Future Network and MobileSummit 2011 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2011
ISBN:978-1-905824-25-0

Cognitive Management of Procotol Composition Enhanced Future Internet Elements

David P. WAGNER[1], Jens MOEDEKER[1], Mathieu BOUET[2], Gerard NGUENGANG, Robin SCHAFFER[3]
[1]*Fraunhofer FOKUS, Birlinghoven, Germany*
Email:{david.wagner,jens.moedeker}@fokus.fraunhofer.de
[2]*Thales Communications, Paris, France*
Email:{Mathieu.BOUET,Gerard.NGUENGANG}@fr.thalesgroup.com
[3]*A1Telekom, Vienna, Austria*
Email:Robin.Schaffer@a1telekom.at

Abstract: This paper presents the final architecture for Cognitive Future Internet Element Management and in-network Dynamic Protocol Composition, as well as the results and insights gained by an prototype implementation and experiments performed. The high complexity of today's network infrastructures and the demand for reliable and robust but inexpensive networks stimulated research on evolving networks to become more autonomous and therefore reducing the need for human intervention. The presented approach is based on integrating a Monitoring, Decision-Making, and Execution (MDE) cycle into the situation aware Network Element Cognitive Manager (NECM). Accepting that the end-to-end paradigm does not hold for all mechanisms, we enable the NECM to control in-network functionality. For the prototype the NECM is complemented by the Dynamic Protocol Composition Framework (DPCF) which allows to control and execute arbitrary protocol functionality in intermediate nodes. For the experiments we considered a wireless mesh backhaul network which is likely to suffer from packet loss depending on external conditions like weather, reflections on moving objects etc. We could show that packet loss or overload have been recognized and appropriate actions have been taken. The analysis of the experiments showed that the different phases of the M-D-E cyle differ in time consumption by several orders of magnitude which requires architectures for cognitive network management to be designed to align the different processes to achieve optimal results.

Keywords: Future Internet, dynamic protocol composition, cognitive network management, autonomic communication, communication protocols

1. Introduction

The overwhelming development of the range and diversity of services offered via Internet in recent years increased the importance of reliable availability of suitable network access. On the other hand this success combined with the increased heterogeneity of converging fixed, wireless and mobile networking technologies has led to high complexity of network infrastructures which makes network management more demanding. Although unified management schemes and information models eased the technical management to some extent, ensuring reliability, robustness and availability has become an complex and expensive challenge. So significant research has been devoted to evolving networks to become more autonomous and therefore reducing the need for explicit network management and at the first place human intervention [1].The presented results have been developed in the Self-NET project [2] which aims at engineering the Future Internet based on cognitive behaviour of network elements with a high degree of autonomy.

The developed concepts of cognition and situation awareness [3] in the network nodes also allow to extend the role of routers: In the traditional Internet model intermediate nodes are expected to provide oblivious forwarding only [4]. Nevertheless today's so-called middle boxes break the end-to-end paradigm which caused fundamental discussions. We accept this development and propose to use the cognitive elements to cooperatively and more transparently configure functionalities at the best suited places in the network. This led us to researching Dynamic Composition approaches which have received increased attention in recent Future Internet research [5]. In contrast to many others working on composing complex service-like functionalities we focussed on the composition of small building blocks like Automatic Repeat Request (ARQ). These mechanisms may even appear in several layers in the traditional protocol stack, e.g. in a 802.11 link layer and TCP. Nevertheless this setup fails to achieve optimal performance in certain scenarios like wireless mesh networks. Therefore we consider them being a perfect target for cognitive situation-aware dynamic composition. Our use case is a wireless mesh backhaul network which is likely to suffer from packet loss depending on external conditions like weather, reflections on moving objects etc. We set up a experimentation scenario where packet loss is cognitively detected and compensated by means like routing adaptation or activation of redundancy mechanism like ARQ or Forward Error Correction (FEC).

So the work presented covers the connection of two topics: how to design a framework for cognition in network elements and how to manage protocol functionalities in such nodes. We present the overall architecture in section 2., provide information about the implementation in section 3. and describe the experiments performed with this implementation in section 4.. In the last section we conclude our insights and outline future work.

2. Cognitive Management Framework

Empowering network elements with cognition through the design of a Monitoring, Decision-Making, and Execution (MDE) cycle enables the self-management of network elements and therefore reduce human intervention in the management process. The scheme of distributed multi-agent frameworks is considered a good candidate for Autonomic Network Management [6] and applied in our approach. A network of nodes controlled by NECMs would be able to adapt on its own running configuration given a set of high level goals, i.e., inference rules, defined by the network operator. In the following we present the overall framework composed of the NECM and the DPCF.

2.1 NECM Architecture

In order to support different equipment and to be evolvable an NECM is composed of a set of functional blocks as depicted in figure 1 that carry all the different aspects of the MDE cycle. The cognitive manager results from the interaction of the following modules:

Topology service: It is in charge of the discovery of the neighbourhood physical topology. It sends hello messages on its network interfaces, collects the hello messages generated by the adjacent nodes and builds adjacency tables that reflect the physical network topology. The topology is for example useful to identify the IP addresses of the neighbouring nodes, and evaluate the impact of a link or node failure on the active network services.

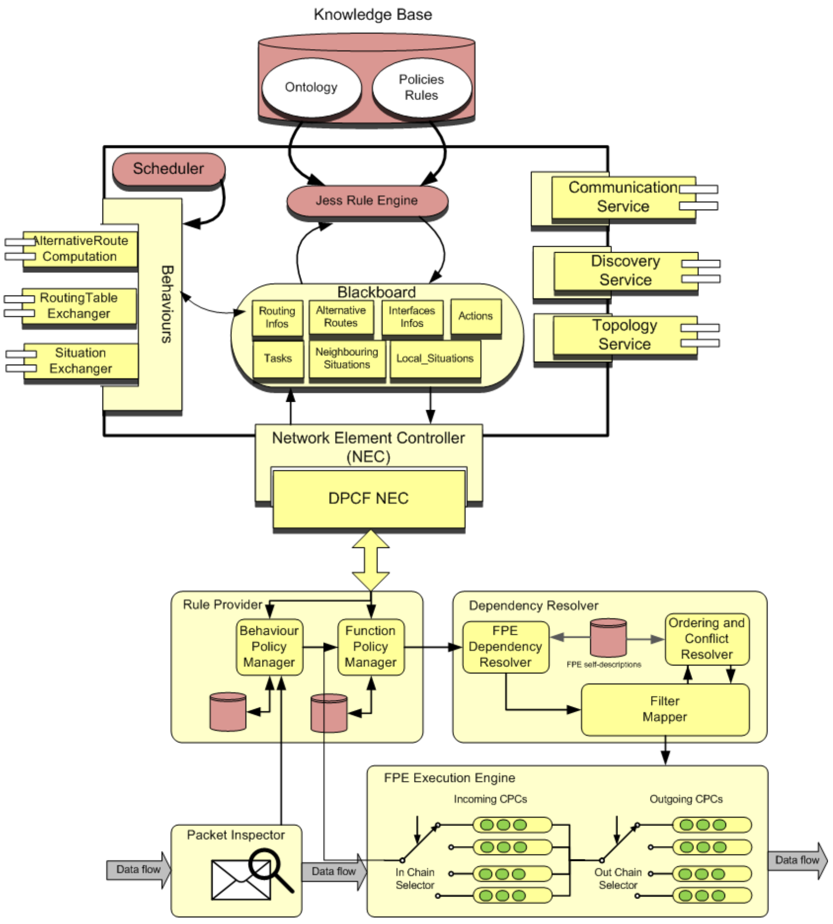

Figure 1: Overall framework.

Discovery service: Cooperation and coordination among the NECMs are based on knowledge exchange. The Discovery service maintains up to date information concerning neighbouring cognitive network managers (IP addresses, network interfaces used to reach them etc.)

Communication service: It provides communication facilities to the cognitive manager. It is in charge of message exchange among NECMs. These three modules are called services because they are fundamental device agnostic features of the cognitive framework.

Network element controller (NEC): It is composed of sensors and actuators. Sensors provide a framework compliant API for device monitoring and allow the collection of relevant metrics that describe the status of the device. Actuators are also vendor-specific API which enforce reconfigurations in the network equipment. The collected data are semantically annotated by the NEC and published in the blackboard.

Behaviours: A behaviour realizes tasks or computations that are inherent to a specific objective. It can work in three modes: one shot, periodic and publish/subscribe.

Scheduler: It triggers the execution of registered schedulable components. Some components, mostly the behaviours, have to be activated either once, either periodically and the scheduler is in charge of orchestrating each components according to its profile (one shot, periodical).

Blackboard: It acts as the knowledge bus of the cognitive manager providing writing and reading facilities to the other modules. The blackboard is composed of a set of topics where related information is written and put at the disposal of the modules that are interested in.

Inference engine: It is the brain of the cognitive manager. It is responsible of high level decision making and relies on the knowledge and rules capture in the ontology. Its decisions are handled, through blackboard's topics, by dedicated behaviours that transform them into a sequence of atomic actions. Some of these atomic actions are set up by the NEC directly into the device.

2.1.1 Instantiation

The central objects are the ontology, the inference rules and the behaviours which are presented in more detail in the following:

Ontology description The role of the ontology is to provide the necessary know-how to cognitive managers to enable them to automatically recognize critical situations and to elaborate relevant remedy actions. Figure 1 presents the top level elements of the ontology.

Owl:Thing is the root concept of mostly all OWL-based ontologies. The notion of *Service* is of high importance for a network administrator and the aim of its daily management activities is to maintain the required QoS for each deployed network service.

A *Policy* is associated to each *Service* and defines the boundaries in which the operational state can be considered as normal. When the associated metrics are out of the bounds defined in the Policy instances, the network is in a critical situation. On the contrary, when the collected metrics are in the bounds, the network is in a normal situation. The *Situation* concept, therefore, characterises the running status of network services with respect to their associated policies. Each situation is characterised by a set of symptoms and the symptoms themselves depend on the respect or the violation of the policies.

The *Action* concept represents the different actions that can be applied by a cognitive manager. All these knowledge entities are at the level of a single node and some actions may require knowledge exchange between neighbour nodes.
The ontology was extended to fit the DPC functionalities (see figure 1). Through DPC API, ARQ can be activated or deactivated if a high packet error rate (PER) is detected on a path and paths can be added and deleted if client streams experience high load. The sub-concepts derived from this are:

Node: Two concepts are added to monitor the node on which the NECM is deployed. The *Interface* concept enables to model the state of a network interface. Its data properties are the name of the interface (for example, eth1), its IP address, its packet loss rate, and its ingress and egress utilisations. These data are collected using the DPC API. The *MonitoredPath* concept allows to describe the paths which have to be monitored for a possible change route action. Its data properties correspond to the data available in the path table. In addition, the *Neighbour* concept is linked to the Node concept in order to describe all the neighbour nodes of the NECM.

Situation: The Situation concept is derived in several sub-concepts. The *HighPER* concept is instantiated if the inference rule that associates a PERPolicy to the packet-Loss of an Interface is matched. It describes the state of a node which is experiencing a high PER. The *NgHighPER* and *NgLowPER* concepts describe nodes neighbours that are experiencing a high PER or no more packet loss respectively. Since ARQ has to be activated upstream, a NECM has to know which neighbours have high or low PER. Each NECM infers if it is in a HighPER state and informs its neighbours. This state becomes a NgHighPER instance for them. A NgLowPER state is deduced by each NECM if it does not received HighPER notifications anymore. Finally, the ***ARQActivated*** concept enables to describe in the ontology the situation of activation of ARQ. It is used to infer the deactivation of ARQ when needed.

Policy: The Policy concept is extended with the ***RoutingPolicy*** and the ***PERPolicy*** concepts that respectively comprise a threshold to set the link utilisation limit for which a change route action has to be launched, a threshold to set the PER limit for which ARQ has to be activated.

Action: The ***ChangeRoute***, ***ActivateARQ*** and ***DeactivateARQ*** concepts are instantiated when the inference rules that define these actions are matched.

Inference rules Four inference rules were defined in order for cognitive managers to make the appropriate decisions when assessing situations. The ***High PER situation rule*** enables to locally detect if the node is experiencing a high PER. The ***ARQ activation rule*** enables to decide to activate ARQ if one of the links of the node is experiencing high PER. This situation is reported downstream. The ***ARQ deactivation rule*** enables to decide to deactivate ARQ when there is no more PER and the ***Change path rule*** to change a high-loaded path that has been assigned to client streams.

Behaviours Behaviours perform tasks or computations that are inherent to specific objectives. Three different behaviours are used in NECM adapted for DPC use case:

- **RoutingTableExchanger:** This behaviour works both in publish/subscribe mode and in periodic mode. It is activated every time that the local routing table is read by the NEC in order to send the collected paths to NECM neighbours. In addition, it periodically collects routing tables sent by NECM neighbours and, if any were received, launches the alternative routing table computation behaviour.
- **SituationExchanger:** This periodic behaviour collects NECMs local situations (HighPER) and sends them to neighbours. It also collects neighbours local situation and processes them for the inference engine; in other words it converts received HighPER facts into NgHighPER facts. Finally, this behaviour monitors PER decrease verifying if a neighbour that experienced high PER has not report this situation since at least 5 seconds. If it is the case, it generates a NgLowPER fact for the inference engine.
- **AlternativeRoutesComputation:** This behaviour works in publish/subscribe mode. It is activated every time that the RoutingTableExchanger behaviour receives a neighbour's routing table. It is in charge of computing the alternative paths in comparison to the local routing table.

2.2 Dynamic Protocol Configuration

The architecture developed for implementing Dynamic Protocol Composition (DPC) was presented in detail in [7], therefore details will be omitted in this paper. In general it allows the NECM to configure policies that define which functionality shall be applied to packets matching a so called packet filter. The active policies are resolved to a consistent configuration and put into action by the Execution Engine. Implemented functionality includes defining forwarding rules, inserting sequence numbers, acknowledging and retransmitting of missing packets, so rerouting and in-network ARQ could be demonstrated.

One important property of this architecture is the support for several levels of behaviour policies [8]. These simple behaviours allow the NECM to define reactions for simple conditions in advance so the need for the full (time and resource consuming) decision making process is limited to exceptional incidents.

3. Implementation

3.1 NECM

The Java-based service-oriented framework OSGi was used to implement NECMs. The CMF modules consist of bundles that can be loaded, started, stopped and replaced at runtime. This technology corresponds to the adopted design guidelines. In addition, the ontology is described with Web Ontology Language (OWL) while rules are written with Semantic Web Rule Language (SWRL). They are incorporated into Jess, a Java-based inference engine.

3.2 DPCF

The DPCF is implemented in C++ for Linux to allow for direct access to network interfaces and acceptable performance. For details, please refer to [7].

4. Experimentation

The prototyping environment consists of routers (four Linux based PCs), two end-nodes and a packet dropper connected with Fast Ethernet which emulates a wireless mesh network environment with redundant paths (the setup is similar to the one used in [7] and are not depicted due to space limitations).

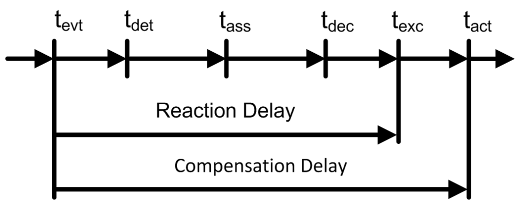

Figure 2: Measured points of time

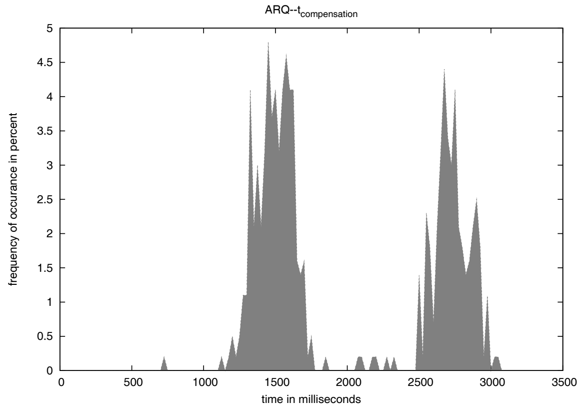

Figure 3: Frequency of Compensation Delay

To induce critical situations either packets may be randomly dropped on one link or additional traffic on one path may be created. After the occurence of the problem (t_{evt}), the implemented architecture will detect the problem by continuous monitoring (t_{det}), assess the situation (t_{ass}), decide on the action to be taken to compensate the cause of the problem (t_{dec}), enforce the respective execution (t_{exc}) and at some time the result of these measures will be visible in the network (t_{act}) and monitored. We repeatedly (500 times) executed uniform experiments for both scenarios and measured the duration of each step. Figure 2 shows the measured points of time and figure 3 shows the frequency of compensation delays for 500 experiments for ARQ activation. Obviously the timescale is in the range of seconds and there are two remarkable peaks which correlate to the periodic exchange of symptoms of the two concerned NECM instances.

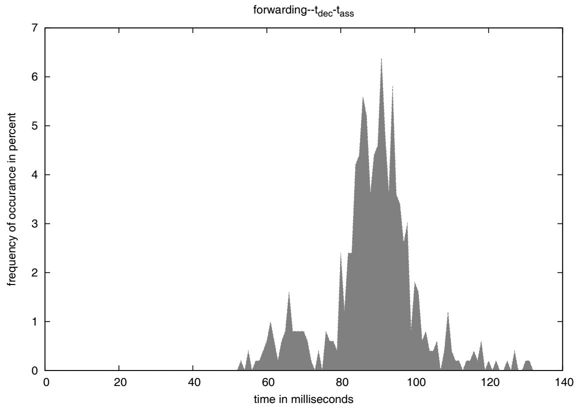

Figure 4: Frequency of decision duration

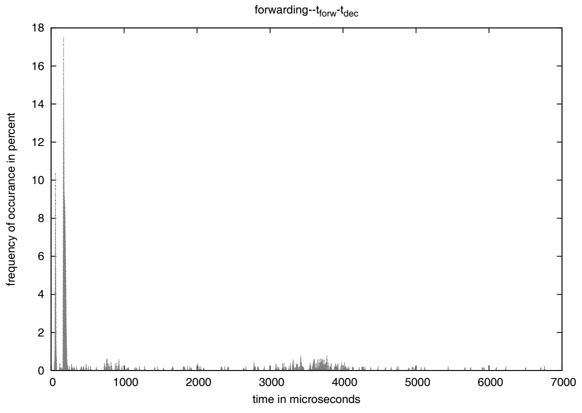

Figure 5: Frequency of execution duration

Figures 4 and 5 show the difference of the different phases: figure 4 shows the delay induced by the decision making for the high-load-scenario which is about eighty to ninety milliseconds. Figure 5 shows the delay induced by execution (which is immediately effective in this case, so $t_{exc} = t_{act}$) which is more scattered due to the user space implementation but has the main peak at about 100 microseconds. So the delay of these two mechanisms differ by three orders of magnitude.

Assessment of experiments' results This shows that mechanisms have to incorporated in the architecture to reduce the number of decisions to be taken by the NECM because of the high overall delay. The results also show that urgent decisions can not be taken directly by the NECM as the entity with the full cognitive view. Here architectures like proposed in [8] have to be elaborated further.

5. Conclusion and Outlook

The concept of integrating cognition in Future Internet Elements is promising and proved to allow increasing network performance in the selected scenario. The primary goal of reducing need for human intervention and by this also reducing frequency and duration of faults could be achieved. Another compelling property of our approach is that the evolved introduction of stateful middle boxes in the network may be developed further to an integrated and well controllable part of the Future Internet ecosystem. The experiments show that the different phases of the Cognitive Cycle differ heavily in operating time scales. Although the implemented rule sets are very basic, we already see the need to consider the different run time properties in the control architecture. The contradiction between instant reaction and well-founded decision-making, taking many also external factors into account, will have to be resolved individually for each scenario and execution capability.

The concept of Cognition in Future Internet Elements allows to handle and make use of more dynamic execution capabilities than human administrators are able to handle. Nevertheless this increase in flexibility comes at a cost: any managed capabilities have to be understood in detail and then modelled in the rules and policies of the NECM. In this area there is more research and development needed to integrate cognition in today's and future networks rendering communication more reliable, robust and efficient.

Acknowledgment The work presented in this paper was supported by the European Commission Seventh Framework Programme through the Self-NET Project [2].

References

[1] A. Kousaridas, C. Polychronopoulos, N. Alonistioti, A. Marikar, J. Mödeker, A. Mihailovic, G. Agapiou, I. Chochliouros, and G. Heliotis, "Future internet elements: cognition and self-management design issues," in *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, Autonomics '08, (ICST, Brussels, Belgium, Belgium), pp. 13:1–13:6, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[2] "Self-NET (Self-Management of Cognitive Future InterNET Elements)." EU FP7 project INFSO-ICT-224344. `https://www.ict-selfnet.eu`.

[3] A. Mihailovic, I. P. Chochliouros, E. Georgiadou, A. S. Spiliopoulou, E. Sfakianakis, M. Belesioti, G. Nguengang, J. Borgel, and N. Alonistioti, "Situation awareness mechanisms for cognitive networks," in *ICUMT*, pp. 1–6, IEEE, 2009.

[4] D. Clark, "The design philosophy of the darpa internet protocols," in *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, (New York, NY, USA), pp. 106–114, ACM, 1988.

[5] C. Henke, A. Siddiqui, and M. R. Khondoker, *Network Functional Composition: State of the Art*. Will be published in the proceedings of ATNAC, Auckland, Newzealand, 31 OCT - 03 NOV, 2010, October 2010.

[6] J. Boite, G. Nguengang, M. Isral, and V. Conan, "Conemaf: A modular multi agent framework for autonomic network management.," in *ICAART (2)* (J. Filipe, A. L. N. Fred, and B. Sharp, eds.), pp. 224–231, INSTICC Press, 2010.

[7] D. Wagner, J. Moedeker, and T. Horstmann, "Dynamic protocol functionality in cognitive future internet elements," in *Proceedings of Future Network & MobileSummit 2010 Conference*, p. 8, Jun 2010.

[8] D. Wagner and J. Moedeker, "Towards dynamic protocol configuration and its configuration and control in autonomous communication environment," in *MOBILIGHT* (P. Chatzimisios, C. V. Verikoukis, I. Santamaría, M. Laddomada, and O. Hoffmann, eds.), vol. 45 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 334–345, Springer, 2010.