ANA Project

Autonomic Network Architecture

Sixth Framework Programme

Priority FP6-2004-IST-4

Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D.3.10

Measurement-based Resilience Mechanisms

Version 1.0

ANA Project

Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP3/D3.10
Document Title	Measurement-based Resilience Mechanisms
Workpackage	WP3
Editor	Dimitrios Pezaros (ULanc)
Authors	Dimitrios Pezaros (ULanc) Angelos Marnerides (ULanc) David Hutchison (ULanc)
Reviewers	Guy Leduc
Dissemination level	Public
Contractual delivery date	31st January 2009
Delivery Date	30th January 2009
Version	1.0

Abstract:

This document presents the architecture and design of a distributed resilience system for the ANA platform.

Keywords:

Resilience, measurement, network compartment, ANA brick

Executive Summary

Work within the Resilience task of the ANA project has evolved from the conceptual definition of resilience as an emergent property of networked systems, to the derivation of mechanisms to enhance network resilience, and the design of a distributed resilience architecture to defend against abnormal network conditions and remediate their effects. Following on the foundations work on defining resilience as a collection of mechanisms to *Defend, Detect, Remediate* and *Recover* network anomalies, and at the same time to *Diagnose* and *Refine* normal operation, we have identified practical modular mechanisms to alleviate the effects of adverse operational conditions caused by either legitimate or malicious activity. For example, the treatment of a Denial of Service attack should be substantially different than the action required in response to a flashcrowd, in order to minimise network impact and sustain high levels of availability. At the same time, always-on measurement and monitoring of the levels of service quality delivered by the network requires additional and complementary mechanisms to ensure optimal operation and to provide for refinement and network self-optimisation.

This document describes the architecture and design of a distributed resilience system to operate on ANA networked nodes, and to provide the framework under which numerous algorithms can be implemented to facilitate system and network self-protection. We have currently focused on the real-time control components of the resilience architecture that implement reactive defence mechanisms (through detection and remediation) at the onset of adverse operational conditions. However, the same architectural and design principles are equally applicable to always-on (pro-active) monitoring and optimisation components.

We describe the main system components that detect and remediate the effects of abnormal network operation, and we identify their main functional blocks and interfaces. The system is distributed in that it is able to operate either at a local network node or via the explicit interactions between components residing at remote network/compartment locations. We give an overview of where the overall system or parts of it can reside within a network compartment, and we identify its interoperation with the core ANA architecture and other functional entities (bricks). Finally, we outline the candidate algorithms that will implement anomaly detection and remediation within the ANA resilience system.

Table of Contents

1 Introduction

The resilience framework defined within task 3.4 of the ANA project proposed a six-step, two-stage strategy to optimise networked systems performance under the inevitable presence of faults and adverse operational conditions [15]. Two distinct yet concentric modes of operation have been identified, a real-time control loop to maintain networks' functional integrity at the onset of legitimate or malicious challenges, and a background diagnosis and refinement plane to self-optimise network operation through the employment of always-on measurement and control mechanisms. The principal actions defined for the real-time control loop include the *Detection* of abnormal behaviour, and the subsequent *Defence*, *Remediation* and *Recovery* of its effects (D^2R^2). At the same time, always-on background plane assures the *Diagnosis* and *Refinement* (DR) of –possibly– suboptimal network operation, through measurement-based mechanisms.

Although in theory the resilience framework provides for multi-layer and multi-dimensional enhancements to the data, control, and management planes of a networking architecture, in practice the deployment of absolute resilience may neither possible nor practical. As, with any enhancement mechanism, the realisation of resilience systems will be always determined by finite resource trade-offs that will include among others processing, memory, bandwidth, power, latency and cost.

We have therefore demonstrated that practical resilience systems should be expected to address certain challenges and threats to normal and optimal network operation, as opposed to being purely architecture-driven and complete with respect to some abstract model. At the same, such systems should not be constrained by narrow deployments focusing on too specialised scenarios, and essentially replacing traditional mechanisms such as e.g. firewalls. Rather, they should adopt a more generic operational model and be able to promote broader emergent network properties such as self-optimisation, high infrastructure availability and autonomicity.

Along these lines, we have proposed a number of modular and extensible systems to deal with challenges of distinct nature without losing their generality, and enhance the overall resilience of autonomic networked systems. We have highlighted the distinction between systems operating on the real-time control loop and take explicit defence and recovery action by reacting to external stimuli triggered by the onset of abnormal network behaviour, and systems operating on an always-on manner that measure aspects of network delivered performance and intervene to alleviate suboptimal operation [12].

Systems operating under the first category should defend the resources of the infrastructure through the *detection* of anomalous behaviour, and restore (recover) normal operation by *remediating* their effects. To demonstrate generality, we have identified two classes of threats, namely (Distributed) Denial of Service attacks and flashcrowds caused by the sudden popularity of an object residing behind a particular network topology, which although they exhibit similar symptoms and consequences, their effective remediation requires the adoption of quite different strategies. Both types of network anomaly can be detected by trigonometric and statistical methods on the number of

requests destined to a particular system per unit of time. However, (D)DoS attacks can be remediated by employing traffic dropping and pushback mechanisms, whereas the legitimate nature of flashcrowd traffic needs to be handled using different principles. Legitimate requests for content, especially when coming from batch-mode applications (i.e. non-interactive), can be subject to increase –rather than stabilising or even decreasing– with time. Although the effects of (D)DoS attacks can be alleviated (even implicitly) targeting their sources, flashcrowds can be effectively mitigated by the rapid propagation of the content to geographically distant locations. As a flashcrowd remediation mechanism, we have therefore suggested load balancing by clustering flashcrowd requests based on the diverse physical paths of the response traffic, and have demonstrated application through increase and global topology stress decrease [13].

For systems operating in the background plane to continuously diagnose and refine network performance, we have proposed the use of in-band measurement instrumentation of the operational traffic to abstract network-internal behaviour over the direct implementation of a performance metric of interest [11]. This is a flexible and low-overhead mechanism to assess the level of service quality exhibited by traffic while it is routed between two points in the network [10]. At the same, it can be integrated with network control structures such as e.g. routing to provide for always-on measurement-based network optimisation over ranging temporal intervals [12].

1.1 Scope of the deliverable

In this deliverable we provide a compact description of the architecture and the design of a distributed resilience system to defend the infrastructural resources and recover normal operation (through detection and remediation) at the onset of network anomalies and threats. We have focused the discussion on system components that operate on the real-time control loop and protect the networked systems from explicit threats and attacks. However, the overall architecture is equally applicable to systems operating on an always-on manner, subject to the appropriate interfaces and functional blocks being defined. The purpose of this deliverable is to demonstrate how the resilience concepts and mechanisms previously defined within ANA Task 3.4 can be realised at a system and network implementation level.

An overall operational description of the real-time distributed resilience system is graphically illustrated in Figure 1. Components of the architecture can reside in different systems over a network (within or across ANA network compartments, depending on the scope) and take synergistic action in order to adequately apply a particular detection or most commonly remediation algorithm. Each instantiation of the ANA resilience system consists of two main functional blocks that undertake *detection* and *remediation*, respectively. These engines mainly provide an encapsulation framework for the different algorithms implemented. In the discussion, we have used DoS and flashcrowd as example scenarios that need to be detected and mitigated by the resilience system. As stated above, upon detection of a particular abnormal event, the system needs to take distinct and appropriate remediation action according to the cause of the anomaly. What is

important to emphasise here is that in most cases mitigation of an anomaly at a system-local level may not be sufficient. In the case of a (D)DoS, when the attack hits the local system, it may be too late to take any recovery action, because bandwidth and/or system resource starvation may have already taken place [9]. It is therefore crucial for a system to be able to notify in-network nodes that can in turn mitigate the event closer to its source and before causing e.g. excessive congestion at the last (network) mile.

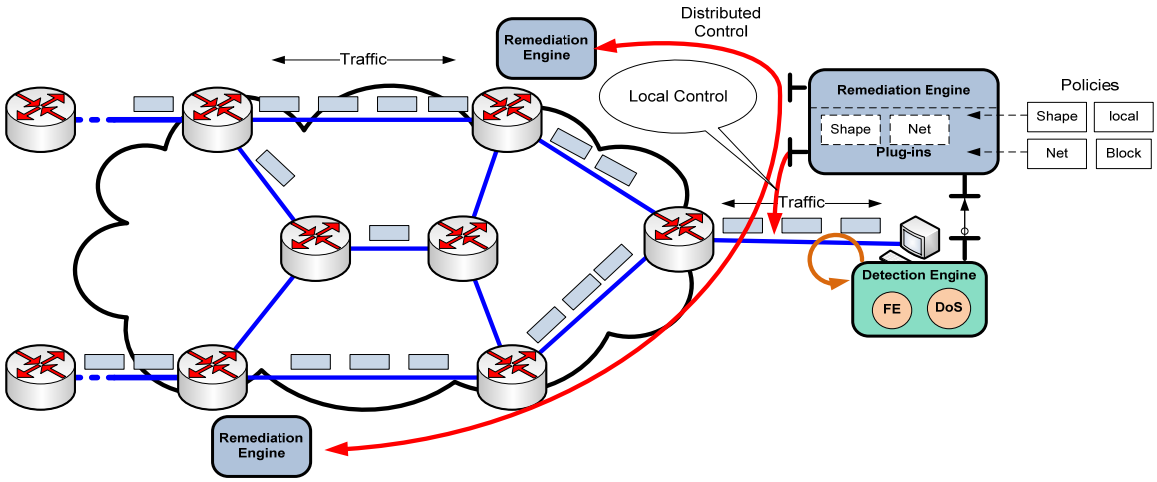

Figure 1: Detection and Distributed Remediation of Abnormal Network Behaviour

Likewise, in the case of a flashcrowd, the target system cannot do much other than selectively dropping requests and hoping that demand for the particular hosted object will cease with time. However, in-network nodes can provide insight regarding the topological (path) diversity of the response traffic, and therefore the ability to cluster requests in a way that traffic is distributed over multiple diverse paths.

The above examples illustrate the multi-dimensional operation of the remediation engine which, depending on the particular scenario, can assume *diverse* recovery actions with a *local* or an *in-network* scope.

1.2 Structure of this document

The remainder of this document is structured as follows. Section 2 presents the architectural design of the resilience system describing its overall interfaces and dependencies to other parts of the ANA architecture. It also describes the system's internal flow of information and control, and demonstrates its intra and inter-compartment operation. Section 3 describes the internal architecture of the Detection Engine (DE) subsystem and the specification of its main functional entities, and section 4 describes the corresponding aspects of the Remediation Engine (RE) subsystem. Finally, Section 5 concludes the document.

2 Resilience Architecture

The resilience architecture adopts a modular design that requires information from a monitoring facility. As showed in Figure 2, the Capturing Brick (CB) [3] sends information of monitored packets to the Detection Engine (DE) of the system where those are received on the designated Information Dispatch Point (IDP). After processing packet data and grouping flows at a desired level of granularity, the DE internally performs entropy estimation on selected packet features in order to detect an anomaly, and applies runtime classification via a supervised Naive Bayes estimator [14]. By these actions, the DE decides whether the observed flows are the result of a local or compartment-wide anomaly. As soon as a concrete decision regarding the precise nature of the anomaly is denoted, the DE composes a summary specification message that sends to its closest Remediation Engine (RE). In the 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 facing the event (e.g perform load balancing over multiple links to relax network congestion).

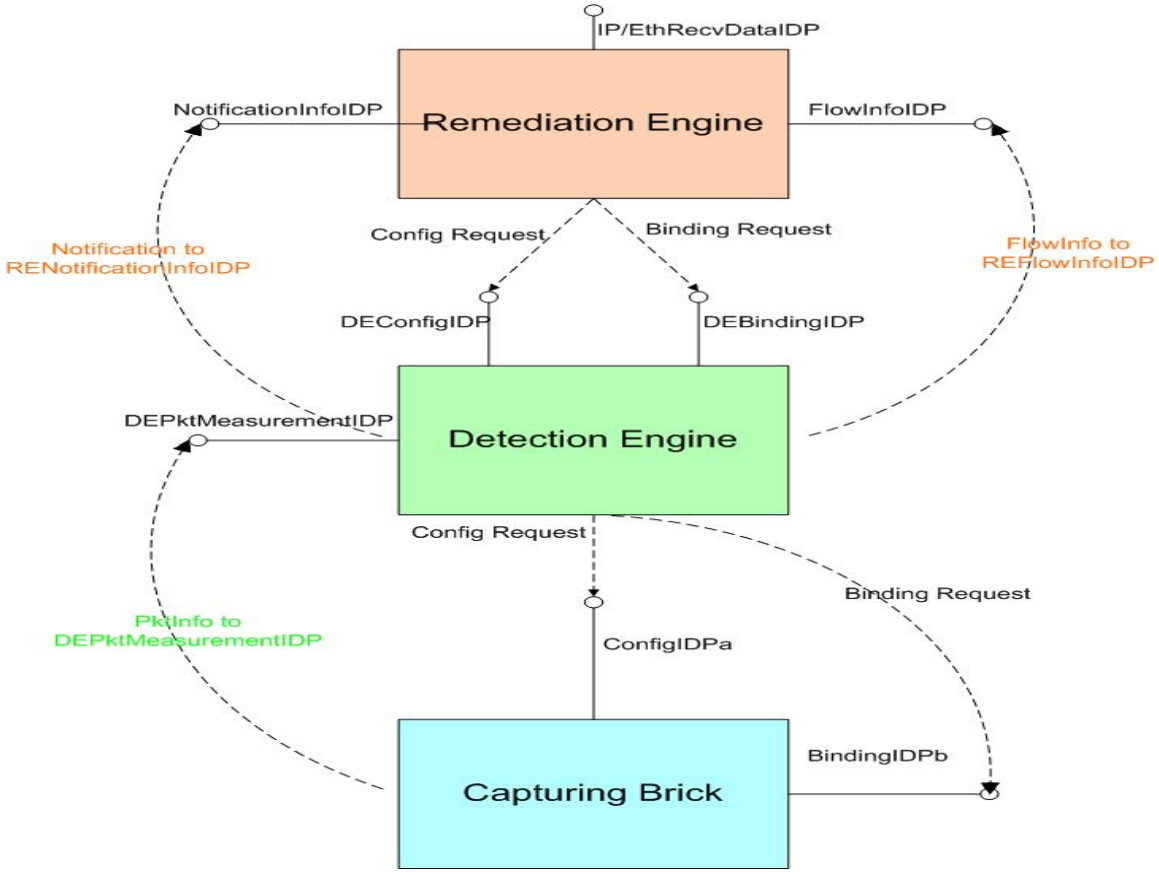

Figure 2: The Resilience Architecture

2.1 Local and Inter-Compartment Communication

2.1.1Resilience within an ANA node

Figure 3, shows how messages are exchanged within an ANA node in case all parts of the resilience system running locally. A scenario that can occur locally is a purpose-specific DoS attack that has an intention to cause a buffer overflow status on the OS, and the target system needs to detect the attack and alleviate its effects [9]. Through this figure we present from a high-level system overview how the DE, CB and RE interact via passing messages through the Minmex. The DE, CB and RE are bricks listed within the Minmex's brick table and therefore in order for each one to be aware of the others' existence (i.e. spot the public IDPs provided by each), it is required to initially communicate with the Minmex [2].

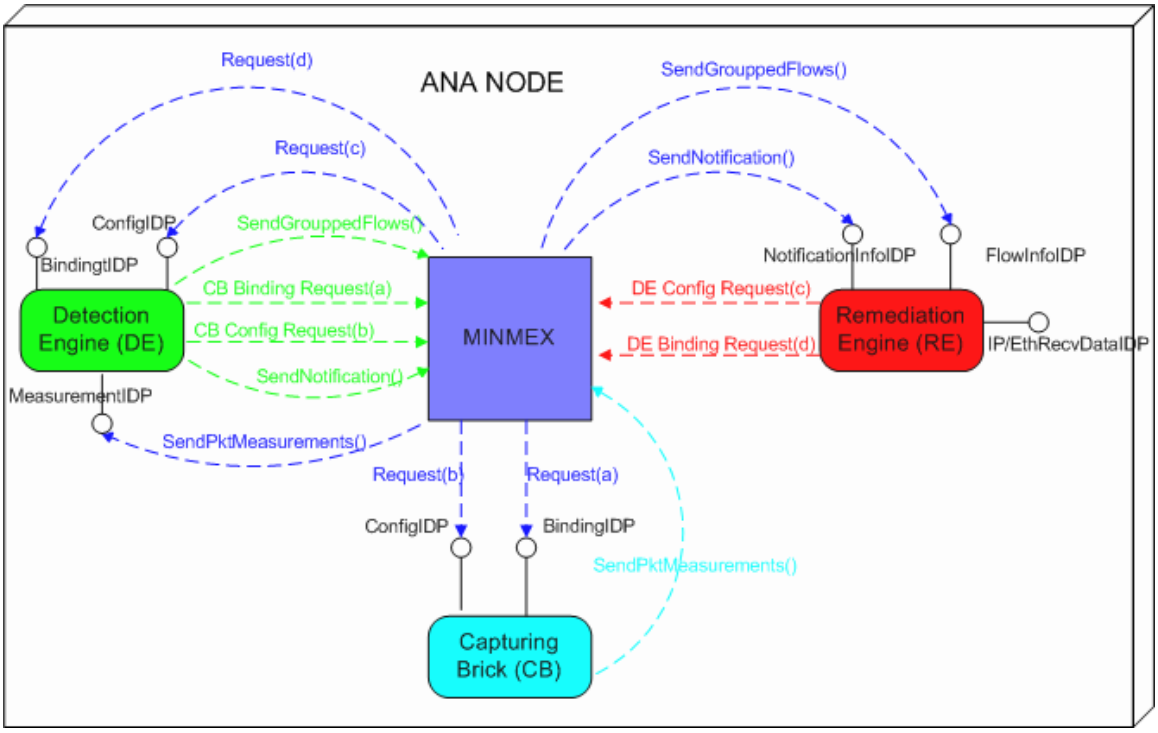

Figure 3: The Resilience Architecture within an ANA node

The DE sends binding and configuration requests to the CB via the Minmex and afterwards a binding under the correct configuration is established between the DE and the CB. At the same time and in a similar manner, the DE is bound and configured with the RE. A vital configuration setting between the DE and the RE that has to be

determined before binding is the *notification time interval*. The notification time interval is the time in which a DE needs to get information from the CB and further compose a conclusive profile regarding the exact traffic behaviour.

After the DE is bound and configured with the RE, measurements produced by the CB are passed through the Minmex to the DE. The DE performs the detection/classification procedure and constructs the notification update within the notification time interval. In the case of an anomaly to be dealt with locally, the DE sends the update to the RE and subsequently the RE initiates the appropriate remediation algorithms. For example, in case of a (D)DoS attack that needs to be alleviated at the local system (either an end host or a gateway), the RE will apply a dropping algorithm and possibly blacklist the sources involved.

2.1.2Resilience within an Inter-Compartment scenario

Apart from the case where a particular challenge is detected and action to remediate its effects is taken at a local system level, remediation can also be distributed across the network to facilitate early anomaly prevention and service restoration. In the case of a flashcrowd for example, load balancing based on physical path diversity can be the most effective remediation mechanism albeit it needs to take place in-network, as opposed to the target system locally. In such scenario, although early detection by the target system is crucial, knowledge of path diversity needs to be acquired by remote systems, such as the local or even neighbouring gateways.

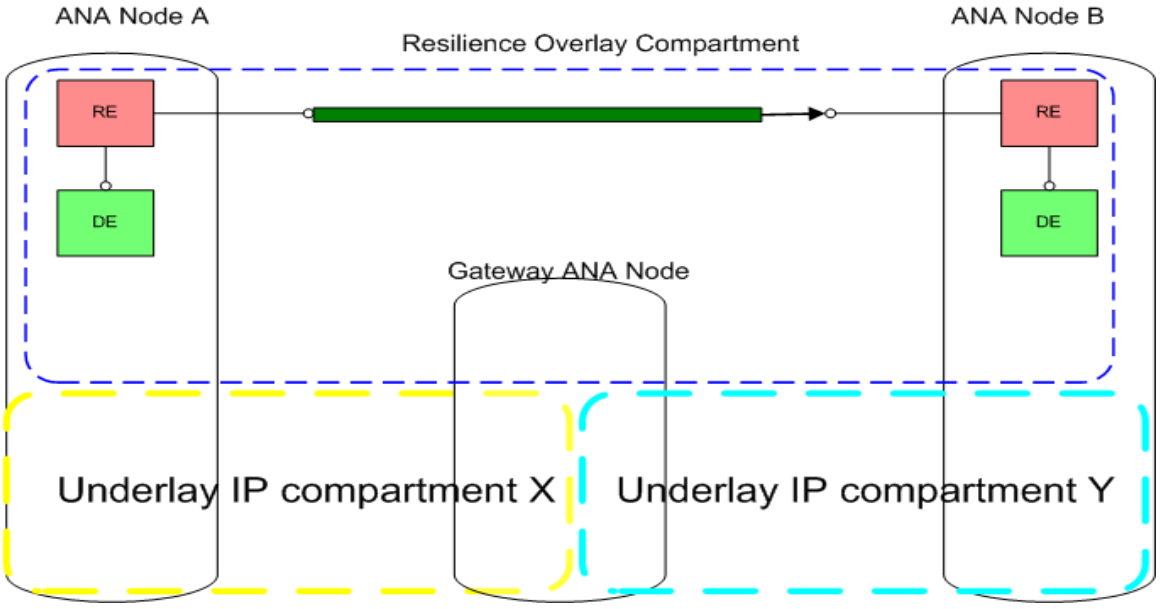

Figure 4: A high-level representation of the Resilience Architecture Inter-Compartment Communication

Assuming that two nodes residing on different IP segments (compartments) need to take synergistic detection/remediation action, a *Resilience Overlay Compartment* needs to be established, as shown in Figure 4.

Node A will need to be aware of the ANA node that acts as a gateway to its local compartment, which will in turn be in a position to initiate a further connection to node B in the adjacent IP compartment. Node B is then connected with node A via the Gateway ANA node and the REs from both nodes may exchange information related to the behaviour of the distributed defence mechanism to be employed. The information exchange between REs is performed within a dedicated Resilience Overlay Compartment as shown on a high-level in Figure 4.

3 Detection Engine (DE) internal architecture

The DE as shown in Figure 5 is composed by the Logic Brick, the Classifier the Notifier brick and a configuration manager.

The Predictor unit in the Logic Brick acts according to the estimates given by the distribution of selected packet features (e.g. checksum field) and applies entropy estimation in order to characterise the evolution of selected fields in each received flow. Entropy estimation provides a detailed and more accurate identification of events that may not be extracted in large traffic volumes [1].

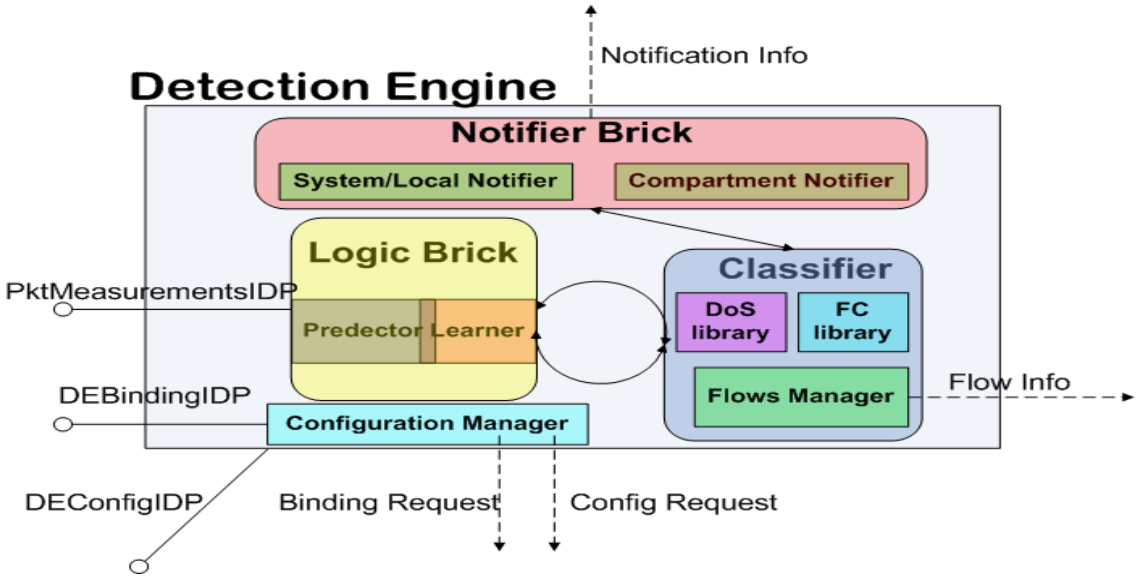

Figure 5: The Detection Engine internal architecture

Our design accommodates traffic prediction using Shannon's entropy as denoted in [4]. We assume that X corresponds to a finite number of selected flow-feature measurements $[x_1, x_2, ..., x_n]$ where each value possesses a probability denoted as $P = [p_1, p_2, ..., p_n]$. 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 [1] and Mathai and Rathie [8], we summarize and reform the relationships between the aforesaid functions and as a result we attain the generalized formula that is also known as the Shannon's entropy:

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

Using formula (1) we will 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 parallel with the operations made within the Predictor, the Learner unit performs run-time traffic classification based on the sample entropy results given by the Predictor and the already categorised past events stored in the Classifier. The Learner unit uses a supervised naive Bayes estimator which –being a probabilistic classifier– accepts a range of training data and classify certain events on runtime. In our case, the training data will be the conclusive entropy outcomes provided by the Predictor. Therefore, the already selected probabilistic distributions of flow features as constructed by the Predictor will compose 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:

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

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

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

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 our classifier and provides a number denoting the Bayes likelihood ratio, therefore making it extremely feasible to minimize the cost or the probability error and resulting in accurate classification [5].

We have selected this classification method because it has been reported to have low processing cost and high accuracy percentage [17]. Finally the internal architecture also has a Notifier unit which is in charge of updating the REs in case of a local or compartment-wide attack diagnosis.

3.1 DE specification

Bricks: Logic Brick, Notification Brick

Infrastructure Units: Configuration Manager, Classifier

Configuration Manager (CM)

The CM infrastructure unit is in charge of handling all configuration settings in order for the DE to bind and configure itself with the Capturing Brick [3]. Additionally, it is the unit that allows the binding and configuration of the Remediation Engine (RE) with the DE. It has two IDPs, one for managing the binding request coming from the RE (DEBindingIDP) and the other for handling the RE's configuration requests (DEConfigIDP). As soon as the RE binds with the DE, it is ready to accept any alarm or update triggered by the Notification Brick. The configuration property accepted by the DEConfigIDP is the time interval set by the RE for receiving system and network updates.

Logic Brick (LB)

The 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 capturing brick and which are received on the DEPktMeasurementsIDP. In order for the prediction algorithms to improve their accuracy and efficiency, there is a need for a dedicated unit to learn (Learner) from previous and current experiences, and to be able to understand a specific type of anomaly in a small temporal interval. This learning process is based on comparing information already stored in the Classifier (see below) from previous incidents with the attributes of the real-time traffic flow. The Predictor and Learner do not need to depend on each other since each one may have different algorithms for prediction and learning respectively.

Classifier

This unit acts purely as the anomalies' database by storing classified anomalies that took place in the past on a system or compartment-wise context. It communicates in real-time with the Logic Brick in order to facilitate real-time inexpensive comparison between known anomalies and the current traffic flow. Newly identified anomalies are stored in the classifier from the Logic Brick for further reference. Finally, this unit passes all the

flow information needed to the RE when requested, and it also sends the conclusion of traffic anomaly classification to the Notification brick.

Notification Brick (NB)

This brick is in charge of sending periodical updates to the RE. Updates given to the RE are initially classifications passed to the NB by the Classifier. According to the nature of the already classified event, it 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's DEConfigIDP. There are four notification info states: None, Low, Medium, High.

4 Remediation Engine (RE) internal architecture

Figure 6 shows the RE which is the component in charge of mitigating the effects of an anomaly (e.g. high system load, increased bandwidth consumption, congestion, etc.) we currently consider two families of remediation algorithms, namely traffic shaping and blocking (in response to an e.g. DDoS attack), and region-aware clustering (in response to a flashcrowd). The second category in particular that tries to perform load balancing and prioritise traffic to ensure path diversity and maximise throughput, empowers the property of self-optimization [13]. In addition, due to path diversity increasing in-network, this remediation algorithm is inherently distributed, where on RE instructs a peer take a clustering action on its behalf. To a lesser extent, the same holds for remediation based on datagram dropping since distribution among multiple REs can be exploited as an efficient pushback mechanism that enhances system and network self-protection and alleviates the detrimental effects of an attack at the "last mile" the overall this distributed operation of the RE attributes autonomic properties to the overall architecture.

The RE is composed by two main functional modules; the Defender and the Messenger. The former executes the system-local remediation algorithms whereas the latter distributes the instance of an event to remote REs within or across network compartments as required.

4.1 RE Specification

Bricks: Defender Brick, Messenger Brick

Infrastructure unit: Configuration Manager

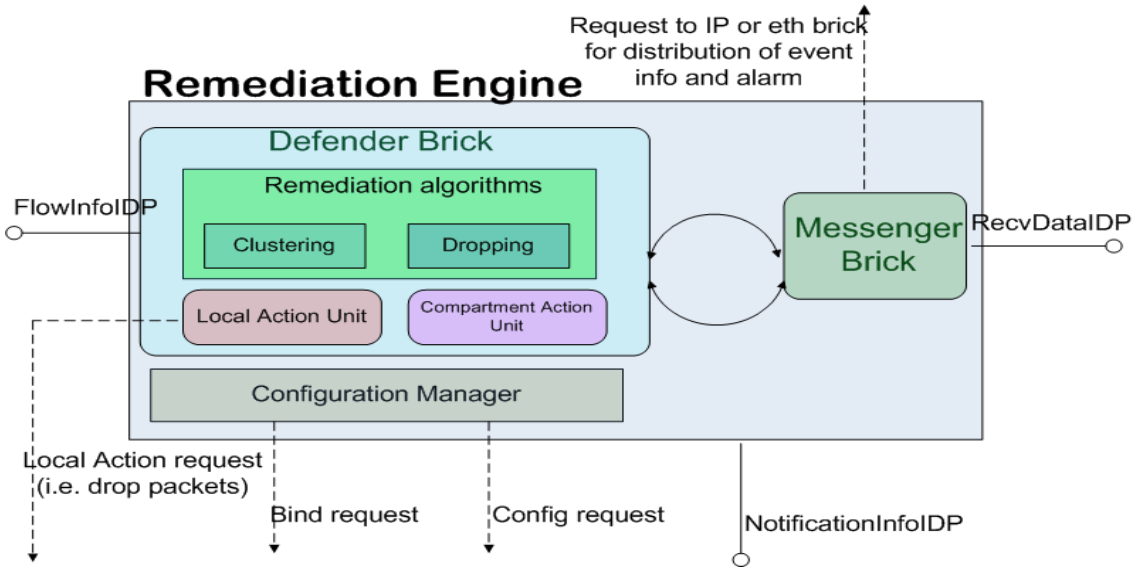

Figure 6: The Remediation Engine internal architecture

Configuration Manager (CM)

The CM is the infrastructure unit holding the responsibility for dynamic binding and configuration of the overall RE with a local or remote Detection Engine (DE). All outgoing requests are always passed through the Minmex.

Defender Brick (DB)

The DB is the main functional block within the RE architecture. It holds one IDP (FlowInfoIDP) responsible for getting all the grouped flows sent from the DE. The grouped flows will be the subjects of the remediation algorithm employed, since they have been identified by the DE to cause the abnormal behaviour. The remediation algorithms are enforced by two action units, depending on whether a system-local or distributed action is required. The *Local Action Unit (LAU)* is the main control entity enforcing local traffic control on the ANA node, such as for example, datagram dropping on the local link. If a distributed remediation action is required, such as for example grouping of flows based the compartment's egress links they are routed to, then the *Compartment Action Unit (CAU)* is responsible to form a control request that is going to be transmitted to a remote RE for execution. Based on the particular remediation scenario, one of the two or both action units can operate within the DB. In addition, the DB is the first element in the architecture that gets all the information passed on from the DE to the NotificationInfoIDP.

Messenger Brick (MB)

The MB is the component within the architecture in charge of the distributed interaction between remote REs in the Resilience Overlay Compartment (as discussed in

section 2.1.2). Interaction among REs may be committed within an intra or inter compartment scenario depending on whether the Resilience Overlay resides on top of the IP or the Ethernet Compartment. MB handles the message passing between REs by encapsulating requests formed by the CAU of the Defender Brick, and is also responsible for the exchange of control information between the REs that communicate the status of remote requests. The MB holds one IDP (RecvDataIDP) to receive remote data.

5 Conclusion

In this document we have presented the architectural design of a distributed resilience system for the ANA architecture. The system provides for defence and recovery from legitimate and/or malicious network abnormalities by *detecting* their onset and *remediating* their effects. It provides for distributed defence in case an anomaly cannot be handled locally, by exchanging requests for detection and most commonly remediation between remote networked systems. We have presented the overall system architecture, its main individual functional blocks and its interfaces, and we have also described how it operates over a single or multiple network compartments and how it interoperates with other components of the ANA architecture. Future work will focus on finalising the implementation of the ANA resilience system, and on incorporating a number of algorithms to practically demonstrate detection and remediation of diverse network anomalies over a range of scenarios.

References

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

[2]. Bouabene G., Jelger C., Keller A., ANA Core Documentation, Deliverable D1.8b, Autonomic Network Architecture (ANA) Project, FP6-IST-27489/WP1/D1.8b, December 2007

[3]. Cantin F., Goebel V., Gueye B., Kaafar D., Leduc G., Martin S., May M., Peluso L., Siekkinen M., Plgemann T., Zseby T., Roth R., The Monitoring Part of the ANA Architecture, Deliverable D3.3,Autonomic Network Architecture (ANA) Project, FP6-IST-27489/WP3/D3.3, December 2007

[4]. Claude E. S., A mathematical theory of communication. Bell System Technical Journal, 27:379–423 and 623–656, July and October 1948.

[5]. Fukunaga K., "Introduction to Statistical Pattern Recognition (2^{nd} edition)", Academic Press Professional, Inc. , San Diego, California, USA, ISBN 0-12-269851-7, 1990

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

[7]. Marnerides, A., Pezaros, D., P., Hutchison, D., Detection and Mitigation of Abnormal Traffic Behaviour in Autonomic Networked Environments, ACM CoNEXT 2008, Student Workshop, Madrid, Spain, December 9, 2008

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

[9]. Peng T., Leckie C., Ramamohanarao K., Survey of network-based mechanisms countering the DoS and DDoS problems, ACM Computer Surveys (CSUR), v.29 n.1, p. 3-es, 2007

[10]. Pezaros, D., P., Hoerdt, M., Hutchison, D., Overhead Assessment of In-band End-to-end Network Performance Measurement (in progress)

[11]. Pezaros, D., P., Hutchison, D., Garcia, F., J., Gardner, R., D., Sventek, J., S., In-line Service Measurements: An IPv6-based Framework for Traffic Evaluation and Network Operations, in Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS'04), Seoul, Korea, April 19-23, 2004

[12]. Pezaros, D., P., Marnerides, A., Hutchison, D., The Resilient Part of the ANA Architecture – Towards Resilient Systems, Deliverable D.3.6v1, Autonomic Network Architecture (ANA) Project, FP6-IST-27489/WP3/D3.6v1, February 2008

[13]. 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

[14]. Rish, I., An empirical study of the naive Bayes classifier, IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Seattle, USA, August 4, 2001

[15]. Sterbenz, J., P., G., Schöller, M., Jabbar, A., Hutchison, D., First Draft of the resilience and Security Framework, Deliverable D3.2, Autonomic Network Architecture (ANA) Project, FP6-IST-27489/WP3/D3.2, February 2007

[16]. Tschudin C., Jelger C., Bouabene G., Leduc G., Peluso L., Sifalakis M., Schoeller M.,May M., Siekkinen M., Roth R., Schmid S., Zsesby T., Goebel V., Plagemann T., ANA Blueprint, Deliverable D3.3,Autonomic Network Architecture (ANA) Project, FP6-IST-27489/WP0/D1.4/5/6_v1.1, February 2008

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