Critique of Network Management systems and their Practicality

Haydn Mearns, John Leaney
Faculty of Engineering and Information Technology
University of Technology, Sydney
Sydney, Australia
Email: haydn,johnl@it.uts.edu.au

Dominique Verchere
Alcatel-Lucent Bell Labs
Paris, France
Email: Dominique.Verchere@alcatel-lucent.com

***Abstract*—Networks have become an integral part of the computing landscape, forming a global interconnection of a staggering number of heterogeneous systems and services. Current research focuses on policy based management and autonomous systems and involves the utilisation of very different languages and technologies in concert. This paper examines four current proposals for autonomous network management and analyses them using architectural modelling, against a measure of practicality, as expressed by scalability, reliability and maintainability.**

***Keywords*-Autonomous; Network Management; Performance; Practicality**

I. Introduction

Telecommunications Network Management systems are of a class of systems which can be described as large and complex, with many interacting subsystems, considerable realtime requirements, and, extensive user interface and business interface requirements. This research focuses on network management systems as being large complex software systems, and therefore argues that there is benefit in using the techniques of design and analysis developed for large, complex systems. The theory, methods and techniques for large systems are usually grouped under the umbrella of software architecture. In particular, this research focuses on the practicality of the design of network management systems, the work that had been performed by others, to date, in this area, and, demonstrated some shortcomings in the software architecture techniques available. In recent years, autonomic, policy-based network management systems are seen as the best approach to the management of networks. This paper is the first in a series of analyses and recommendations based on current proposals for autonomic network management systems. This paper analyses four currently proposed autonomic management designs, with regard to the validity of their designs, and, with regard to practicality, as expressed by scalability(focussed on time performance), reliability and maintainability.

II. Related Work

Although there are many papers which propose autonomic NM systems, we found no papers which attempted to estimate the performance in any form. Even survey papers such as [1] make no mention of the issue of performance, quite apart from any attempt to predict the likely practicality of the proposals.

III. Critique of NM systems and their practicality

To understand the current state of autonomic Network Management, four models of current frameworks were analysed using software/systems architectural theory, processes and tool. The seven NM systems surveyed are FOCALE [2], Optimizing QoE [3], Pronto [4] ANEMA [5], AORTA [6], DNA framework [7] and DNSP maintenance management [8]. These seven were the only published autonomous telecommunications network management systems which had anything like sufficient detail to be used for any form of modelling of performance. Eventually, AORTA, DNA framework and DNSP maintenance management had to be discarded has having no too little information for architecture, and therefore performance, modelling. The remaining four architectures were chosen as the most complete frameworks, providing a solution to more than one aspect of autonomic management. The focus of this analysis was on the practicality of implementation of these AMNs with respect to time performance, scalability, reliability and maintainability. This analysis is done from ??our own point of view and reflect the questions derived from modelling and simulating the architectures. The frameworks that were reviewed were FOCALE [2], Optimizing QoE [3], Pronto [4] and ANEMA [5].

A. FOCALE

The Telecommunication systems and software group, based at the Waterford institute of Technology, has introduced the FOCALE (Foundation - Observation - Comparison - Action - Learn - Reason) autonomic architecture [2]. It describes the design as split into a hierarchical distributed design, with the base element being a AME (autonomic management element) which handles a managed resource, be it single device or network. This AME controls the functionality of the managed resource by marrying an Autonomic Manager (AM) with a Model Based Translation Layer (MBTL) which translates the vendor specific data and commands to the AM's vendor neutral commands. The AME is contained in a Autonomic Management Domain and Autonomic Management Environment with each layer providing context, discovery, security, policy and analysis services. For the practical implementation of the AME FOCALE utilises a combination of information models (DEN-ng), ontologies (OWL) and Domain specific languages to derive the context model which represents the current state of the network and services. In its own parlance it does this by dividing the Monitor, Analyse, Plan, Execute (MAPE) control loop, described by Kephart & Chess [9] into two, A maintenance control loop and a adjustment control loop.

However from a practical perspective, the prototype is still not complete, as it ignores the functionality of the Domain and Environment layers. It also does not contain any learning and reasoning elements of the AME and seems to ignore the functionality of the MBTL, utilising vendor specific commands in the policy decision point. Finally there is a question as to why it is necessary to specify all the information in the DEN-ng information model when it is immediately translated into specific Domain Service Languages and system ontologies and then translated again into a set of rules in the rete-oo based JBOSS rules engine.

B. Optimising QoE

The university of Ghent has designed its own autonomic network management architecture, with the specific goal of improving quality of experience in access networks for multimedia [3]. It utilises three planes, Knowledge, Action and Monitor to complete the autonomic control loop discussed by Kephart & chess [9]. The monitor plane retrieves monitoring information from the devices using various methods, from standard SNMP for routers and switches to the ANTMA algorithm for TCP connection monitoring see [10]. The Action plane translates correction information into device commands. It stores current and historical monitoring information and system information in a specifically purposed ontology [11] called the knowledge base. The Architecture describes four components of the knowledge plane, a problem detection component, a problem tracking component, a problem solving component and a learning and reasoning component, and introduces 2 potential versions of the problem solving component, a analytical reasoner and a neural network based reasoner. The paper showed that, in the authors opinion the neural network reasoner is the superior decision maker for user QoE. Described as future work is the implementation and coordination of several planes to make the architecture distributed.

Of concern in this architecture is that although the problem detection, problem tracking and learning and reasoning components are mentioned there is no particular implementation described, other than the inference that some of the detection work will be done in the knowledge base ontology and that the neural network could be trained in real time to provide the learning and reasoning components. This use of a neural network in the problem solving and partial learning and reasoning component, while giving the architecture a fast decision time, requires such a long training time for every new service that it seems unworkable in a large heterogeneous environment.

C. Pronto

Pronto, developed at the university of technology Sydney specifies a Policy based service definition language to describe services and the system model through service definitions[4]. Each service definition defines the resources, devices, services and event condition action policies involved in each service. The architecture utilises three main component types, the Policy based management sub-system, a generic policy engine which interprets service definitions and other low level policies, a series of Domain Experts which perform policy translation or refinement for particular high level abstract policies to low level concrete policies [12], and the virtual device, which controls the configuration of the associated network devices.

Unfortunately this architecture is the least developed with the implementation of the design been written entirely by the author. As such there is no specification of the generic policy engine for the management subsystem. Furthermore the Domain Experts are listed as being specifically written for their particular class which implies that any extension to that particular domain requires either a new Domain expert or a rewriting of the current domain expert. Further in the simulation results the functionality of the Domain Experts is ignored, despite the large role they play in the functionality of the system.

D. ANEMA

Autonomic Network Management Architecture (ANEMA) proposed by [5] describes an architecture which utilises high level strategies to define goal policies to configure network elements. The high level strategies are implemented by the Objective Definition Point (ODP) component using expert defined analytical optimisation models that express the network functionality in terms of Network utility functions (NUF). The NUF's are forwarded to the Goal definition point (GDP) component which selects appropriate management strategies, specifically the configuration and optimisation strategies with which to optimise the NUF. These strategies represent goal policies that are defined as an aggregation of management strategies which are needed to achieve one or more quality metrics related to the NUF. These goal policies are distributed to the Distributed goal definition points (DGDP) and analysed to identify expert given behavioural policies and rules that can be distributed to the base level of the architecture, the Autonomic Management Element. The design of this AME is based on IBM's MAPE element.

Practically there is an issue with this architecture in the specification of high level optimisation models. To specify these analytically means that a level of expertise is required for any new service or functionality resulting in slow provisioning, and potential difficulties if the expertise is not available.

IV. Initial Results

A. *Overall Description and context of Network Architectures studied*

Looking at the network management systems from an architectural perspective there is similarity in all four network management architectures and some clear architectural structures. The maintenance control loop described by all four architectures and taken from the IBM vision for autonomic communications can be seen as as component and connection pattern, Jennings et al. [2] specifically states that FOCALE implements a adjustment control loop. The control loop would be a component and connection pattern for high level policy transformation. Derbel et al. [5] with ANEMA and Sheridan-smith with PRONTO [4] while not stating this specifically describe methods for online Policy adjustment which fulfil the adjustment control loop pattern. ANEMA states Policies which can be passed down through the Objectives layer, to the goal layer, while PRONTO utilises the PBM subsystem and domain experts to translate policies at different levels. Likewise, FOCALE, PRONTO and ANEMA all determine levels of policy characterised by a version of Strassners Policy continuum [13], which utilises the clear architectural concept of views.

1) Table of Comparison: The table I gives comparison of the schemas of the the network management systems. In the table, NM refers to to the name of the system, Architecture describes how much of the network is managed, Class describes whether or not it is autonomic or only automatic, Implementation is the proposed technology for implementation, Services describes the type of services managed, QoE describes the extent of the user quality being sought.

The table I gives a summary of the common structure, implementation and purpose of the studied architectures. As can be seen from the table three out of four frameworks have stated that the destined role of the architecture is the complete management of the Network, where as the Optimising QoE architecture is just focused on the edge. All four architectures are also deemed automatic, rather than autonomic as there is little discussion in the literature on these architectures of their ability to perform the stabilising aspect of autonomic behaviour. The difference in implementation, whether by centralised hierarchial distribution or true distribution through agents is highlighted, and will be discussed later in the paper. Again three of the four architectures specify their service offering as a more general network device configuration, with only Optimising QoE focussing on the multimedia delivery. However this focus on multimedia gives Optimising QoE a greater focus on QoE for its goal states, rather than the QoS focus of the other three architectures.

B. *Architectural Diagrams and Descriptions*

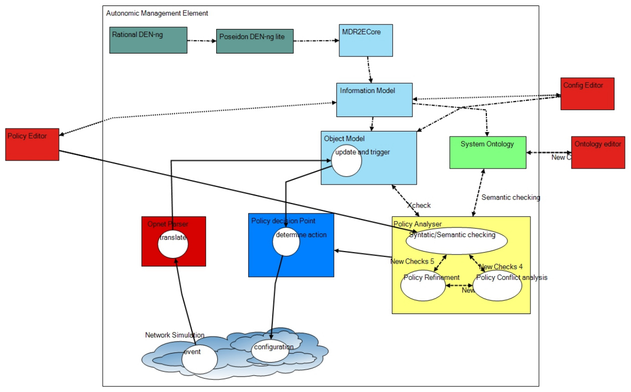


Figure 1. Focale AME Prototype model

1) FOCALE: The simulation of FOCALE includes only the base elements in the FOCALE architecture, that of the Autonomic Network Element. This reflects our understanding of the current state of development of the FOCALE architecture, and its current prototype developed by [14]. Figure IV-B1 shows the prototype which utilises the DEN-ng information model to build the object model, through the creation of a configuration DSL and a Event Condition Action Policy DSL. It also used the DEN-ng information model to create the system ontology. The object model as well as the system ontology is then used by the policy analyser to create a set of policy rules that can be interpreted by the policy decision point (PDP).

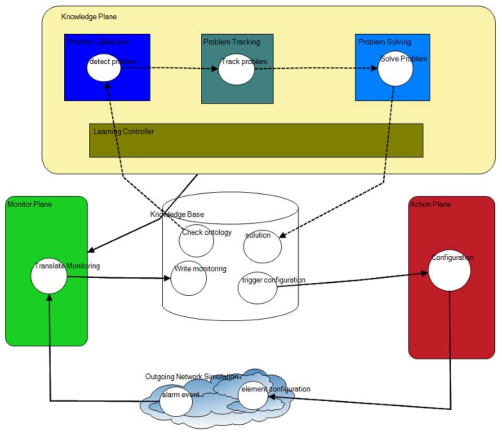


Figure 2. Optimising QoE CAANL model

Table I
CLASSIFICATION SCHEMAS FOR NM SYSTEMS

NM	Architecture	Class	Implementation	Services	QoE/QoS
FOCALE	Whole network	Automatic	agents	network configuration	QoS
Optimising QoE	Edge only	Automatic	centralised / agents	multimedia	QoS/ QoE
ANEMA	Whole Network	Automatic	agents	Network Configuration	QoS
PRONTO	Whole Network	Automatic	centralised	Policy configuration	QoS

2) Optimising QoE: The simulation of this architecture involves one instance of the three planes called, the central autonomic access network layer (CAANL). Figure IV-B2 shows the structure of the CAANL as described above. The model describes a maintenance control loop where actions are taken to maximise the QoE for the end user. The path that this maintenance loop describes starts with the collating, summarising and transformation of monitoring information to store in the ontological knowledge base. In the simulation described by Latré et al. [3] there is no definition of the problem detection and problem tracking components, therefore the model simply describes the neural network based problem solving component which provides the problem solution to the knowledge base. This solution triggers the action plane which configures the individual devices in the network.

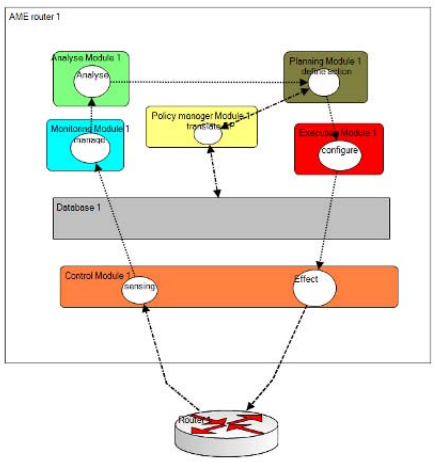


Figure 3. ANEMA AME model

3) ANEMA: The model of this architecture shows the base layer of the design, the objective achievement layer. This requires modelling the architecture of the Autonomic Management Element (AME) as each AME has the capability to make its own decisions utilising the monitoring analysing planing and executing capabilities with regards to meeting the target requirements of the goal policies and NUF optimisation models. As described in the literature by derbel et al. [5] Figure 3 describes the control module, with its sensors and effectors which read and control the network device. The monitoring module performs the monitoring functionality of the AME while the Analyse module analyses this monitoring information to detect change events. Once an event is detected the information is sent to the Planning module which in coordination with the Policy Manager module defines the set of elementary actions to respond to the event. These actions are sent to the execution module which controls the actions that are defined by the planning module. The coordination of theses modules defines a maintenance control loop, while the Policy manager module defines a step in the adjustment control loop as it translates the higher policies into router executable configuration commands using the policy descriptions and router capabilities that are stored in the database.

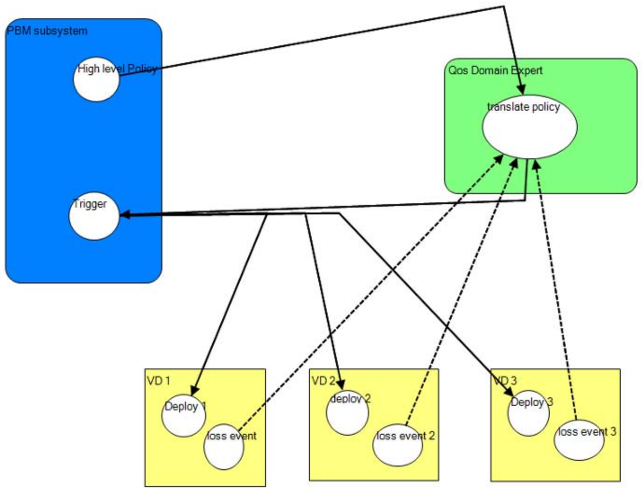


Figure 4. Pronto Model

4) PRONTO: The model of the PRONTO architecture is a simple implementation of the framework. Utilising the simple example of ensuring levels of QoS for network simulation, it utilises one Domain Expert for QoS and three virtual devices, representing three elements of the core. Figure 4 shows the adjustment control loop as it moves from the detection event in the virtual device, to the QoS Domain expert where new policies will be defined in reaction to that event. The Domain expert will pass on the policies to the PBM subsystem, which in turn executes them on the virtual devices affected. For the adjustment control loop, high level policies are defined and sent to the PBM subsystem which, forwards them to the required Domain expert, in this case the QoS DE, which translates the high level policies to low level policies with specific QoS information. these low level policies are pushed back to the PBM to be distributed to the virtual devices.

C. *Time Performance*

To simulate the time performance of all four frameworks, the architectural modelling tool ABACUS [15] was used. The time performance simulation relied on specification of the architectural components described in the literature. This specification was incomplete to varying degrees for all four frameworks. To make up this lack, some assumptions and inferences about general performance parameters of the particular components were made.

The Framework for Optimising QoE was the most complete with performance results for reading and writing the ontology to the knowledge base specified in the literature. However there was no specification of the performance of the Neural Network. A search turned up some results of the Levenberg-Marquardt Neural network performing classifications on problem sets from the machine learning and intelligent systems at the University of California, see [16]. The article demonstrates that the neural networks performance is most likely constant or O(1) [1] for data sets of varying sizes. However there was no performance characteristics of the Monitor or Action Planes.

Since the configuration time of the Action plane is dependant on the speed of the devices configuration and that configuration speed would be different for heterogeneous devices it can be assumed that the performance of the architecture is not dependant on the performance of the Action Plane. Likewise the monitoring planes receiving, summarising and processing of monitoring data is dependant on the type of data being sent, which would also be varied, dependant as it is on the type of device sending the monitoring data. Therefore the general performance of the architecture is dependant on the individual performance of the knowledge base and the knowledge plane. This paper [11] describes the results of a performance test of the knowledge base's ontology, specifically its speed at writing and reading data. The article showed an average read value of 2.3 ms which is interpreted as the processing time of the check ontology and trigger configuration processes of the knowledge base. The processing time for the write monitoring and solution processes is the average write time of 5.7ms. this test was performed on a single core 2.2 GHz machine which gives us the knowledge bases processing speed and number of processors. Since there was no information on the performance of this implementation of the neural network algorithm in the literature of the architecture some performance statistics had to be assumed based on alternative resources. The article on the performance of the neural network algorithm [16] described above gives an average response time of 35ms for all sized classification problems set and is therefore taken as an estimate for the processing time of the solve problem process. The processing speed of 1.8 GHz of the test equipment in the article was also used by the component.

[1]big O notation describes the limiting behaviour of a function. Provides an upper bound on the growth rate of the function

To model the responsiveness of FOCALE with regards to time, the performance of the maintenance control loop needs to be simulated. To simulate the maintenance control loop the network data is sent to the parser which updates the object model. The update to the Object model triggers an evaluation in the Jboss rules engine (the PDP), which determines through the satisfaction of the conditions of one or more policies, whether or not actions need to be taken [2]. For simulation it can be seen that the timing of this loop is entirely dependant on the performance of the JBoss rules engine, as the translation of the parser can assumed to be a constant, and the update of the object model always results in triggering the re-evaluation of the policies. Unfortunately there was quite a lot of difficulty in determining a generic timing for the JBoss rules engine, only one web based article [17] showed concrete decision making response time for a particular rule set from a comparison study done with the Microsoft rules engine and the DROOLS rules engine, using no particular optimisations in the rules engines. However utilising this article shows that there is a linear or O(n) response time for increased rule sets. For the object model as well, there were difficulties in deriving a definitive response time and the timing of O(n^2) has been assumed based upon the complexity and time performance of similar object model systems.

Performance numbers were not specified in the published literature of FOCALE. As with Optimising QoE some reasoning on the relevance of some components and some assumptions based on the general characteristics of the identified components needed to be made. The Opnet parser component, for example, is specifically built to translate OPNET xml data to the Domain Specific Language XML of the object model. However since in a practical application, the performance of this translation would be dependant on the type of translation needed for specific devices, which vary widely, it can be seen to be less relevant than the performance on the Object Model and the Policy Decision Point. As for the assumptions made, the basis of the performance of the Policy Decision Point component is taken from a performance comparison of the DROOLS rule engine, around which JBOSS is built. This comparison [17] specifies a response time for the DROOLS engine when working with a certain number of rules in the set. To be conservative the number of rules chosen was 1000 which gives a processing time of 62 ms. The processing speed of the component is also taken from the comparison articles test bed with a single processor running at 2.6 GHz. Furthermore since the model of FOCALE is focusing on the components of one element of the architecture, the AME, this same processing speed can be applied to the Object Model. For the processing time of the update and trigger process of the Object Model, a variable figure based on the scalability scenario described below and the assumed O(n^2) response time.

The PRONTO architectures responsiveness to time is dependant on the the runtime execution of the the management nodes. The literature gives a simulation of the performance of these management nodes with regards to simple deployment policies. However, since the Domain Expert is coded separately, it is difficult to judge its responsiveness. However future work will endeavour to compensate for this difficulty and produce some time results.

Unfortunately there is a lack of literature on the ANEMA architecture. The result of which is that while the AME of the architecture is described and is based on the Kephart and Chess MAPE model. There is no current information on the components implementation which makes simulation difficult, however since the MAPE model is a well known, future work will involve an estimation of its performance from other discovered implementations.

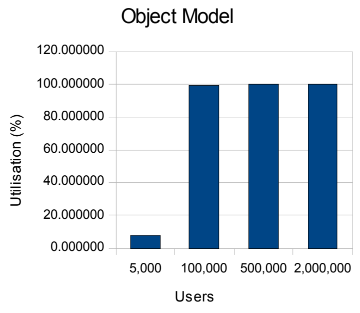
D. Results

1) Scalability: In testing the two architectures, FOCALE and Optimising QoE, with regards to scalability, we looked at the Utilisation of the components whose performance most affected the adjustment control loop as well as the connection response time. In FOCALE there were two components of the AME that warranted attention, the JBoss rules engine and the object model which it is dependant upon. In Optimising QoE, the two components that are relevant are the neural network and the knowledge base. To simulate an increase in users for the architectures a crude scenario was devised which estimated the number of network devices needed for service delivery to 5,000 to 2,000,000 users, shown in the table II, and increased the message rate to the components with respect to the device numbers. The number of devices was estimated from experience of a telecommunications network in Australia, given common capacities for routers and DSLAM's.

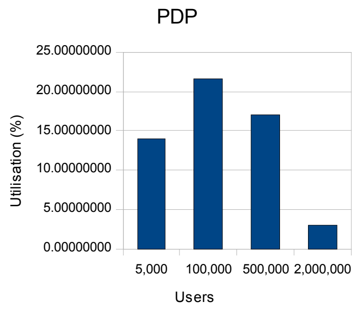
The utilisation of the components was then calculated, as well as the common connection response time. The calculation was done with respect to the increased message rate, the components processing speed as well as each components process processing time, which is described above in section 4.3. Utilisation was chosen for these calculations as it is not the absolute values that are important in this initial stage, rather it is the shape of the curve that matters. Some better absolute dependent variable will be determined before the next paper.

For the scalability of FOCALE Figure 6(a) Shows the utilisation of the object model assuming that the processing time of the object model is as stated in section 4.3. Figure

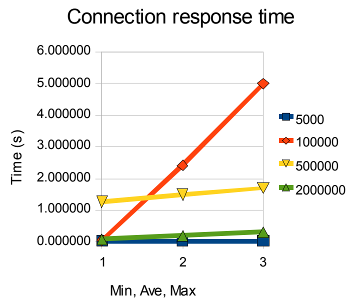
Figure 5. FOCALE scalability



(a) Object Model Utilisation



(b) PDP Utilisation



(c) Response time

6(b) shows the utilisation of the Policy Decision Point, while Figure 6(c) shows the connection response time between these two components. As can be seen in the graphs, when the utilisation of the Object model is overwhelmed by the message rate, it degrades the performance of the connection and the PDP.

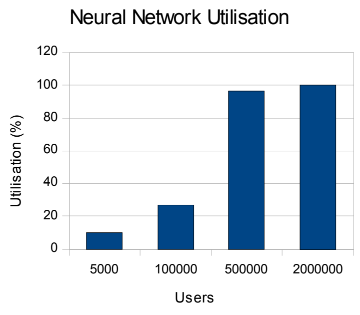
Figures 7(a), 7(b), 7(c) represents the utilisation of the Neural Network, and the Knowledge Base components of the Optimising QoE Architecture, as well as the response time of the linking connection. As can be seen in the graphs the performance of the knowledge base reflects a linear response to the input message rates, while the neural networks performance can be interpreted to have a potential upper limit to its performance.

A couple of observations can be inferred from the results of the scalability simulations. First for FOCALE, as the

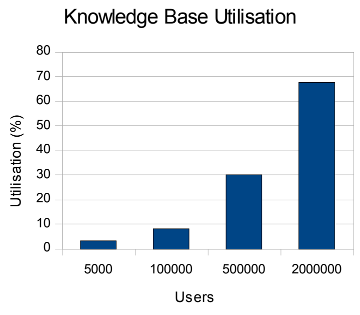
Table II
No of Devices to Users

residential	servers	switches	core routers	outer routers	edge routers	DSLAMs
2,000,000	400	40	6	100	69	6,944
500,000	100	10	6	25	17	1,736
100,000	20	2	6	5	3	347
5,000	1	0	6	0	0	17

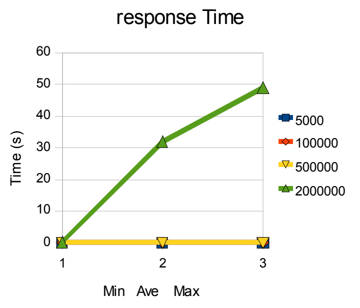
Figure 6. Optimising QoE scalability



(a) Neural Network Utilisation



(b) Knowledge Base Utilisation



(c) Response time

performance of the Object model degrades with the size of the network it models, and further degrades the performance of the PDP it would be impractical to implement complete Object models in each AME. Therefore there must be some form of distribution or delegation of the object model in each AME. Secondly for Optimising QoE, the results indicate that there will be some limit on the ability of the neural network to deal with increased message rates, considering the design calls for a centralised layer to deal with problems affecting the whole network, it introduces the possibility of a ceiling on the amount of devices this architecture can deal with.

2) *Resilience:*

Reliability: While there has not been any simulation done yet on the reliability of the four architectures, some analysis of the chosen components can be made. One of the most basic observations is in the Optimising QoE architecture, which uses a neural network for its problem solving, The neural network can be considered an unreliable component as a certain percentage of the decisions made will be in error. Since the system will implement these erroneous decisions, this will introduce instability into the system.

Redundancy: None of the Network management systems prototypes deal with the issue of redundancy, other to indicate that the use of separate AME's, for FOCALE, ANEMA, and Optimising QoE, and the use of separate management nodes, for PRONTO, will limit the problems to the devices managed by whatever particular AME or Node that is affected.

3) Maintainability: Considering the purpose of these architectures is to be able to provide systems of self management, a focus of the maintainability becomes a question of adaptive maintenance, which is the difficulty in integration of new devices or services. There are issues with the provisioning of new devices services in ANEMA, PRONTO and Optimising QoE. In ANEMA for example, any new service which needs to be provisioned requires an expert to build the high level optimisation function, a task that would be difficult for business managers or even general network operators. PRONTO on the other hand can define new services simply through its Policy language however requires, for any new technology that is added, a Domain Expert to be coded and integrated with the existing architecture, certainly a non trivial task. For Optimising QoE the issues lie in the long process of training the Neural network which would need to be done for all new services added. FOCALE with its use of the growing DEN-ng information model is the only system which offers a chance at integration that is relatively simple, However this would rely on the level of coverage of the information model which is currently unknown.

E. Centralised Optimisation vs Distributed Optimisation

The specification of the distribution of work is somewhat incomplete for the four architectures studied, being mostly described as future work. For Optimising QoE, the described architecture requires that information is passed to the Central autonomous access network layer which has the authority to solve global problems and coordinate actions amongst its distributed layers. This indicates a hierarchical distributed design with centralised control. This form of centralised optimisation is also indicated in the PRONTO design with the PBM subsystem being responsible for overall policies and splitting and delegating individual policies to lower management nodes. FOCALE as well describes a centralised optimisation with the Autonomic Management Domain and Environment Layer possessing both a analysis and policy service. On the other hand ANEMA, utilises a more distributed optimisation, for while it describes a centralised goal layer and distributed goal points for the distribution of policies, the base AME is completely autonomous in its execution of these policies.

V. Summary & Conclusions

This paper has covered the analysis and modelling of four policy based or autonomic network management systems. A overview of each system was described from the point of view of architectural research, showing the architectural similarities in the network management systems that have been derived from work done in policy based and autonomous network management, as well as a critique of their practicality. Simulations were performed on models of the prototype implementations with regards to time performance and scalability while gross observations were made about the management systems resilience and maintainability. The results of the critique, observations and simulations showed that these systems are still in an early stage of development and are currently impractical for use. In light of this development a few opportunities have risen for improvement. For example the initial results of the scalability simulation showed that the system must be scalable, highlighting the idea that the distribution and coordination of policies is every bit as important as the translation of those policies. Another opportunity is the issue of stability in autonomous systems, not just as a part of policy conflict resolution but rather as an active part of the maintenance control loop.

Acknowledgement

This work was partly funded by Alcatel-Lucent Bell Labs and a research grant from the Australian Research Council, The authors would like to thank the respective bodies, which have provided a scholarship for the first author and support for the project. The authors would also like to thank the UTS Centre for Quantum Computation and Intelligent Systems (QCIS)for its support and Dr Mark Denford and Dr Tim O'Neill (Avolution ^{PtyLtd}) for their support with the modelling presented in this paper.

References

[1] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Trans. Auton. Adapt. Syst.*, vol. 1, no. 2, pp. 223–259, 2006.

[2] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. Foghlu, W. Donnelly, and J. Strassner, "Towards autonomic management of communications networks," *Communications Magazine, IEEE*, vol. 45, no. 10, pp. 112–121, October 2007.

[3] S. Latré, P. Simoens, B. D. Vleeschauwer, W. V. de Meerssche, F. D. Turck, B. Dhoedt, P. Demeester, S. V. den Berghe, and E. G. de Lumley, "An autonomic architecture for optimizing qoe in multimedia access networks," *Computer Networks*, vol. 53, no. 10, pp. 1587 – 1602, 2009, autonomic and Self-Organising Systems. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4V0TD93-1/2/628c1ff0c33c804ceb963c225ee1c1fd

[4] N. Sheridan-Smith, T. O'Neill, J. Leaney, and M. Hunter, "A policy-based service definition language for service management," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, April 2006, pp. 282–293.

[5] H. Derbel, N. Agoulmine, and M. Salan, "Anema: Autonomic network management architecture to support self-configuration and self-optimization in ip networks," *Computer Networks*, vol. 53, no. 3, pp. 418 – 430, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4TW14YJ-3/2/41147806c839b89928697fc9b724d880

[6] A. Tizghadam and A. Leon-Garcia, "Aorta: Autonomic network control and management system," in *INFOCOM Workshops 2008, IEEE*, April 2008, pp. 1–4.

[7] L. Jun, Z. Shunyi, Z. Zailong, and W. Pan, "A novel network management architecture for self-organizing network," in *Networking, Architecture, and Storage, 2007. NAS 2007. International Conference on*, July 2007, pp. 146–154.

[8] J. F. G. Fernndez and A. C. Márquez, "Framework for implementation of maintenance management in distribution network service providers," *Reliability Engineering System Safety*, vol. 94, no. 10, pp. 1639 – 1649, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V4T-4W3HX4R-3/2/5d35a620a2149a5a95b3d297b56cd97b

[9] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan 2003.

[10] B. D. Vleeschauwer, W. V. de Meerssche, P. Simoens, B. Dhoedt, P. Demeester, T. V. Caenegem, H. Dequeker, K. Struyve, E. Gilon, and E. Six, "Enabling autonomic access network qoe management through tcp connection monitoring." in *ACNM2007*, 2007, pp. 56–63.

[11] S. Latrée, P. Simoens, B. De Vleeschauwer, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, S. Van Den Berghe, and E. de Lumley, "Design for a generic knowledge base for autonomic qoe optimization in multimedia access networks," in *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, April 2008, pp. 335–342.

[12] N. Sheridan-Smith, "A distributed policy-based management (pbm) system for complex networks and services," Ph.D. dissertation, University of Technology, 2007.

[13] S. van der Meer, A. Davy, S. Davy, R. Carroll, B. Jennings, and J. Strassner, "Autonomic networking: Prototype implementation of the policy continuum," in *Broadband Convergence Networks, 2006. BcN 2006. The 1st International Workshop on*, April 2006, pp. 1–10.

[14] K. Barrett, S. Davy, J. Strassner, B. Jennings, S. van der Meer, and W. Donnelly, "A model based approach for policy tool generation and policy analysis," in *Global Information Infrastructure Symposium, 2007. GIIS 2007. First International*, July 2007, pp. 99–105.

[15] K. Dunsire, T. O'Neill, M. Denford, and J. Leaney, "The abacus architectural approach to computer-based system and enterprise evolution," in *Engineering of Computer-Based Systems, 2005. ECBS '05. 12th IEEE International Conference and Workshops on the*, April 2005, pp. 62–69.

[16] E. Al-Daoud, "A comparison between three neural network models for classification problems," *Journal of Artificial Intelligence*, 2009.

[17] C. Young, "Microsofts rule engine scalability results - a comparison with jess and drools," http://geekswithblogs.net/cyoung/articles/54022.aspx, September 2005.