are required to bootstrap and run ANA. In parallel, it facilitates the generic sets of methods (API) that are used by "clients" of the MINMEX (i.e. protocols, applications).

The ANA Playground acts as a development framework and couples with a dedicated execution environment, where the more advanced and complex networking functionalities of ANA are placed [6]. This is the place where both commodity and bespoke functionalities are hosted. Commodity elements are "public" components that one can re-use such as cryptographic primitives, compression schemes and error recovery codes.

The resilience architecture we have designed and implemented resides within the ANA Playground. Fig. 1 shows a conceptual representation in respect to the location of our architecture within an ANA node. As already mentioned, we conduct resilience instrumentation using a monitoring facility that also resides within the Playground.

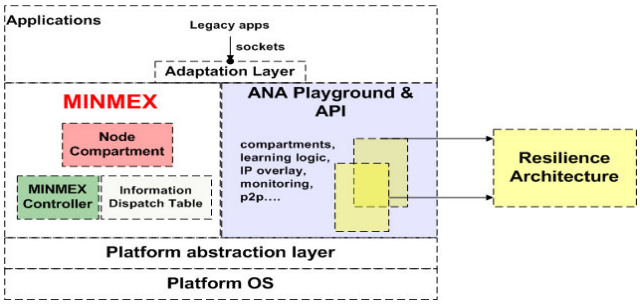

Figure 1: Resilience architecture within an ANA node.

Fig. 2 shows the resilience architecture emphasizing on the basic data flow interactions between the FBs. Initially, a monitoring entity sends real-time per-flow information of monitored traffic to the DE on a designated interface which in ANA terms we refer to as the Information Dispatch Point (IDP). IDPs act as generic communication pivots between the various FBs and they offer the advantage of re-organising communication paths between FBs. In addition, IDPs permit to implement forwarding tables which are fully decoupled from addresses and names: i.e., in ANA, the next hop "entity" (local or remote) is always identified by an IDP. This allows easily adding (and using) new networking technology and protocols as long as they export their communication services as IDPs do not revise any of the current designations.

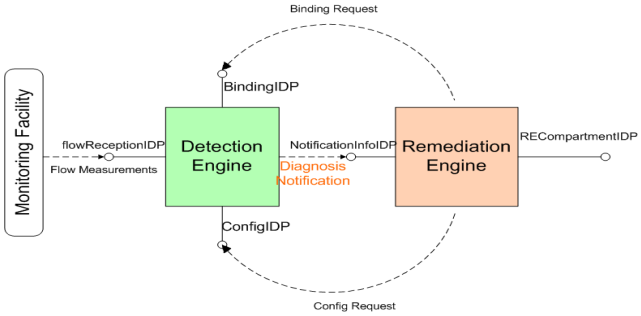

Figure 2: The Resilience Architecture.

After receiving per-flow data, the DE internally performs entropy estimation on selected flow features in order to detect an anomaly, and subsequently applies runtime classification via a supervised Naïve Bayes estimator. By these actions, the DE is eligible to decide whether the observed flows are the result of a local or compartment-wide anomaly. As soon as a decision regarding the precise nature of the anomaly is made, the DE composes a summary specification message that sends to its closest RE. Depending on the nature and causes of the identified event, and according to its classification by the DE, an appropriate remediation action is taken by RE. For example, in response to an attack, traffic shaping and possibly host blacklisting can be enforced, whereas in case of legitimate very high traffic demands towards particular network resources, appropriate load balancing algorithms are put in place to enable fast content propagation [16]. In case of a compartment-wide threat, the RE distributes the threat

information to regionally-close REs. These subsequently decide at which compartment region they should take action for confronting the event. Nevertheless, further description of the internals of the Remediation Engine is outside the scope of this paper, and the interested reader should refer to [16].

III. THE DETECTION ENGINE DESIGN

A. An Anomaly Diagnosis Framework

A milestone in our design was to identify the correct methodologies to apply in our prototype. Recently, considerable attention has been drawn to the analysis of the distributions of selected traffic features (i.e. combination of fields in packet headers) without the need for analysing large traffic volumes.

It has been mentioned in [10] that several anomalies are coupled with particular features present on each packet, and that their direct observation would, in theory, cause less processing cost and increased efficiency on the extraction of abnormal traffic characteristics. This work has evaluated the practical feasibility and detailed accuracy of selecting particular traffic features as opposed to the old volume-based traffic analysis methods, such as those presented in [2][12]. Alongside the practical feasibility, there were also some promising results showing that initial entropy estimation on the selected features provides a valuable heuristic for anomaly detection. Therefore, we have included the entropy metric as a basic initial detection scheme.

A topic that always accompanies detection frameworks is that of traffic classification whose challenges are unmet. Techniques employed by most classification mechanisms have been based on traffic volume metrics that are not reliable in providing sufficient information with respect to the structure of a specific phenomenon. For the classification of particular anomalies, we had initially considered several techniques from past literature but the most promising was the one presented in [11][13]. It is based on a Naïve Bayesian Classifier that accurately classified events whose traffic characteristics look almost identical from a volume-based perspective (e.g. WWW traffic from attack traffic).

Apart from the generic requirements of reasonably accurate and cost-effective detection/classification, the basic algorithms for the initial prototype were chosen based on their similarity in theoretically treating traffic behaviour. As we explain in the next section, both entropy estimation and Bayesian classification adopt a probabilistic, stochastic view of operational traffic. We have selected the particular combination since we have a strong hypothesis that traffic behaviour within autonomic environments is dynamic and random, mainly due to the diversity of networks that ANA incorporates. As we explain next, ANA's principles as well as the diversity and dynamicity within traffic subsequently lead to a hypothesis that spotted anomalies would promote discriminative properties beyond those already defined in traditional backbone networks.

Within ANA, inter and intra-compartment communication sessions go beyond traditional layering since they are achieved by abstractions such as the IDPs and compartments. Even though they facilitate node interaction under heterogeneous networks, they enable opaqueness to any mechanism that tries to monitor the environmental configurations (e.g. routing) that take place on lower levels of session initiation between two entities. Also, apart from IP-based networks, ANA by principle allows the incorporation of IP-independent networks (e.g. sensor networks) that can exhibit particular and idiosyncratic anomalies within their traffic behaviour. Under such a scheme, anomalies within a compartment are the aggregation of challenges triggered by several different environments. This fact leads to the assumption that statistical (e.g the distribution of flow interarrival times) properties that characterise normal behaviour and distinguish types of anomalies will capture different values than those observed in traditional backbone networks.

As an example, we can consider the case where a malicious, blackhole node that already has compromised the routing protocol of a large wireless sensor network (WSN) wants to exploit the initial protocol negotiations with an IP compartment. In such a scenario, the attacker would easily perform eavesdropping on the application-layer services that the particular compartment offers (since the services provided by each compartment are public to every node and compartment IDPs have no security mechanisms) and further "legally" request for a particular service in order to join the IP compartment. Subsequently the attacker node would be able to tamper the TCP negotiation (e.g. by using a fake RST or ACK flag) and create a blackhole route in its new environment. In parallel with the newly formed blackhole, the attacker node still behaves as a blackhole to its old WSN compartment and is in a position to trigger anomalies such as a (D)DoS on both compartments. Obviously, due to the WSN nature, a (D)DoS in such an environment is volume-wise comparatively much smaller rather than a (D)DoS in an IP environment. Nevertheless, such an action would lead to different pattern on the ANA traffic distributions since a monitoring entity under ANA is not concerned only with a single compartment but rather with all inter-connected compartments.

Having as a basis the aforementioned scenario as well as the hypothesis in respect with the different values we'll possibly observe on traffic distributions, we constructed some ANA-specific characteristics that our detection/classification mechanism should follow. We required an anomaly diagnosis framework that would exhibit common mathematical properties on both the detection and classification phase having at the same the capability to process a range of features from all the traditional layers. In parallel we considered as necessary that our component should be able to extract and further classify "lightweight" unobserved anomalies (e.g. the

WSN (D)DoS example). In addition, an important aspect is for our framework to be adaptive. This capability is not only determined by the algorithmic design but also from a pure engineering point of view. Our Detection Engine was formulated in a pluggable fashion where a diversity of various detection/classification methodologies may be plugged/removed based on the customised requirements of each compartment (i.e. based on which other types of compartments is connected with) in order for our engine to adapt at the environment-specific traffic (e.g. WSN traffic).

B. DE Internal Design

As shown in Fig. 3, the DE is internally composed by the Logic Brick, the Classifier, the Notifier brick, and a Configuration Manager (CM). The latter is in charge of handling all configuration settings in order for the DE to bind and configure itself with the interface provided by the adaptive measurement FB. In addition, it is the unit that allows dynamic binding and configuration of the RE with the DE as it will be described in the following subsection.

The Notifier brick (NB) is in charge of sending periodical updates to the RE. These are initially classifications passed to the NB by the Classifier who purely acts as the storage facility for the classified events. According to the nature of the already classified event, the NB instructs the RE to take either a local or a distributed remediation action. In case of no anomaly being reported, the NB still updates the RE within the time interval defined from the configuration made at the CM.

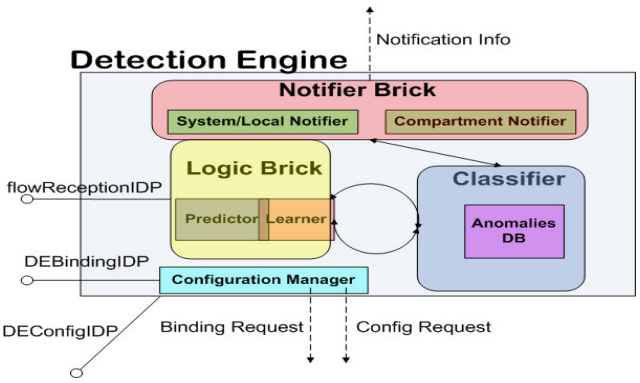

Figure 3:The DE high-level internal design.

The Logic Brick (LB) is the main control processing component in the DE. This is where the two most basic capabilities exist, *prediction* and *learning*. The prediction unit (Predictor) is in charge of applying all the prediction algorithms based on the runtime measurements taken by the adaptive measurements FB which are received on the dedicated measurement reception IDP [8].

On our current prototype, the Predictor acts according to the estimates given by the distribution of selected flow features (e.g. total number of packets of a complete bidirectional flow) and applies entropy estimation in order to characterise the evolution of selected fields in each flow received. We assume X corresponds to a finite number of selected flow-feature measurements $[x_1, x_2, ..., x_n]$ where each value possesses a probability $P = [p_1, p_2, ..., p_n]$. For each n, there is a corresponding uncertainty function $a(p)$ and therefore the function $A_n(p_1, p_2, ...p_n)$ states the average uncertainty for the range of all the set of finite random values in X. Based on axioms by Azcel and Daroczy [14] and Mathai and Rathie [15], we summarise and reform the relationships between the aforesaid functions, and as a result we attain the generalised formula that is also known as the Shannon's entropy:

$$H(x) = A_n(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i \qquad (1)$$

Using formula (1), we achieve an approximation of the evolutionary probabilistic behaviour that a selected flow feature would take. This technique will enable prediction of a possible anomaly within the network traffic on a local or compartment-wide scenario. In more detail, the prediction achieved is based on a simple comparison accommodated between the new entropy value and past pre-known entropy values observed in the particular compartment for the selected flow feature throughout time. As we explain in the following section (III), past entropy values are stored in a dedicated database present within the Predictor.

However, at this point we need to clarify that for the particular case of ANA, we need to go beyond single-layer features and exploit properties from some or even all of the traditional layers. Therefore, our current Predictor implementation accepts a range of features to be processed in a way to provide a summary of joint multivariate entropy results. For example, in a case where a wired Ethernet compartment interacts with a variation of other wireless and IP compartments, there is a possible selection of features ranging from the traditional data link layer (e.g Ethernet packets size distribution) to the network layer (e.g. TCP/UDP packets or bytes interarrival time) and application layer (e.g. SMTP byte size in case of mail-based attacks). This way, a deeper and at the same time broader view of the network statistics may be visualised and post-processed in the learning phase as employed by the Learner unit.

The Learner unit is in charge of performing run-time traffic classification based on the sample multivariate entropy results given by the Predictor and the already categorised past events stored in the Classifier. The Learner unit currently uses a supervised Naïve Bayes estimator which –being a probabilistic classifier– accepts a range of training data and classifies certain events at runtime. In our case, the training data are the conclusive entropy outcomes provided by the Predictor. Therefore, the already selected probability distributions of flow features constructed by the Predictor will be the input in

our Naïve Bayes classifier, in order to achieve a statistical model defining the different anomaly groups (i.e. classes). So, if we let X be a random vector with n variables:

$$X = [x_1, x_2, ..., x_n] \quad (2)$$

we can use it as input to the classifier and include it in the Bayes theorem that gives the following formula:

$$P(c_j \mid X) = \frac{P(c_j) f(X \mid c_j)}{\sum_{c_j} P(c_j) f(X \mid c_j)} \quad (3)$$

Formula (3) considers the discriminant functions given by any n variable in X as independent and that $P(c_j)$ is the probability of obtaining the anomaly class independently of the observed data given by X. The function $f(X \mid c_j)$ is a distribution function defining the probability of X given by c_j. The calculation of this formula is the main action taken by the Learner and provides a number denoting the Bayes likelihood ratio, therefore making it extremely feasible to minimise the cost or the probability error and resulting in accurate classification [11].

Obviously, since we employ a supervised scheme, complete runtime classification is achieved with a threshold-based comparison between a training data set and the likelihood estimation performed on the new data input. The training data set contained within the Learner holds likelihood values for the joint entropy (univariate or multivariate feature distributions) of past inputs given by the Predictor. Those likelihood values were initially grouped based on simply whether they belong to the normal or abnormal traffic spectrum and subsequently, those within the abnormal group are clustered into several sub-groups indicating the exact anomaly. In our current prototype, the training data set holds values for Flash Events and local as well as distributed Denial of Service (DoS) attacks. The values we embedded in our prototype were decided based on results obtained in [11] and [13].

IV. The Detection Engine Implementation

There are three main components encapsulating the functionality entailed in the DE. The Logic Brick (LB), the Notifier Brick (NB) and the Configuration Manager (CM). Fig. 4 shows a diagrammatical view of the data flow taking place within our implementation.

The Adaptive Measurement (AM) FB that is composed by 5 bricks has generic IDPs in charge of handling configuration and binding parameters. The AM IDPs are published as services within the local MINMEX and at the same time are advertised as a compartment-wide service. By exploiting this flexibility, it is therefore feasible for a remote DE to attach itself on a physically distant ANA *node*.

As bootstrapping stage, the CM sends binding parameters to the AM and at the same time transmits AM-compatible configuration parameters that will subsequently serve the measurement requirements coming from the DE. The binding parameters are simple messages denoting the compartment-related presence of the client DE to the AM (e.g. IP address).

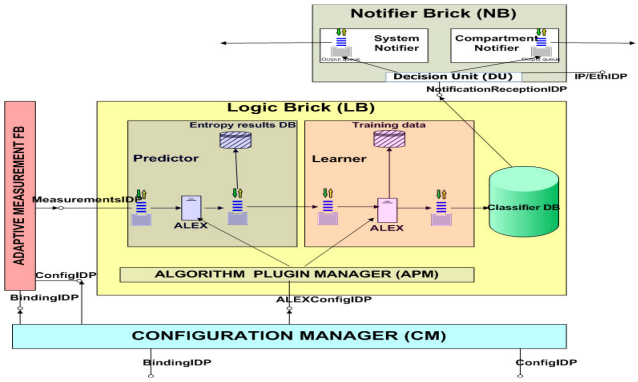

Figure 4: DE data flow & functionality.

On the other hand, the configuration parameters are series of filtering commands composed within a file requesting a customized operation of the bricks orchestrating the AM. An example might be the request by a DE to get measurements only for TCP flows from specific source/destination addresses within the local compartment.

At the same time, the CM accepts binding and configuration requests from a local or remote RE. Similarly to the binding done between the DE and the AM, a RE may bind itself by stating its position within the compartment. The configuration currently implemented allows a RE to determine a diagnosis notification time interval as received by the NB (which resides within the DE) and furthermore, to request from the DE to run preferred algorithms in a pluggable fashion. The algorithms to be plugged are meant to complement the standard algorithms we have implemented and described in the previous section.

Prediction and learning algorithms are initially plugged through the configuration done from an RE towards the CM, and are then placed correctly in the DE by the Algorithm Plugin Manager (APM). The APM is a management sub-component that resides within the LB and is in charge for initially cancelling the operations of the current running algorithms and for substituting them with the requested algorithm "plugins". These plugins are pre-compiled binaries and are not obliged to follow the ANA API since they are called as subroutines from the ALgorithm EXecutor (ALEX) units, which are placed within the Predictor and Learner components. However, a plugin request is not full if it is composed only by a pre-compiled binary. A request is considered to be fully compatible with the APM if it is accompanied with a "location" parameter denoting which unit (e.g. the Predictor or the Learner) to be attached to. Any wrong configuration parameters are discarded from the APM configuration IDP (i.e. ALEXConfigIDP). The Predictor

component holds a dedicated IDP (i.e. MeasurementIDP) that publishes itself on the local MINMEX and receives measurements according to the initial configuration agreed between the CM and the AM. A FIFO queue is on the reception of the Predictor and measurement data are then passed to the ALEX unit. Based upon the estimations done in the ALEX, they are then stored on a local database and subsequently sent to an output FIFO queue.

The Predictor's conclusive estimations are passed to a reception queue that resides within the Learner component which employs all learning algorithms (in our current prototype a Supervised Naïve Bayesian Classifier) via the operations done by its dedicated ALEX unit. Since the classification scheme within the Learner is supervised, a training set database is present in order for the ALEX to perform comparison with the real-time input coming from the reception queue. As soon as a coherent classification conclusion is structured, it is sent to the notification queue. The notification queue updates the Classifier DB within a dedicated time interval (currently 4 seconds). While the Classifier DB updates its data, it also sends a conclusive notification that includes the type of anomaly (e.g. local DoS) and the related traffic flow group to the Notifier Brick (NB).

As can be seen from Fig. 4, the NB's notification reception IDP is attached to a Decision Unit (DU) which is in charge of separating local from compartment-wide classified events. Every local threat is transmitted from the DU to the System Notifier (SN) which will then notify the local RE. Similarly, the Compartment Notifier (CN) is the unit that publishes notifications in case of a compartment-wide anomaly (e.g. a DDoS) where the local RE is required to interact with remote REs in order to mitigate such event. The NB holds one IDP that is mainly for handling data received from remote REs. Whether or not a network anomaly is manifested, the NB regularly updates the locally attached RE regarding the traffic status within an agreed time interval.

V. DISCUSSION & CONCLUSION

In this paper, we have presented the design and implementation of a traffic anomaly detection component that can be an integral part of next generation autonomic network infrastructures.

Via this work we have shown that the exploitation of carefully designed infrastructures, complex issues such as anomaly detection may be reasonably resolved in an affordable manner. Our diagnosis framework contributes to the composition of important autonomic capabilities such as adaptability, self-learning and self-protection, and also to self-optimization by being a component of the overall resilience architecture. In parallel, our framework prototype sets a core foundational contribution towards the realisation of an always-on measurement and control framework which to the best of our knowledge has not been seriously considered.

The results presented in [10][11] ensure that our algorithmic design satisfies the requirements of real-time anomaly diagnosis. Therefore, we have already started evaluating our engine's behaviour within the overall resilience architecture that we have also presented in this document. The overall architecture is in the process of being tested on the autonomic communications testbed provided within the ANA project (ANA-Lab). The ANA-Lab is a distributed ANA node infrastructure that provides the capability of virtual topology instrumentation through distributed monitoring and control facilities. Our target is to accommodate a series of experiments using (live as well as captured) operational traffic traces in order to examine the practical system performance of our architecture.

REFERENCES

[1]. Autonomic Network Architecture (ANA) Project details available at: http://www.ana-project.org

[2]. Barford P., Plonka D., "Characteristics of network traffic flow anomalies.", in Proceedings of ACM SIGCOMM, Internet Measurement Workshop, San Francisco, California, USA, November 2001.

[3]. Lakhina A., Crovella M., Diot C., "Diagnosing Network-wide traffic Anomalies.", in ACM SIGCOMM, Aug. 30- Sept 3, Oregon, Portland, USA, 2004

[4]. Lakhina A., Crovella M., Diot C., "Characterization of Network-wide Anomalies in Traffic Flows.", in ACM SIGCOMM Internet Measurement Conference (IMC), 2004

[5]. Sifalakis M., Louca A., Peluso L., Mauthe A., Zseby T., "A Functional Composition Framework for Autonomic Network Architectures ". In proceedings of 2nd IEEE International Workshop on Autonomic Communications and Network Management (IEEE NOMS/ACNM '08), Salvador, Bahia, Brazil, April 7-11, 2008.

[6]. Jelger C., Bouabene G., Schmid S., "Autonomic Network Architecture (ANA) Blueprint – Second Version", ANA, February 2009

[7]. Hutchison, D., Sterbenz, J. P.G, Jabbar, A. Sholler, M., 2006 D3.2: Resilience/Security Framework, Deliverable D3.2 ANA December 2006

[8]. Marnerides A. K., Pezaros D. P., Hutchison D., "Detection and Mitigation of Abnormal Traffic Behaviour in Autonomic Networked Environments"., 4th ACM CoNEXT Student Workshop, December 9-12, 2008, Madrid, Spain.

[9]. Bouabene G., Jelger C., Keller A., Rodriguez D., "ANA Core Documentation" Deliverable D.1.11, ANA, December 2008

[10]. Lahkina, A., Crovella, M., Diot, C., 2005, "Mining Anomalies Using Traffic Feature Distributions", ACM SIGCOMM 2005, Philadelphia, Pensylvania, USA.

[11]. Zuev, D., Moore, W., A., "Traffic Classification using a Statistical Approach", Intel Research Paper, 2005

[12]. Barford P., Kline J., Plonka D., Ron A., "A Signal Analysis for Network Traffic Anomalies.", in ACM IMW'02, Nov. 6-11, Marseille, France, 2002

[13]. Moore A. ,W., Zuev D., "Internet Traffic Classification Using Bayesian Analysis Techniques" in the Proceedings of the ACM SIGMETRICS Banff, Canada, June 2005.

[14]. Aczél J., Daróczy Z., "On measures of information and their characterizations." Mathematics in Science and Engineering, vol. 115, Academic Press, New York, San Francisco, London, 1975

[15]. Mathai A. M., Rathie P. N., "Basic Concepts in Information Theory and Statistics: Axiomatic Foundations and Applications", Wiley Halstead, New York, Wiley Eastern, New Delhi (1975).

[16]. Pezaros, D., P., Mathy, L., "Explicit Application-Network Cross-layer Optimisation", 4th International Telecommunication Networking WorkShop (IT-NEWS) on QoS in Multiservice IP Networks (QoS-IP 2008), Venice, Italy, February 13-15, 2008