ETSI GS AFI 002 V1.1.1 (2013-04)

Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture (An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management)

Disclaimer

This document has been produced and approved by the Autonomic network engineering for the self-managing Future Internet (AFI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/AFI-0002

Keywords
architecture, autonomic networking, cognition,
generic, model, network, self-management

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification (ISG) Autonomic network engineering for the self-managing Future Internet (AFI).

NOTE 1: The present document contains some content that was published earlier in some scientific publications and has been re-used or evolved by the original authors (copyright is acknowledged and citations provided). Examples include concepts related to the earlier versions of the GANA Model that has been evolved in the present document.

NOTE 2: Where there are some figures taken from the public domain and have been included for providing information/illustrations to readers or have been modified to fit into the architectural framework, Copyright Acknowledgments to the authors and publishers have been added apart from referencing/citing the original publisher. Acknowledged Copright owners include: Taylor and Fransis Group, LLC, Auerbach Publications, publishers cited for MAPE, GENI, FOCALE, 4D, Knowledge Plane for the Internet, CONMan, Deriving Knowledge for the Knowledge Plane (a NIST Draft), Concepts developed in EC funded EFIPSANS project and other EC funded projects listed at the end of the present document in acknowledgements appendix.

1 Scope

The main objective of the present document is to define, iteratively, a generic, conceptual architectural reference model intended to serve as guideline for the design of the future generation networks exhibiting autonomic characteristics or capabilities. The technical content of the present document consists of a set of fundamental design principles elaborating on the functions, processes and interfaces of autonomic networks and systems.

The present document starts by analysising. The starting point of this work relies on the analysis of the existing research initiatives, autonomic architectures and standards for telecommunications network operation and management. This analysis allows to identify the gaps (or enablers to be introduced) between the current situation and the requirements and expectations expressed by the actors of the telecommunications environment.

These requirements are detailed in the GS AFI 001 [i.51] Scenarios, Use Cases and Requirements for future self-managing networks. In this way, the approach used to elaborate the present document is requirement-driven. In addition to these requirements, the analysis allows to list the networking functions commonly needed or employed in the current or emerging architectures, and identifying the missing functionalities.

The present document defines and describes what we call a ***Generic Autonomic Network Architecture (GANA) Reference Model for Autonomic Network Engineering, Cognitive Networking and Self-Management***. The present document was produced by AFI WI#2 as shown on the figure below and subsequent releases of the Specification will be produced and published as the Reference Model evolves. The Generic Autonomic Network Architecture (GANA) Model is a Conceptual Architectural Reference Model for Autonomic Network Engineering, Cognition and Self-Management. Its purpose is to serve as a "blueprint model" that prescribes the design and operational principles of "autonomic decision-making manager components/elements" responsible for performing "autonomic" and "cognitive" management and adaptive control of resources. It is not an implementation architecture per se. [An elaborate definition is given in the next clauses]. Two aspects need to be distinguished as indicated on Figure 1:

1) a Generic Reference Model (specified in the present document); and

2) autonomicity-Enabled Reference Architectures that are the result of "instantiating" selected or all Functional Blocks and Reference Points defined in the Reference Model onto a target implementation-oriented standardized Reference Architecture. Clause 5 discusses the subject in detail.

In this first release of the GS AFI 002 (the present document), we aim to sketch a primary generic architecture model definition, it is essential properties, its principles, its building blocks and their interactions. In this sense, this work may be considered as a continuous process advocating for an incremental release of the AFI Generic Architecture reference model intended to include enhancement, refinement and revision as the techniques mature, and contributions from the community grow.

Figure 1 presents the current AFI Work Items and their relationships.

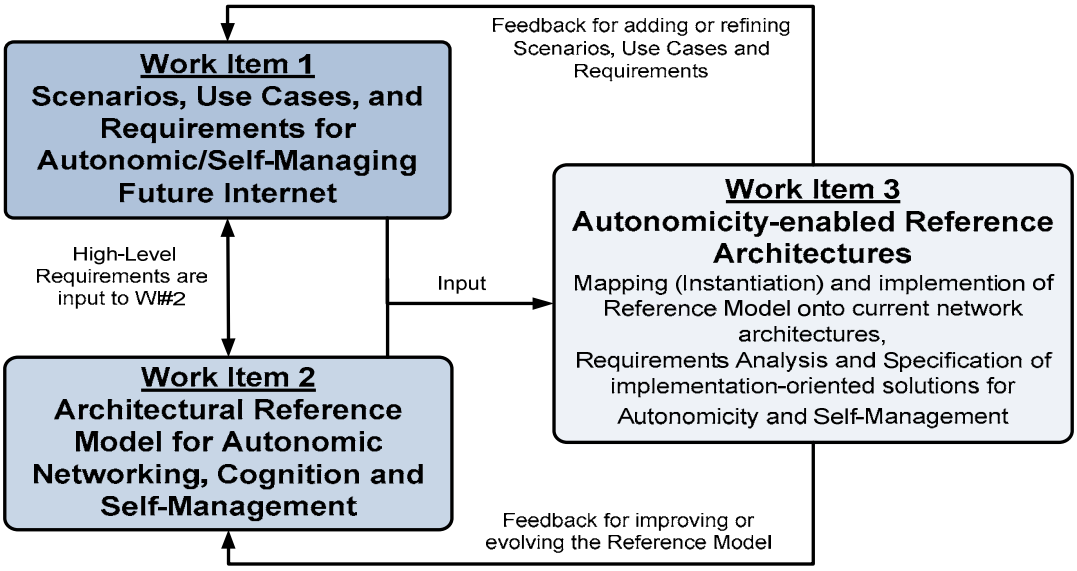


Figure 1: Current AFI Work Items and their relationships

The present document provides the definition of an abstract reference representation (or architecture) for specifying physical autonomic network elements and systems.

Individual Functional Blocks (and the associated principles) of said abstract representation could be seen as functional elements, or architectural components, performing certain functions, and interfaces (through which they interact).

The Abstract Architectural Reference Model presented in the present document is described in a technology independent way.

Characteristics of the Reference Model, including where reference points and governance issues are presented and elaborated, what to standardize, etc, are all presented in the present document. The approach taken in the present document is twofold:

1) Working out and Unifying viable Concepts and Design Principles for Autonomic Networking, Cognition and Self-Management within a single Unified Framework (i.e. the Reference Model). In so doing, the present document also includes the Techniques and Guidelines for addressing certain types of problems specific to the process of designing, implementing, validating and deploying Autonomic Networks. Since the approach is "to put ourselves in the shoes of the designer (primarily)", we seek to provide all necessary guidelines to the industry w.r.t. to designing autonomic functions and incorporating them into the network and its elements.

2) Identifying the items that can be standardized as guided by the Reference Model for Autonomic & Cognitive Networking and Self-Management, e.g. Functional Blocks (FBs) specific to realizing autonomicity, cognition and self-management, Reference Points and Data Models describing the information/data communicated on FBs' Interfaces, Autonomic Behaviours (i.e. control-loops) spanning multiple network elements, etc.

AFI is not only addressing the autonomic network management in the sense of automation of management processes as well as introducing control-loops in the traditional Management Plane, but AFI takes a holistic approach to the problem of "Self-Management and Adaptive Control" in the network elements and the network as a whole. This means there are 3 views to "management" that we consider and introduce Autonomics/Self-Management:

1) The traditional Vertical View that looks at the interface between a Network Element and the Management System (EMS's/NMS's), as well as the whole Vertical Management Framework.

2) A Horizontal View of a "management-like" behaviour that may be called a "network-intrinsic" management or "in-network management" that involves the collaboration of network elements along an E2E path (either hop-by-hop or elements on some elements that are not necessarily on-link neighbours). It may simply be viewed as an enhancement to the Control Plane with interacting distributed control-loops that enable network elements to negotiate configurations and to adaptively control the behaviour or resources (protocols included) i.e. to collaboratively self-optimize.

3) The "device/element-intrinsic" self-management aspects i.e. autonomic functions introduced into the architecture of a Network Element.

All the 3 views have some implications on Architectural Principles for both Evolved Networks and Future Network Architectures that exhibit Self-Management Capabilities from the dawn of their design, and therefore the Concepts and Principles defined by the Reference Model can also be applied to Future Network Architectures designed from scratch, apart from being applicable to the evolution of existing network architectures towards "Autonomicity-Enabled Architectures". The Reference Model presented in the present document covers all the 3 views listed above, in order to capture a holistic picture on the evolution of management paradigms.

1.1 Reading Guide and Snapshots of the Reference Model

The following is a Reading Guide aimed at guiding readers (users) with different levels of knowledge and background in Autonomics, Cognitive networking and Self-Management, and with diverse interests on various aspects of the Reference Model and contents of the present document.

To be directed to specific chapters of interest for you the Reader (User) here is the guide:

1) The scope summarizes the scope and links to other documents or Work Items in AFI (Readers/Users: ALL are recommended to read).

2) Readers/Users who are not familiar with the domain of Autonomic Networking, Cognition and Self-Management and want to know about Characteristics and Properties of systems exhibiting autonomic/self-management capabilities, can start their reading from clauses: 4. Definition of the AFI Architectural Reference Model of a Generic Autonomic Network Architecture and its fundamental requirements; 5. Properties of a Generic Autonomic Network Architecture Reference Model; 6. Enabling concepts and mechanisms.

3) Readers/Users who are already familiar with Autonomic Networking, Cognition and Self-Management, and simply want to focus on understanding the Reference Model can move on to clause 5 and then (just browse through clause 8) or move straight to clauses 9 to 12, 13 and 14 that summarizes the various Reference Points at the end. The snapshots of the Reference Model provided in this clause (see below) give a nutshell view that should be enhanced by knowledge acquired in reading clauses 9 to 14 where Functional Blocks, Reference Points and Principles are described in detail. Figures in the clauses 9 to 14 define the Reference Model. The main figures are: Figure 2 to Figure 5; Figure 7 to Figure 11; Figure 14 to Figure 30; Figure 34, Figure 39, Figure 49, Figure 63, Figure 64, Figure 67, Figure 68 and Figure 69.

4) Readers/Users who are already familiar with the domain of Autonomic Networking, Cognition and Self-Management, and want to know how Concepts and Principles from various Models related to viable approaches to the field, such as the IBM-MAPE model, FOCALE, 4D, CONMan, GENI, Knowledge Plane for the Internet, etc., are all incorporated, harmonized and accommodated within GANA Reference Model as a single Unifying Model, can move to clause 8 and also read the table in clause 10.12. The Table describes how concepts from the different viable approaches to the domain have been selectively combined in a harmonized way or accommodated within a single unifying model referred to as the GANA Reference Model in the present document.

5) Readers/Users who want to know where "Place-holders for Controls-Loops" and Hierarchical and Horizontal relationships/interfaces between them are defined, as well as where Cognition can be developed, can go through the main figures of the Reference Model in clauses 9 to 11 (see also the snapshots in this clause as well). Wherever there is a Decision-Element (DE), a Control-Loop for the DE can be designed. A "one-to-one" mapping/assignment between specific DEs and the specific types of Managed Enties (MEs) and their Configurable and Controllable Parameters over which a control-loop can be designed for a particular DE, is provided in a table in clause 10. The notions of "fast control-loops" that can go into a Network Element (NE) and "slower outer control-loops" are described in clause 9.

6) Readers/Users who want to know about Techniques and Guidelines for addressing certain types of problems specific to designing, implementing, validating and deploying Autonomic Networks, such as how to address Stability of control-loops, Trust and Confidence building by the operator can find details in clauses 10 and 11.

7) Readers/Users who want to know about the Reference Model's mappings and implications to the TMN Logical Layered Architecture (LLA), reflection of the FCAPS Framework within the Reference Model, and how the Reference Model impacts or can be used to evolve/enhance existing EMSs/NMSs (OSS's) and other types of implementation-oriented aspects can see details in clause 12. Topics such as Network Governance are also covered in the corresponding clauses.

1.2 Snapshots of the Reference Model (a nutshell view)

The GANA (Generic Autonomic Network Architecture) Reference Model is a unified model for Autonomic Networking, Cognition, and Self-Management. It defines generic **Functional Blocks** and associated **Reference Points** and Characteristic Information that are specific to enabling autonomics, cognition, and self-management in a target architecture. Therefore, it can be "instantiated" onto an implementation-oriented reference architecture such as the 3GPP architecture, BBF architecture, or ITU-T (NGN) architecture. The generic Functional Blocks and Reference Points can also be applied in designing future network architectures that exhibit self-management capabilities from the dawn (outset) of their design.

To what types of stakeholders is the Reference Model addressed to? Network architects, researchers, and developers/implementers "refer" to the Reference Model when reasoning about or applying the concepts and principles defining the domain of autonomic communication, autonomic networking, autonomic and cognitive management and control-all as part of the "big-picture" of Self-Management.

Figures 2 and 3 provide just snapshots of the Reference Model for selected key aspects. Figure 2 provides an overview of some of the key aspects of the GANA Reference Model. Figure 3 presents an instantiation of GANA Model within a router, to illustrate instantiation (though not complete instantiation), with respect to autonomic routing as a working example. Figures 2 and 3, are simplified versions of the more detailed Figure 34 in the core part of the present document. In general, self-manageability in GANA is achieved through instrumenting the network elements (in Figure 3, the case of routers) with autonomic Decision-making-Elements (DEs) that collaboratively work together. DEs may form "peers" along a path within the fundamental E2E transport architecture. The DE-2-DE peers need not necessarily be hop-by-hop neighbours (i.e. being resident in on-link neighbouring nodes) but the peer relationships may relate to e.g. border-relationships management in a heterogeneous network or may related to some DEs in certain network elements along an E2E path. The Reference Model defines a hierarchy of DEs, i.e. four basic levels of self-management: the protocol, function, node, and network levels. Each DE manages one or more lower-level DEs through a control loop. These lower-level DEs are therefore considered Managed Entities (MEs). Over the control loop, the DE sends commands, objectives, and policies to an ME and receives feedback in the form of monitoring information or other type of knowledge. Each DE realizes some specific control-loop(s), and therefore, represents an "Autonomic Activity" or "Autonomic Function" (AF). Examples of Autonomic Functions: Autonomic QoS Management-DE; Autonomic Security Management-DE; Autonomic Fault Management-DE; Autonomic Mobility Management-DE, etc.

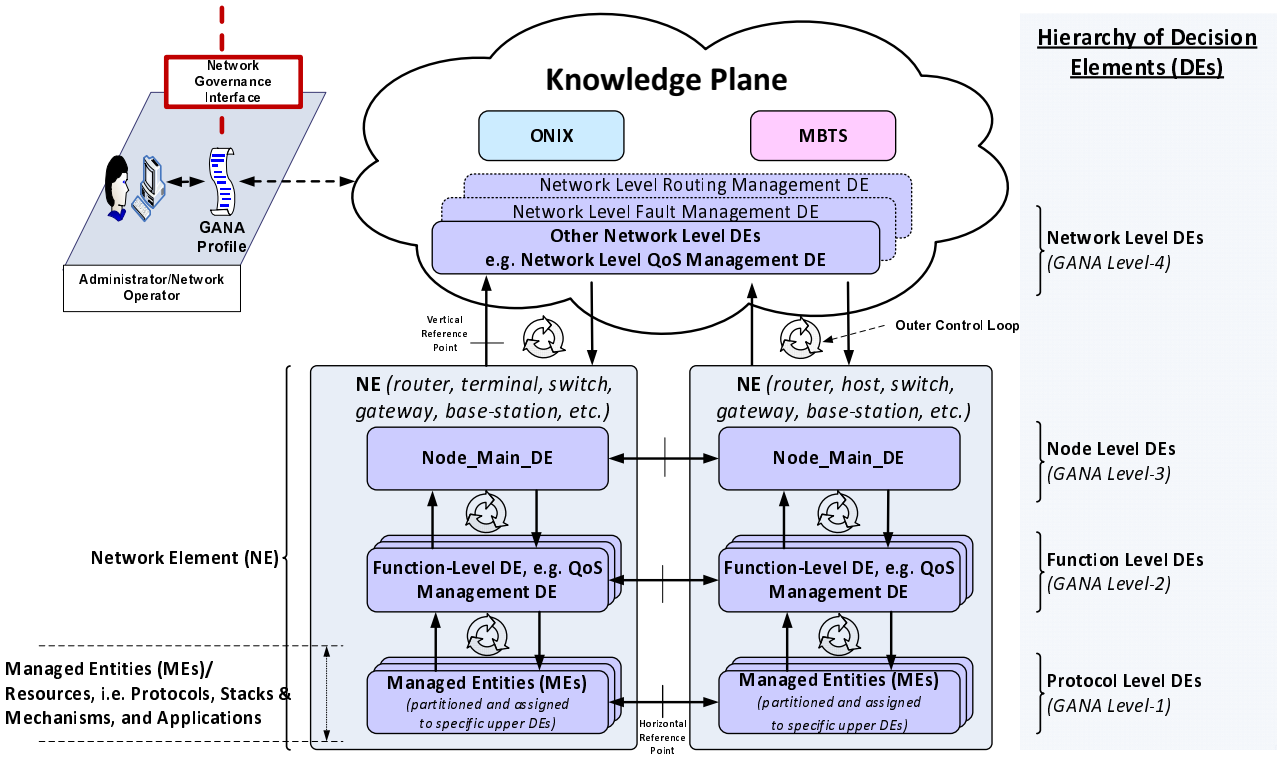


Figure 2: Global view of GANA Reference Model (a simplified view: Reference Points between Functional Blocks are omitted)

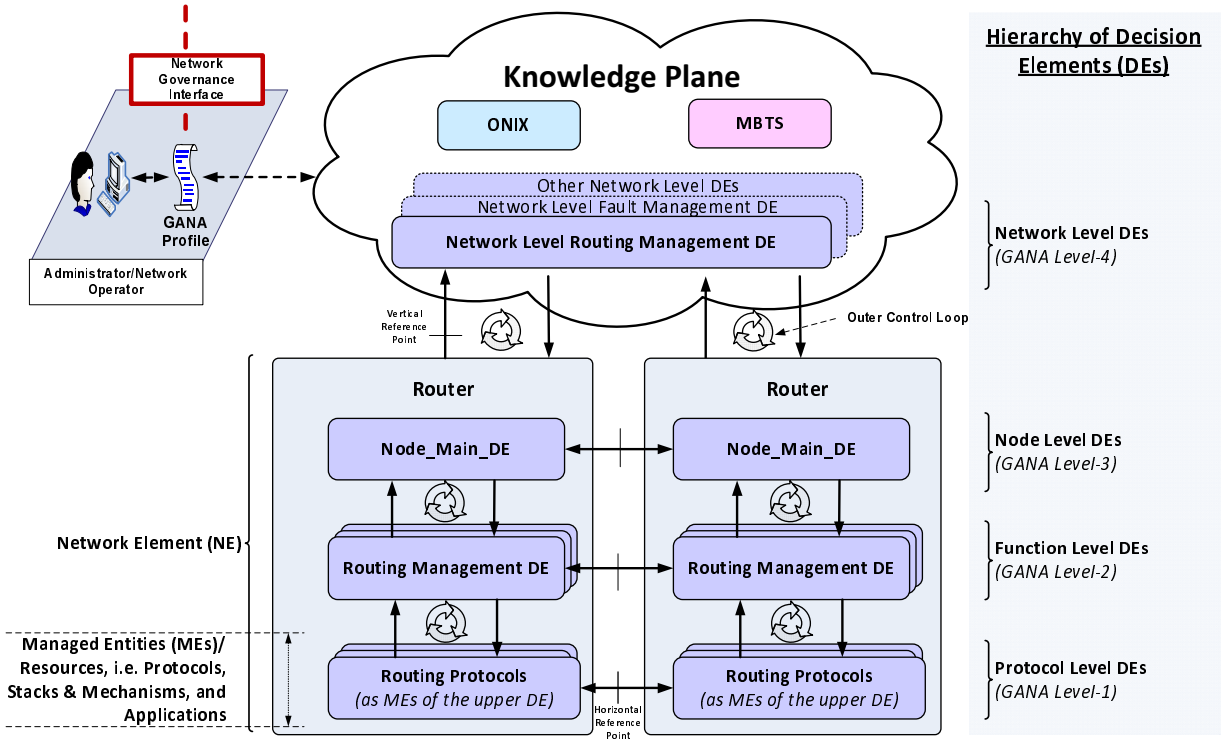


Figure 3: GANA Instantiation within a Router, w.r.t Autonomic Routing Functionality

The lowest level DEs are at the Protocol Level. They represent protocols, services, and other fundamental mechanisms, possibly already implemented in the network. OSPF can be considered an example of a Protocol Level DE. These DEs are managed by Function Level DEs such as the Routing Management DE, Monitoring DE, and QoS Management DE. Currently there are seven Function Level DEs defined in the Reference Model. Each of them is present in every Network Element. The orchestration of the Function Level DEs is performed by a Node Level DE (the Node Main DE) of which there is one in every Network Element. At the highest DE level, the Network Level DEs address similar aspects to the Function Level DEs but on a wider scope. Therefore there is a Network Level Routing Management DE, Network Level Monitoring DE, Network Level QoS Management DE, etc.

The Network Level DEs constitute the Functional Blocks of the Knowledge Plane, together with ONIX (Overlay Network for Information eXchange) and MBTS (Model-Based-Translation Service). ONIX is a distributed scalable system of information servers that form an Overlay Network for Information eXchange and provides a information publish/subscribe paradigm to effect advanced Auto-Discovery, while MBTS forms an intermediation layer between the Knowledge Plane and the Network Elements for the purpose of translating information and commands/responses (more details are given later in the present document). According to the Reference Model, 3-Levels of Hierarchical Control-Loops (GANA Level-2 to Level-4) that are realized by the corresponding Decision-making-Elements (DEs), which collaboratively work together from within a Network-Element (NE) up to the "Network-Level/Knowledge Plane", demonstrate how Autonomics/Cognition/Self-Management can be gracefully (non-disruptively) introduced in today's existing architectures. The Reference Model defines Four Basic Levels of Self-Management, but the three levels indicated are the most important ones when one considers not to embed a control-loop into an individual protocol (i.e. avoiding "protocol-intrinsic control-loops" since they may complicate network manageability and may create undesired emergent behaviour in complex protocol interaction scenarios as known today). This subject is discussed in more detail later in the present document. The place-holders for internal control-loops (inside a Network Element) depicted by the Reference Model enable to design and embed "node-local" Self-Management behaviours/algorithms, including node-local Self-Optimization, i.e. some degree of network element intelligence through the internal Decision Elements (DEs) that realize the internal control-loops. Example node-scoped Self-* behaviours that do not necessarily require collaboration/negotiation with other network elements include: Plug-n-Play; Energy Savings through autonomic functions; Autonomic Security Management (self-protection and self-defending behaviour); Autonomic Fault-Management and Resilience (proactively and reactively), etc.

On the Network Governance Interface, the network administrator can manage the operation of the whole autonomic network by authoring, validating and submitting Policies, High-Level Network Objectives and some Configuration-Data to the Knowledge Plane, all encapsulated together in the form of a GANA Profile. This profile is then translated by the Network Level DEs and commands are issued to lower level DEs for enforcement. In addition to defining the abovementioned Functional Blocks, the GANA Reference Model also specifies Reference Points (a Summary Table of Reference Points is provided at the end of the Specification). These can be either horizontal (i.e. between DEs belonging to same levels) or vertical (i.e. between DEs belonging to a different level). In horizontal DE-to-DE interactions, DEs form peering relationships used to exchange additional information as described in the Summary Table of the Reference Points, at the end of the Specification. Such "in-network" DE-to-DE communication and interaction of associated distributed Control-Loops may simply be viewed as an enhancement to the traditional distributed Control Plane with distributed Control that enable network elements to (re)-negotiate some configuration aspects and to adaptively control the behaviour of resources (protocols included) i.e. to collaboratively self-optimize the E2E transport network. More details and further elaborations on the various aspects of the Reference Model are covered in clauses 9 to 14.

Illustration of a few selected Reference Points in the Reference Model: figures 4 and 5 illustrate a few selected Reference Points and Characteristic Information exchanged.

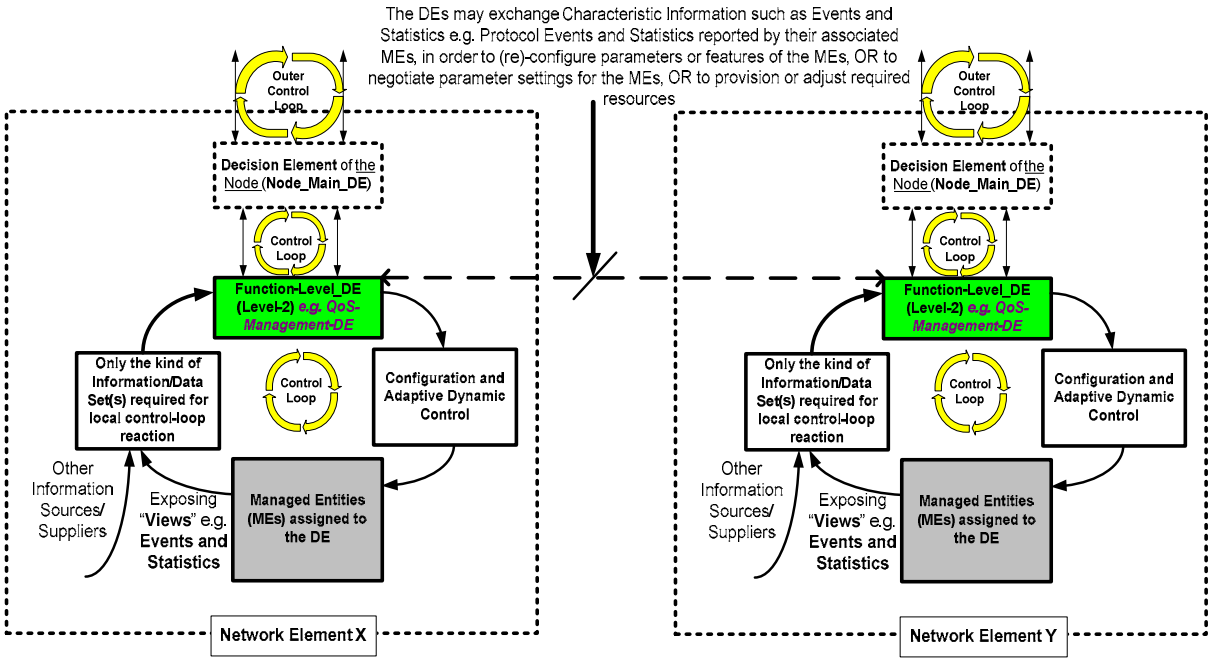


Figure 4: Example illustration of "in-network" DE-to-DE Communication and interaction of distributed Control-Loops

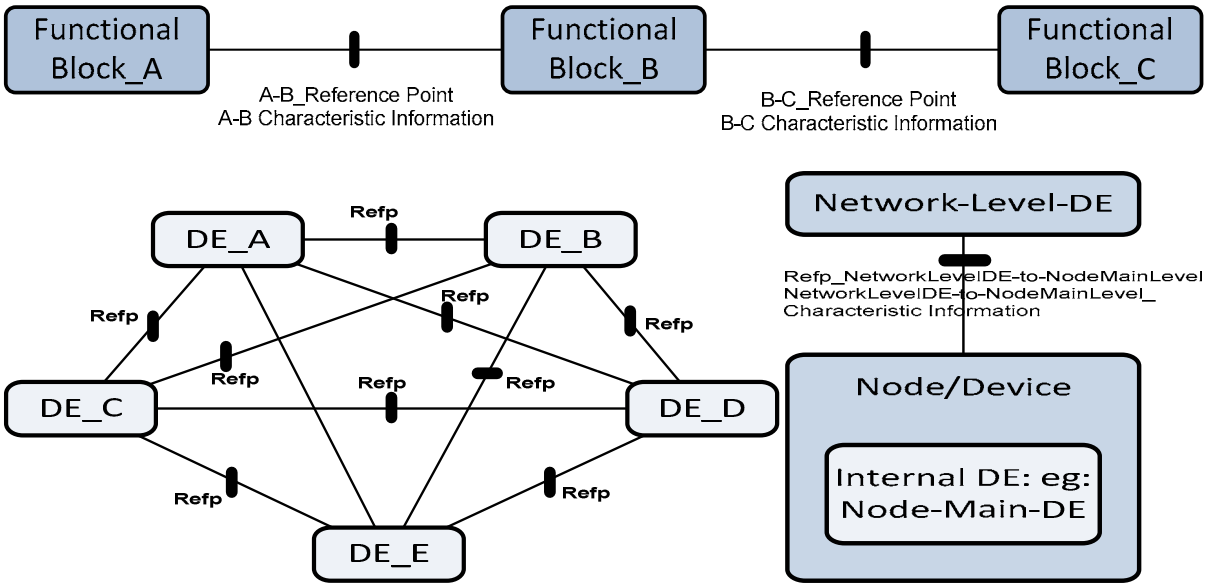


Figure 5: Illustration of some of the Reference Points and Characteristic Information

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Hierarchical Control System.

NOTE: See http://en.wikipedia.org/wiki/Hierarchical-control-system.

[i.2] The FCAPS Management Framework.

NOTE: Available at http://www.itu.int/rec/T-REC-M.3400/en.

[i.3] European IST FP6 ANA (Autonomic Network Architecture) Project, January 2006.

NOTE: Available at http://www.ana-project.org/.

[i.4] European IST FP6 Haggle Project, January 2006.

NOTE: Available at http://www.haggleproject.org/.

[i.5] EC FP7-EFIPSANS Project, 2008.

NOTE: Available at http://www.efipsans.org/.

[i.6] ETSI TS 122 127: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Service requirement for the Open Services Access (OSA); Stage 1 (3GPP TS 22.127 Release 9)".

[i.7] ETSI TS 122 105: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Services and service capabilities (3GPP TS 22.105 Release 9)".

[i.8] Hitesh Ballani and Paul Francis: "CONMan: A Step towards Network Manageability". In SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, pages 205-216, New York, NY, USA, 2007. ACM.

[i.9] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt and Andrew Warfield: "Xen and the art of virtualization". In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164-177, New York, NY, USA, 2003. ACM.

[i.10] Jason Bell: "Understand the Autonomic Manager Concept", October 2004.

NOTE: Available at http://www.ibm.com/developerworks/autonomic/library/ac-amconcept/index.html.

[i.11] IETF RFC 4012 (March 2005): "Routing Policy Specification Language next generation (RPSLng)", L. Blunk, J. Damas, F. Parent and A. Robachevsky.

[i.12] Raouf Boutaba, Jin Xiao, and Qi Zhang: "Autonomic Computing and Networking", chapter "Toward Autonomic Networks": Knowledge Management and Self-Stabilization, pages 239-260. 2009.

[i.13] Ranganai Chaparadza: "Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behaviour Specifications for Diverse Networking Environments". International Engineering Consortium (IEC), Annual Review of Communications, 61, 2008.

[i.14] Ranganai Chaparadza. UniFAFF: "A Unified Framework for Implementing Autonomic Fault-Management and Failure-Detection for Self-Managing Networks. International Journal of Network Management", 19(4):271-290, July 2009.

[i.15] Ranganai Chaparadza, Symeon Papavassiliou, Timotheos Kastrinogiannis, Martin Vigoureux, Emmanuel Dotaro, Alan Davy, Kevin Quinn, Michal Wódczak, Andras Toth, Athanassios Liakopoulos, and Mick Wilson: "Towards the Future Internet - A European Research Perspective", chapter "Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering", pages 313-324. IOS Press, 2009.

[i.16] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba: "A survey of network virtualization". Comput. Netw., 54(5):862-876, 2010.

[i.17] David D. Clark, Craig Partridge, and J. Christopher Ramming: "A knowledge plane for the Internet". In SIGCOMM, pages 3-10, 2003.

[i.18] Yixin Diao, Joseph L. Hellerstein, Sujay Parekh, Rean Griffith, Gail Kaiser and Dan Phung: "Self-Managing Systems: A Control Theory Foundation". In ECBS '05: Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, pages 441-448, Washington, DC, USA, 2005. IEEE Computer Society.

[i.19] EFIPSANS Consortium: "Second Draft of Autonomic Behaviours Specifications (ABs) for the Diverse Networking Environments". Project Deliverable - D1.3v1, December 2008. [Refer to the final D1.7].

[i.20] EFIPSANS Consortium: "Third Draft of Requirements Specifications (RQs) regarding the initially identified features, required in IPv6 protocols, network architectures and paradigms, in order to implement the specified Autonomic Behaviours". D1.6v1 Core Document, December 2009.

[i.21] Mark Felegyhazi and Jean-Pierre Hubaux: "Game Theory in Wireless Networks: A Tutorial". Technical report, 2006.

[i.22] Laura Ferrari, Antonio Manzalini, Corrado Moiso and Peter H. Deussen: "Highly Distributed Supervision for Autonomic Networks and Services". In AICT '09: Proceedings of the 2009 Fifth Advanced International Conference on Telecommunications, pages 111-116, Washington, DC, USA, 2009. IEEE Computer Society.

[i.23] Paul Francis. Deep-4D: "Reality-based Network Management". Self-Managing Networks Summit 2005, June 2005. Microsoft Research.

[i.24] F. Fu and M. van der Schaar: "A New Systematic Framework for Autonomous Cross-Layer Optimization. Vehicular Technology", IEEE Transactions on, 58(4):1887-1903, may 2009.

[i.25] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan and Hui Zhang: "A Clean Slate 4D Approach to Network Control and Management. SIGCOMM Computer Communication Review", 35(5):41-54, 2005.

[i.26] Recommendation ITU-T Z.100 (November 2007): "Specification and Description Language (SDL)".

[i.27] Márk Jelasity, Alberto Montresor and Ozalp Babaoglu: "Gossip-based aggregation in large dynamic networks". ACM Trans. Comput. Syst., 23(3):219-252, 2005.

[i.28] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M.O. Foghlu, W. Donnelly and J. Strassner: "Towards autonomic management of communications networks". Communications Magazine, IEEE, 45(10):112-121, October 2007.

[i.29] Tao Jiang and J.S. Baras: "Fundamental Tradeoffs and Constrained Coalitional Games in Autonomic Wireless Networks". In Modelling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007. WiOpt 2007. 5th International Symposium on, pages 1 -8, 16-20 2007.

[i.30] Void.

[i.31] P. N. Johnson-Laird: "The computer and the mind : an introduction to cognitive science". Harvard University Press, Cambridge, Mass, 1988.

[i.32] Dan Marinescu and Reinhold Kröger: "State of the art in autonomic computing and virtualization". Technical report, Distributed Systems Lab, Wiesbaden, 2007.

[i.33] A. Hamed Mohsenian-Rad, Jianwei Huang, Mung Chiang and Vincent W. S. Wong: "Utility-optimal random access: reduced complexity, fast convergence, and robust performance". Trans. Wireless. Comm., 8(2):898-911, 2009.

[i.34] Richard Mortier and Emre Kiciman: "Autonomic Network Management: Some Pragmatic Considerations". In INM '06: Proceedings of the 2006 SIGCOMM workshop on Internet network management, pages 89-93, New York, NY, USA, 2006. ACM.

[i.35] Arun Prakash, Ranganai Chaparadza and Zoltan Theiz: "Requirements of a Model-Driven Methodology and Tool-Chain for the Design and Verification of Hierarchical Controllers of an Autonomic Network". In MOPAS '10: Proceedings of the 1st International Conference on Models and Ontology-based Design of Protocols, June 2010.

[i.36] Stephen Quirolgico, Kevin Mills, and Doug Montgomery: "Deriving Knowledge for the Knowledge Plane". Draft from National Institute of Standards and Technology Advanced Network Technologies Division Gaithersburg, June 2003. MD 20899-8920.

[i.37] Walid Saad, Zhu Han, Mérouane Debbah, Are Hjørungnes and Tamer Basar: "Coalitional Game Theory for Communication Networks: A Tutorial". CoRR, abs/0905.4057, 2009.

[i.38] John C Strassner, Nazim Agoulmine and Elyes Lehtihet: "FOCALE - A Novel Autonomic Networking Architecture". In Latin American Autonomic Computing Symposium (LAACS), Campo Grande, MS, Brazil, 2006.

[i.39] Nikolay Tcholtchev, Ranganai Chaparadza and Arun Prakash: "Addressing Stability of Control-Loops in the Context of the GANA Architecture: Synchronization of Actions and Policies". In IWSOS '09: Proceedings of the 4th IFIP TC 6 International Workshop on Self-Organizing Systems, pages 262-268, Berlin, Heidelberg, 2009. Springer-Verlag.

[i.40] Mandayam A.L. Thathachar and P.S. Sastry: "Networks of learning automata: techniques for online stochastic optimization". Kluwer Academic, 2004.

[i.41] Jeroen van der Ham, Paola Grosso, Ronald van der Pol, Andree Toonk and Cees de Laat: "Using the Network Description Language in Optical Networks". In Tenth IFIP/IEEE Symposium on Integrated Network Management, May 2007.

[i.42] Void.

[i.43] EFIPSANS Consortium: "Final Draft of Autonomic Behaviours Specifications (ABs) for the Diverse Networking Environments". Project Deliverable - D1.7 core (3 parts in total), December 2010.

NOTE: Available at http://www.efipsans.org/dmdocuments/INFSO-ICT-215549_EFIPSANS_CO_WP1_D1_7_Part1.pdf.

[i.44] Gabor Retvari, Felician Nemeth, Ranganai Chaparadza and Robert Szabo: "OSPF for Implementing Self-adaptive Routing in Autonomic Networks: a Case Study". In proceedings of the 4th IEEE International Workshop on Modelling Autonomic Communication Environments (MACE 2009), October 26-27 2009, Venice, Italy.

[i.45] Void.

[i.46] GENI Initiative.

NOTE: Available at http://www.geni.net/.

[i.47] R.Brennan, D.Lewis, J.Keeney, Z.Etzioni, K.Feeney, D.O'Sullivan, J.Lozano and B.Jennings: "Policy-based integration of multiprovider digital home services". Network, IEEE, vol.23, no.6, pp.50-56, November-December 2009.

[i.48] M. Serrano, S. van Der Meer, V. Holum, J. Murphy and J. Strassner: "Federation, a Matter of Autonomic Management in the Future Internet", in Proc. IEEE Network Operations and Management Symposium (NOMS), 2010, pp 845-849.

[i.49] T.Magedanz, F.Schreiner and S.Wahle: "Service-oriented testbed infrastructures and cross-domain federation for Future Internet research", in Proc. Integrated Network Management-Workshops, 2009. IM '09. IFIP/IEEE International Symposium on, vol., no., pp.101-106, 1-5 June 2009.

[i.50] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama and Scott Shenker: "Onix: A Distributed Control Platform for Large-scale Production Networks". 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10), October 4-6, 2010, Vancouver, BC, Canada.

[i.51] ETSI GS AFI 001: "Autonomic network engineering for the self-managing Future Internet (AFI); Scenarios, Use Cases and Requirements for Autonomic/Self-Managing Future Internet".

[i.52] Mitola J Cognitive Radio Architecture, Chapter 7 in "Cognitive Networks- Towards Self-Aware Networks" Wiley London 2007, Mahmoud Q. H. ed. in [i.53].

[i.53] Mahmoud Q. H. ed.: "Cognitive Networks- Towards Self-Aware Networks" Wiley London 2007.

[i.54] Kramer J. and McGee J.: "Self Managed Systems - An architectural Challenge" FOSE '07 2007 Future of Software Engineering.

[i.55] Friend D. H. et al.: "Distributed Reasoning and Learning in Cognitive Networks-Methods and Design Decisions", Chapter 9 in "Cognitive Networks- Towards Self-Aware Networks" Wiley London 2007, Mahmoud Q. H. ed. in [i.53].

[i.56] Gat E.: "On Three Layer Architectures" Proc. Artificial Intelligence and Mobile Robots, 1998 MIT Press.

[i.57] Thomas R.W., DaSilva L.A. and MacKenzie A.B: "Cognitive Networks" New Frontiers in Dynamic Spectrum Access Networks DySPAN 2005.

[i.58] Copyright of the original diagram that was modified belongs to Taylor and Fransis Group, LLC, Auerbach Publications: Book "Advances in Network Management by Jianguo Ding: ISBN 978-1-4200-6452-0.

[i.59] A. Mihailovic, I. Chochliouros, A. Kousaridas, G. Nguengang, C. Polychronopoulos, J. Borgel, M. Israel, V. Conan, M. Belesioti, E. Sfakianakis, G. Agapiou, H. Aghvami and N. Alonistioti: "Architectural Principles for Synergy of Self-management and Future Internet Evolution", Proceedings of ICT Mobile Summit, June 2009.

[i.60] Nikolay Tcholtchev, Monika Grajzer, Bruno Vidalenc: "Towards a unified architecture for resilience, survivability and autonomic fault-management for self-managing networks". In ICSOC/ServiceWave'09 Proceedings of the 2009 international conference on Service-oriented computing, Stockhom, Sweden.

[i.61] EC funded FP7 EFIPSANS Deliverable: INFSO-ICT-215549-EFIPSANS-WP4-D4.5.pdf: "Components and Mechanisms for Autonomic Fault Management".

NOTE: The deliverable will soon be available for download from this site (http://www.efipsans.org/).

[i.62] EC funded FP7 EFIPSANS Deliverable: INFSO-ICT-215549-EFIPSANS-WP3-D3.4.pdf: "Resilience, Survivability and/for Autonomicity in IPv6 Networks".

NOTE: The deliverable will soon be available for download from this site (http://www.efipsans.org/).

[i.63] Chowdhury, N.M.M.K. and Boutaba, R.: "Network virtualization: state of the art and research challenges," Communications Magazine, IEEE, vol.47, no.7, pp.20-26, July 2009.

[i.64] ISO/IEC 7498-1: "Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model".

[i.65] IETF RFC 3444: "On the Difference between Information Models and Data Models".

[i.66] IETF RFC 2474 (December 1998): "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", Nichols, K., Blake, S. Baker, F. and D. Black.

[i.67] IETF RFC 2710: "Multicast Listener Discovery (MLD) for IPv6".

[i.68] IETF RFC 3810: "Multicast Listener Discovery Version 2 (MLDv2) for IPv6".

[i.69] OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1.

NOTE: Available at http://www.omg.org/spec/MOF/2.4.1.

[i.70] Recommendation ITU-T X.200 (1994) | ISO/IEC 7498-1:1994: "Information Technology -- Open Systems Interconnection -- Basic Reference Model".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Additional Note: example of an autonomic behaviour is: self-description and self-advertisement, self-healing, self-configuration, all triggered by a DE

NOTE: Therefore, it is important to note that an autonomic behaviour is bound to a DE, and possibly (though not necessarily) to information supply parts of the Control-Loop implemented by the DE together with the Managed Entity (or Entities) under the control of the DE.

Autonomic Behaviour (AB): in GANA, an Autonomic Behaviour(AB) is defined as a behaviour or action that may consist of a set of sub-behaviours or sub-actions triggered by a Decision-Making-Element (DME or a DE-in short) in an attempt to achieve the goal defined by how the Decision-Making-Element manages a Managed Entity (or Entities) - ME(s) under its control in a Control-Loop Structure

NOTE: An AB is an observable and a verifiable behaviour on interfaces of an autonomic manager element (i.e. a Decision Element). The autonomic behaviour is considered as a behaviour of a DE, triggered as a result of reception of information from its information suppliers such as its associated Managed Entity (ies) in an attempt to regulate or (re-)configure the behaviour of the Managed Entity (/Entities), OR starts as a behaviour spontaneously triggered by the DE. A behaviour triggered spontaneously by a DE is simply a spontaneous transition in the Finite-State-Machine describing the overall behaviours of the DE.

Autonomic Manager Element: functional entity that drives a control-loop to configure and adapt (i.e. regulate) the behaviour of a managed resource.

NOTE: An Autonomic Manager Element can configure and adapt a managed resource like a protocol module or some other type of a managed entity such as a component, by processing sensory information from the managed resource and from other types of primarily required information sources and reacting to observed conditions by effecting a change in the behaviour of the managed resource to achieve some goal.

Autonomic systems with Cognitive capabilities: systems, which determine their behaviour, in a reactive or proactive manner, based on the external stimuli (environment aspects), as well as the goals they are required to fulfil, principles of operation, capabilities, experience and knowledge.

NOTE: In the case of telecommunications networks, this definition means that a cognitive system has the ability to dynamically select the network's configuration, through self-management functionality that reaches optimal decisions, taking into account the context of operation (environment requirements and characteristics), goals and policies (corresponding to principles of operation), profiles (corresponding to capabilities i.e. functional features supported), and machine learning (for managing and exploiting knowledge and experience.

cognition: combination of learning and reasoning as defined by Clark, Partridge, Ramming & Wroclawski in [i.17]

data: definition taken from Wikipedia: Data are values of qualitative or quantitative variables, belonging to a set of items

NOTE 1: Data in computing (or data processing) are often represented by a combination of items organized in rows and multiple variables organized in columns. Data are typically the results of measurements and can be visualised using graphs or images. Data as abstract concept can be viewed as the lowest level of abstraction from which information and then knowledge are derived. This lowest level named by the well accepted term of Raw data, i.e. unprocessed data, refers to a collection of numbers, characters and is a relative term; data processing commonly occurs by stages, and the "processed data" from one stage may be considered the "raw data" of the next.

NOTE 2: Data in computing (or data processing) are often represented by a combination of items organized in rows and multiple variables organized in columns. Data are typically the results of measurements and can be visualised using graphs or images.

Data Model: definition and format of data, including data-types and values for the purposes of storing or communicating the data from one entity to another

NOTE: A Data Model models data. The definition of Data Model adopted in the present document is the same as known in IETF, TMF, ITU-T, where there are a number of Data Models defined such as SNMP's SMI definiftions of MIB modules, SID (Shared Information Data) for TMF, IRP (Integration Reference Point) for 3GPP, CMIP's Management Objects definitions, and other types of Data Models. RFC 3444 [i.65] gives information On the Difference between Information Models and Data Models.

Decision Element: Decision-Making-Element (DME) referred to in short as Decision Element (DE) is a functional entity that fulfils the role of an Autonomic Manager Element

NOTE: Each DE is designed and assigned to autonomically manage and control some Managed Entities (MEs). MEs and their associated configurable parameters are assigned to be managed and controlled by a concrete DE such that an ME Parameter is mapped to "one DE". An ME is a protocol or a mechanism implemented by some functional entity. A Decision Element (DE) is an "Autonomic Manager Element" that implements the logic that drives a control-loop over the "management interfaces" of its assigned Managed Entities (MEs). Therefore, self-* functionalities are the functionalities implemented by the Decision Element(s).

Decision Plane: real made up of autonomic manager elements called Decision-Elements (DEs), and makes all decisions driving a node's behaviour (including the behaviour of all managed entities of the node) and network-wide control, including reachability, load balancing, access control, security, and interface configuration

NOTE: Replacing today's Management Plane, the decision plane operates in real time on a network-wide view of the topology, the traffic, events, context and context changes, network objectives/goals/policies, and the capabilities and resource limitations of the nodes and devices of a network of some scope (**Definition adopted but with modification, from the 4D architecture**). The **GANA Decision Plane** encapsulates today's Vertical **Management Plane** and replaces it in the long term, and adds the Horizontal view of the Decision Plane to allow distributed DE-to-DE interactions for network-intrinsic management (for those aspects requiring network-intrinsic management i.e. distributed control-loops).

Generic Autonomic Network Architecture (GANA): Conceptual Architectural Reference Model for Autonomic Network Engineering, Cognition and Self-Management

NOTE: Its purpose is to serve as a "blueprint model" that prescribes the design and operational principles of "autonomic decision-making manager elements" responsible for performing "autonomic" and "cognitive" management and control of resources (i.e. adaptively). It is not an implementation architecture per-se. Its elaborated definition and specification is a subject of this present document.

Holistic: property of a Reference Model, such as the one defined in the present document, as being "holistic", stems from the following aspects: in autonomic computing and networking models developed in the past, the abstraction levels at which to place control-loops and distinctions between "fast-control loops" and "slow control-loops", have often not been fully defined as some models were limited to assuming and considering only control-loops outside of Network Elements, yet a holistic model defines all the viable interworking abstractions at which control-loops can be designed

NOTE 1: Some models (reviewed and now unified within a single model in the Reference Model defined in the present document), initially introduced autonomics in the outer loops while assuming Network Elements to be managed "dumb" devices, due to the assumption that was taken, of limited processing power and resources that could be available in Network Elements. A more complete model is now one that also further defines lower levels of abstractions (and nesting) at which control-loops may be introduced into the Network Elements themselves since the limitation of resources in Network Elements does not necessarily hold any longer. Meaning that, where distributed control-loops encompassing multiple network elements could be considered while at the same time accommodating interworking outer "logically centralized" loops, the Reference Model presents a "holistic" view of the different possibilities as a "hybrid model". The model should combine "centralization of some Decision-Making Capabilities" while allowing also some "distributed Decision-Making Capabilities" of the network as well as some degree of self-management to be exercised by individual network elements at the lower abstraction levels of self-management. Four basic levels of abstractions for introducing control-loops are defined in the Reference Model.

NOTE 2: A more complete model is now one that also further defines lower levels of abstractions (and nesting) at which control-loops may be introduced into the Network Elements themselves since the limitation of resources in Network Elements does not necessarily hold any longer. Meaning that, where distributed control-loops encompassing multiple network elements could be considered while at the same time accommodating interworking outer "logically centralized" loops, the Reference Model presents a "holistic" view of the different possibilities as a "hybrid model". The model should combine "centralization of some Decision-Making Capabilities" while allowing also some "distributed Decision-Making Capabilities" of the network as well as some degree of self-management to be exercised by individual network elements at the lower abstraction levels of self-management.

information: Processed Data or a Model. The Model could be a result of processing data and representing it by following and respecting a Data Model, or the Model can be one that is created by humans e.g. an Information Model, a System-Model, etc. [i.36] simplifies things (than in other approaches) by simply using one term "Information" to inclusively describe both Data and Information, hence the so-called "Information-layer".

Information Model: defines concepts representing entities that require management information and data to be defined, as well as the relationships, constraints, rules, operations, and structure in relation to the concepts

NOTE: The definition of Information Model adopted in the present document is the same as known in IETF, TMF, DMTF, where there are a number of Information Models defined such SID and CIM. RFC 3444 [i.65] gives information On the Difference between Information Models and Data Models. Remark: The concepts and their relationships, defined by the Reference Model for Autonomic Networking, Cognition and Self-Management presented in the present document, should be used to extend/evolve the existing Information Models such as DEN-ng, SID, CIM, etc.

Interface(s) for Network Governance: it is the interface to the network, through which the human operator supplies to the network the following as input: Network Goals/Objectives, Policies, Profiles and Configuration Data that the autonomic network uses to self-configure and reconfigure in case of changes to the initial input

knowledge: information correlated according to a Model that is theoretically considered as a valid instance of a Meta-Space that defines the correlation among the abstract concepts of the Information Elements

NOTE: Why "theoretically"?, because it may be difficult to achieve an ideal case/situation whereby all the elements required of correlated information are instantiated, due to the fact that some data or information from which to derive Knowledge, may be corrupted, incomplete or invalid. Therefore, an attempt to derive knowledge may result in Partial or Incomplete knowledge. Therefore, as such, Knowledge can be represented by a Model and its associated Meta-Space of which the Model is an instance. The Model is completely a valid one if it conforms to the Meta-Space that describes valid instances of the Meta-Space. The Meta-Space can be a Meta-Model or could be some Ontology and associated Schemas and together with the associated models, is provided as input to the cognitive process that claims to be able to operate on the knowledge and do something with it e.g. make some decision(s). Cognitive processes may modify the Meta-Space and the elements of its instances (Models) during a learning process, by adding new "concepts" to the Meta-Space OR manipulating existing instances (Models) of the Meta-Space that are already known (i.e. are in the Knowledge Base) OR adding new instances (Models).

Knowledge Plane: Citing "Clark, Partridge, Ramming & Wroclawski, from whom the original idea of the Knowledge Plane came from" [i.17]: The Knowledge Plane, is a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network

NOTE: It is a distributed and decentralized construct within the Internet to gather, aggregate and act upon information about network behaviour and operation (citing "Stephen Quirolgico et al") [i.36]. The subject of the kind of functional entities (mainly GANA Network-Level-DEs) that realize the Knowledge Plane, is covered in clause 9.13 of the present document: "Cognition and Knowledge Plane as part of the GANA Decision Plane".

learning: ability that enables the system to gradually obtain knowledge on how to handle complex situations

NOTE: This can increase the speed of the decision making process, and also the degree of certainty w.r.t. the quality of the decisions.

Managed Entity (ME): is a protocol or a mechanism implemented by some functional entity that does a specific job it is designed to perform, and can be managed by an Autonomic Manager Element (i.e. a Decision Element) in terms of the ME's orchestration, configuration and (re-)configuration through parameter settings

Meta-Model: abstraction scope/level that is a collection of concepts (i.e. abstract representation models), their relationships and constraints among each other as they define a "domain"

NOTE: For example, a car can be considered as a domain (i.e. a composite model), that consists of constituent interrelated concepts such as a wheel, an engine, etc. The domain being defined could also be a "language" e.g. a high-level language such as UML, ITU-T SDL, ITU-T MSC, or C, C++, as being defined by the particular Meta-Model. A correct instance of the meta-model is governed by the instantiation of the individual constituent concepts belonging to the domain in their completeness and having valid and well formed complete links between the instantiated constituent elements. The domain, also viewed as a model, is called the "meta-level" of abstraction while different types of instances (e.g. different cars in the case of the "car domain"), are called instances or simply models. (Refer to the OMG's MDA architecture that defines four layers of models, the lower being an instance of the intermediate upper model, starting from M0 Models, through to M1, M2 and M3 levels of abstractions (also called Models)).

Meta-Space: can be considered as a Meta-Model

NOTE: A Meta-Space can be represented as a Meta-Model or could be some Ontology and associated Schemas, and together with the associated models, it is provided as input to the cognitive process that claims to be able to operate on the knowledge and do something with it e.g. make some decision(s).

ontology: defines concepts, their relationships and constraints

NOTE: In contrast to a Meta-Model, which also defines concepts, their relationships and constraints, an ontology includes both the "meta-aspects" i.e. concepts, relationships and constraints, and also their actual instances if any have been created.

reference model: Blueprint Specification that defines the Concepts, Abstractions, Concept-Relationships and Principles of a "domain" e.g. autonomic networking domain, as well as the Functional Blocks (FBs) and Reference Points/Interfaces between FBs, in such a generic way that Implementation-Oriented Details are left out (because there may be diverse ways of implementing the prescribed concepts)

Reference Point(s): "logical interface" between at least two Functional Blocks (which include the traditionally so-called Function Blocks), that indicates that the associated Functional Blocks (FBs) communicate with each other, as peers, towards some goal

NOTE: This term come from ITU-T SG15 and defined for SDH Functional Architecture where Atomic Function could be mapped to FB in this AFI specification. The information (messages and data) communicated by the associated FBs is called "Characteristic Information" i.e. it is information that characterizes what is communicated between the FBs. Since a Reference Point defines a logical interface in a way that is agnostic to the actual physical means/channel by which the communicating FBs are connected to facilitate their communication, a protocol (usually) or some other type of mechanism can be designed to be the "vehicle" that conveys the "characteristic information" between the FBs.

self-awareness: represents the knowledge building process as a continuous necessity in self-managing systems

NOTE: Knowledge building is a continuous process where the awareness is seen as conclusions derived by the system on being present in a particular operational state or status related to a particular operational state at a given time. This method is not directly linked to triggering executions but rather to assessing the operational state or status, i.e. gathering knowledge. Hence, the execution triggering can be implicitly related to this method.

self-configuration: automated configuration of components and adaptation of the system to dynamically changing environment conditions

NOTE: The ability of the system to accommodate new operational aspects in terms of the network elements, hardware, software, functional improvements and services that have been provisioned by the operator. The essence of this method is in having the ability to add and accommodate new functional components. It could be overlapping with self-optimisation as the impact of the components on the operational aspects might be subject to the evaluation of the optimality points but this is not the key factor in viewing the ability of the system to self-configure.

self-healing: it encompasses processes for problem discovery through fault-detection, diagnosis and triggering appropriate actions to prevent disruptions

NOTE: The ability of the system to respond to unplanned events, such as failures, requiring corrective actions and restore or improve the operational aspects of the system accordingly. This method is quite diverse in terms of the targeted operational aspects that are affected, as it is relevant to the degree of the "healing" required. Self-healing can be perceived as a reactive property of self-management systems to events like failures.

self-management: process by which computer systems and other communication systems in general can manage their own operation without human intervention

NOTE 1: Self-management includes the functionality required for self-configuration, self-optimisation, self-healing and self-protection.

NOTE 2: The term self-management is a general term (i.e. it is the "big-picture") describing all properties of a system falling under the umbrella of autonomic and cognition-based operations for performing the relevant operational aspects of the system. Hence, under the umbrella of self-management there are distinct methods of self-management with specific realisations and purposes in the system. The six distinct self-management methods are the following and depicted on figure 6. For more details, refer to [i.59] from which the definitions below were adopted.

1) Self-optimisation
2) Self-configuration
3) Self-healing
4) Self-protection
5) Self-awareness
6) Self-organisation

Figure 6: The six distinct self-management methods (from [i.59])

self-optimisation: process by which a component or system continuously tries to tune resources and balance workloads in order to maximize the use of resources and improve its own performance and efficiency

NOTE: Self-optimisation may exploit optimisation techniques to overcome/solve problematic situations. It is therefore considered as the ability of a component/system to perform adjustments of its operations for achieving the targeted optimality point in terms of the relevant performance metrics for the given event. Optimality point is introduced as the broad term that includes a variety of parameters all subject to particular types of events in the component/system that is being optimised and it is related to the evaluation criteria applied by the involved functionality of the component/system. Hence, self-optimisation can take three distinct objectives:

a) Maintaining system operations where the optimality point is defined as the collective or specific set of operational conditions and/or parameters before the ("disturbing") event triggering self-optimisation, expressed in the relevant performance metrics (systems goes back to as it was before the event triggering self-optimisation).

b) Improving the operational aspects for targeting the newly calculated optimality point, i.e. improving some aspects of the system's operation (system improves an aspect of the operation, better than before).

c) Maximising the possible operational aspects of the system to the newly calculated optimality point with inclusion of newly present constrains in the system.

self-organisation: specific method indicating ways of collaborations of network elements or clusters in the context of specific management functions

NOTE: Self-organisation builds on specific management principles: absence of centralized control, continual adaptation to a changing environment, autonomous interacting elements, simple rules for autonomic entities local interaction and interactions based on local knowledge.

self-protection: it encompasses processes for anticipation, detection, identification, and triggering appropriate protection mechanisms against threats

NOTE 1: The ability of the system to compensate for the effects of foreseen events or overcome them completely in terms of their impact on the operational aspects of the system. This ability of the system is in using the gathered knowledge for deducing the events in advance to their occurrence, and then proactively directing operations of the system. Emphasis is on the proactive nature of the system (assuming the knowledge has been appropriately gathered) and can also be generalised to detection of external attacks to the system for which the detection process may follow the similar pattern of gathering knowledge (but can be overlapping with self-healing in some cases).

NOTE 2: The above definition of self-management methods (from [i.59]) indicate that the six self-management methods vary in terms of the perspectives on how the systems invoke executions and relevance to the detection processes, i.e. use of the gathered knowledge. They also indicate a broad scope of disciplines that might be present in a self-managed and cognitive system in Future Internet networks, and variety of events that can trigger invocations of self-management processes. The self-management methods have different operational implications and relate to different properties of the systems, e.g. self-awareness is a background fallback process in any self-managed and cognitive system and stands as dedicated and continuous process of gathering knowledge as rendered system data collected via monitoring.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

4D 4D Architecture

NOTE: With 4 planes, namely: Decision, Data, Dissemination and Discovery planes.

A Assurance
AB Autonomic Behaviour

NOTE: See definition.

ABS Autonomic Behaviour Specification(s)
AF Autonomic Function
AGH Akademia Górniczno-Hutnicza im. S. Staszica w Krakowie: AGH University of Science and Technology, Poland
ANA Autonomic Network Architecture
API Application Programming Interface
AS Autonomous System
ASBR AS Border Router
ASF Action Synchronization Function(s)
B Billing
BBF BroadBand Forum
BFD Bidirectional Forwarding Detection
BGP Border Gateway Protocol
BML Business Management Layer
BUPT Beijing University of Posts and Telecommunications, China
CAPEX CAPital EXpenditure
CHOP Configuration Healing Optimization Protection

NOTE: In autonomics, Self-CHOP refers to these Self-* features: Self-Configuration, Self-Healing, Self-Protection, etc.

CIM Common Information Model
CLI Call Line Identification
CM Component Manager functionality in GENI

NOTE: See GENI initiative.

COM Component i.e. an object

NOTE: Refer to so-called COM objects in the COM programming model.

CONMan Complexity Oblivious Network Management
CPU Computer Processing Unit
DCOM Distributed Component

NOTE: I.e. an object, refer to so-called COM objects in the COM programming model.

DE Decision Element
DE-ME Decision Element - Managed Entity
DHCPv6++ Dynamic Host Configuration Protocol for IPv6

NOTE: Extended by the EC-funded EFIPSANS FP7 project.

DHT Distributed Hash Table
DME Decision Making Element
DMTF Distributed Management Task Force
DoS Denial of Service
DS Differentiated Services Field (DS Field)

NOTE: In the IPv4 and IPv6 Headers, see RFC 2474 [i.66].

EC European Commission
ECA Event-Condition-Action rule

NOTE: Refers to the ECA policy framework.

EFIPSANS Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services

NOTE: EC-funded FP7 project.

EML Element Management Layer
EMS Element Management System
EPC Evolved Packet Core
ETRI Electronics and Telecommunications Research Institute, S. Korea
F Fulfilment
FCAPS Fault, Configuration, Accounting, Performance and Security
FIB Forwarding Information Base
FOCALE Foundation-Observe-Compare-Act-Learn-rEason
GANA Generic Autonomic Network Architecture
GENI Global Environment for Network Innovations
GMPLS Generalised MPLS
GRNET Greek Research & Technology Network (GRNET), Greece
GUI Graphical User Interface
HLA High Level Architecture

NOTE: Refers to IEEE/DoD HLA standard, and multi-agent systems.

HTTP Hyper Text Transport Protocol
ICMPv6 Internet Control Message Protocol version 6
ID IDentifier
IETF Internet Engineering Task Force
IM Integrated Network Management

NOTE: An IFIP/IEEE International Symposium.

IMS IP Multimedia Subsystem
IP Internet Protocol
IPC Inter-Process Communication

NOTE: Refers to programming models in computer science.

IPFIX IP Flow Information Export
IPv4 Internet Protocol version 4
IPv6 Internet Protocol version 6
ISIS Integrated System to Integrated System
KPI Key Performance Indicators
LLA Logical Layered Architecture in TMN

NOTE: Refers to the TMN recommendation from ITU-T.

MA Management Agent

NOTE: Also refers to the CONMan framework.

MAP Mapping

NOTE: MAP is used in the present document as a terminology used in GANA.

MAPE Monitor the state/behaviour of managed resource, Analyze the monitoring data obtained, Plan a set of actions to perform in order to effect a change in the behaviour/state of the managed resource, Execute the set of actions at the appropriate time to adapt the resource(s)

NOTE: I.e. an autonomic behavior of an Autonomic Manager.

MAPE-K MAPE Knowledge
MAS Multi-Agent Systems
MB MegaByte
MBTS Model-Based Translation Service
ME Managed Entity
MIB Management Information Base
MIPv6 Mobile IPv6
MLD Multicast Listener Discovery

NOTE: Refers to RFC 2710 [i.67] and RFC 3810 [i.68] for MLDv1 and MLDv2 respectively.

MOF Meta-Object Facility

NOTE: Refers to OMG Meta Object Facility (MOF) Core Specification [i.69].

MPLS Multi-Protocol Label Switching
NCSR National Centre of Scientific Research "Demokritos", Greece
ND Neighbor Discovery protocol (IPv6)
NDL Network Description Language
NDP Neighbor Discovery Protocol

NOTE: Refer to IPv6 protocols.

NE Network Element
NET Network
NGN Next Generation Network
NIST National Institute of Standards (USA body)
NM Network Manager/Management

NOTE: Also refers to the CONMan framework.

NML Network Management Layer
NMS Network Management System
NOMS IEEE Network Operations and Management Symposium
NRDL Network Resource Description Language
NUM Network Utility Maximization
O Operations Support & Readiness
OA&M Operation Administration and Maintenance
ONIX Overlay Network for Information eXchange
OPEX OPerational Expenditure
OSI Open Systems Interconnection Reference Model

NOTE: See Recommendation ITU-T X.200 (1994) | ISO/IEC 7498-1:1994 [i.70].

OSPF Open Shortest Path First
OSPFv3 Open Shortest Path First version 3
OSS Operations Support System
PBM Policy Based Management
PC Personal Computer
QoS Quality of Service
RAM Random Access Memory
RAT Representation, Acquisition and Translation

NOTE: Concerns knowledge synthesis from raw data.

RDF Resource Description Framework
RF Radio Frequency

NOTE: Referring to RF tags.

RFC Request for Comments

NOTE: IETF type of standards.

RIB Routing Information Base
RIP Routing Information Protocol
RSVP Resource ReSerVation Protocol
SDL Specification and Description Language

NOTE: A language standardized by ITU-T.

SDN Software Driven/Defined Network
SID Shared Information and Data
SIXAN System for Information eXchange in Autonomic Networks

NOTE: It is synonymous with ONIX.

SLA Service Level Agreement
SMI Structure of Management Information
SML Service Management Layer
SNMP Simple Network Management Protocol
SON Self Organizing Network
SPL Simplified Policy Language

NOTE: Refering to CIM-SPL from the DMTF Group.

SW SoftWare
TCP Transfer Control Protocol

NOTE: Refers to TCP/IP model of the current internet.

TI Telecom Italia, Italy
TISPAN Telecommunications and Internet converged Services and Protocols for Advanced Networking
TMF Tele Management Forum
TMN Telecommunications Management Network
UDP User Datagram Protocol
UE User Equipment
UML Unified Modeling Language

NOTE: Standardized by OMG.

UPRC University of Piraeus Research Center, Greece
USA United States of America
VM Virtual Machine
WLAN Wireless Local Area Network
XEN Xen virtualization technique

NOTE: See Xen Hypervisors.

XML eXtensible Markup Language
XORP eXtensible Open Router Platform

NOTE: An exensible open source routing platform.

4 Definition of the AFI Architectural Reference Model of a Generic Autonomic Network Architecture and its fundamental requirements

4.1 Definition

The AFI Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management consists of the AFI-evolved GANA Reference Model that was introduced earlier in [i.13], [i.14]. The GANA Model is evolvable and has been evolved by AFI to address different aspects that were not included in the previous GANA descriptions e.g. Knowledge Plane and cognition, impact on evolution of management paradigms, reference points characterizations, enablers for autonomicity, etc. The evolution of the Reference Model is expected to continue, but as of now, it can already be used in extracting the design principles required for instantiating (applying) the model for the purposes of creating implementation-oriented architectures within which to implement the requirements for autonomicity and self-management operational behaviours of networks.

The bullet points below we define in a nutshell the key items/concepts that need to be distinguished, and discuss the purposes they serve. They also indicate the AFI Work Items working on the Specifications related to the specific items/concepts defined and distinguished below:

1) A Generic Autonomic Network Architecture (GANA) Reference Model for Autonomic Network Engineering, Cognition and Self-Management. As a conceptual model, its purpose is to serve as a "blueprint model" that prescribes design and operational principles of "autonomic decision-making manager components/elements" responsible for performing "autonomic" and "cognitive" management and adaptive control of resources. It is not an implementation architecture per se. [An elaborate definition of its Functional Blocks and Reference Points/Interfaces is given in the next clauses].

2) An Autonomicity-enabled Reference Architecture(s) - a result of "Instantiation" of the GANA Reference Model on a concrete target standardized element/device architecture, standardized network architecture and network environment. The process of Instantiating the Reference Model by mapping, fusion and superposition of the conceptual building blocks takes as input a "Standardized Reference Architecture" such as NGN architecture (ITU-T/ETSI-TISPAN), or 3GPP LTE/EPC architecture, or BBF (ADSL/FTTH) architecture, etc. As described below, an "Autonomicity-enabled implementable Architecture" that is a refinement of a particular "Autonomicity-enabled Reference Architecture" e.g. "Autonomicity-enabled NGN architecture", towards implementation-oriented solutions for Autonomicity and Self-Management, may be produced in a second step after the instantiation step. This is because not every aspect can be implementable in the short and mid-term. Firstly, AFI WI#3 focuses on a few types of targeted Reference Architectures, each of which will lead to the production of dedicated standalone document(s): "Autonomicity-enabled NGN architecture" (Fixed); "Autonomicity-enabled BBF (ADSL/FTTH) architecture"; "Autonomicity-enabled 3GPP and non-3GPP Mobile Network architectures"; "Autonomicity-enabled Ad-Hoc/Mesh/Sensor Wireless Network architectures".

4.2 The Generic Autonomic Network Architecture (GANA) Reference Model

GANA Reference Model is a Conceptual Architectural Reference Model for Autonomic Network Engineering, Cognition and Self-Management. Its purpose is to serve as a "blueprint model" that prescribes the design and operational principles of "autonomic decision-making manager components/elements" responsible for performing "autonomic" and "cognitive" management and control of resources. Here, the notion of "resources" includes protocols, stacks and mechanisms. In this context, the notion of "control" is behaviour associated with a combination of "observing, supervising, regulating, and dynamically adapting the behaviour of managed resources" i.e. something a controller from control-theory would do while realizing a control-loop. The aspect of "management" on the other hand, includes the traditional management plane and some aspects of "node/element-intrinsic" management as well as "network-intrinsic" management that does not involve the Network Management System role. The principles prescribed by the Reference Model are considered "generic" such that they can be applied and further refined during the design and implementation of autonomic decision-making manager components/elements for diverse implementation architectures and network environments.

Why "autonomic management and control"? The reason is to "automate management operations/functions while at the same time ensuring self-adaptation through interacting control-loop structures designed to operate at various levels of abstraction of functionality". The basic levels of abstractions are defined by the Model.

What is meant by being "Generic" in the "Generic Autonomic Network Architecture" (GANA) Reference Model:

The following bullet points summarize the properties of GANA as a "Generic Model" and what is required of it as Reference Model for Autonomic Networking, Cognition and Self-Management, starting with the main three properties, followed by what the Model defines and describes:

1) The Reference Model enables to produce Specification and Description Models (e.g. formal models such as SDL Models (see ITU-T Recommendation Z.100 [i.26] language) of the fundamental building blocks i.e. "autonomic manager components/elements" referred to as "Decision-making Elements" ("DEs" in short) and their Interfaces, that leave out the implementation-oriented details. Such formal models can be used in simulation and validations of the design models so as to detect and eliminate problems such as conflicting DE behaviours or potential stability related problems.

2) Fundamental Interfaces and associated Primitives/Operations of the DEs as defined by the Reference Model shall be generic to support different types of Data Models of data/information communicated on interfaces that are used later in the actual implementation.

3) The Decision Elements (DEs) that can be instantiated by design, for a particular network device/node, are decided upon by the context and role the device/node can play in the target network.

The GANA Model defines the following concepts:

1) The Model describes the Enablers for Autonomicity, Cognition and Self-Management, in a top-down approach, without being constrained by a specific implementation architecture, communication protocols and existing technologies, etc. Place-holders for Controls-Loops and Hierarchical and Horizontal relationships/interfaces between them are defined.

2) The concepts and fundamental building blocks ("autonomic manager components/elements"), referred to as Decision-making Elements ("DEs" in short), for holistically realizing autonomicity (control-loops and their required nested nature) and self-management at various levels within node/device architectures and an overall network architecture. A DE applies policies/goals and implements a control-loop over the "management interfaces" of its assigned Managed Entities (MEs) e.g. protocols, protocol stacks and mechanisms. This means a Table that maps DEs to their assigned types of MEs is provided as part of the Reference Model Core Specification, e.g. a Mobility-Management-DE autonomically manages and controls Mobility Protocols and Mechanisms. The table should define the assignment of specific types of Managed Entities (MEs) to specific DEs that autonomically manage and regulate the behaviour of the assigned MEs.

3) Characterization of the "Management Interfaces" of Managed Entities (MEs) i.e. basic primitives/operations to be supported as required by autonomicity and self-management.

4) Design Principles and Operational Principles (i.e. behavioural characteristics) of Decision Elements (DEs). The principles include:

   a) The nature and types of Control-Loops, their nested nature, time-scaling and associated types of DEs that drive specific control-loops.

   b) Hierarchical, Peer and Sibling Relationships of DEs.

   c) Levels of Self-Management and associated Hierarchical Control-Loop structures within the Vertical-View of the Decision Plane (also defined by the Model).

   d) The Horizontal-View of the Decision Plane and DE-to-DE peer interfaces for "network-intrinsic" management (in-network management), etc.

   e) Vertical and Horizontal Interfaces of DEs and their characterizations e.g. basic primitives/operations to be supported.

   f) Design and Operational Principles for DEs that enable the support for virtualization, domain-awareness, federation, etc.

5) An Information/Knowledge sharing system required in an autonomic network.

6) The Model combines Autonomicity and Cognition as part of the big-picture "Self-Management", within a single Unified Reference Model.

7) Knowledge Plane and its relation to the Decision Plane.

8) The Interfaces between the Network Operator and the Autonomic Network, i.e. Interfaces for Network Governance.

9) Reference Points between DEs and between DEs and other Functional Blocks introduced by autonomicity and self-management.

10) The Model ensures its Evolvability.

11) The Model is a Hierarchical Autonomic Management and Control Architectural Framework. It is a hybrid model that enables combining centralized and distributed decision-making based management and control of resources (which include monitoring functions and associated resources), while pointing out the main limitations of decentralization and distributed decision-making in network management and control, and providing techniques for addressing stability of control-loops in such as framework.

12) Mappings and implications to the TMN Logical Layered Architecture (LLA).

13) Reflection of the FCAPS Framework within the Model.

14) Other desired properties/requirements.

5 Properties of a Generic Autonomic Network Architecture Reference Model

Through analysis of the state-of-the art, the existing standards and current practices , the following have been identified as essential properties needed, foreseen or desirable for the systems and networks intended to apply the Generic Autonomic Network Architecture (GANA) Reference Model. The way they shall be realized in future networks and systems is left for further study and will be documented on in the work related to the Instantiation of the Model onto concrete implementation-oriented reference architectures:

- Property 1: Automation.
- Property 2: Awareness.
- Property 3: Adaptiveness.
- Property 5: Stability.
- Property 6: Scalability.
- Property 7: Robustness.
- Property 8: Security.
- Property 9: Switchable.
- Property 10: Federation.

An autonomic system should be able to detect, reconfigure and reregister its managed resources or managed devices such as router or user equipment even if it is mobile and allow session continuity with no disruption. An autonomic system should manage and control the mobility of an ambient system, in order to provide session continuity, local mobility decision should take into account the preferences, the capabilities, the objectives of the different players involved in session in order to identify a common decision able to provide session continuity. The mobility enabler will be used to retrieve the different monitoring process, security process, configuration process in order to decide and disseminate the mobility decision. A mobility enabler should be managed by different type of players. It will avoid today incoherence decision e.g. User Equipment (UE), Wireless Local Access Network (WLAN), 3GPP Mobile access network (e.g. UTRAN), core network(e.g. Home Agent) or application provider (e.g. Service Centralisation and Continuity Server AS-SCC). today each of them take local decision with no common knowledge.

6 Enabling concepts and mechanisms

Innovative aspects of the AFI Reference Model of a Generic Autonomic Network Architecture are: the dynamic and architecture-agnostic federation of intelligence; identification of the place-holders for control-loops design and the abstraction levels at which to introduce control-loops; its embodiment/instrumentation into the network and device architectures for optimum efficiency; intelligence and knowledge management for cognition in decision-making processes of the network; provision for advanced network Governance mechanisms. The Reference Model is expected to convince Service and Network Operators that autonomics becomes real when worked simultaneously from the two viewpoints - concept federation and network embedding. This framework will be materialized through behavioural requirements of autonomic blocks and standardized interfaces. It will ensure that these blocks contain the capabilities and mechanisms required to govern the integrated behaviour and operations of all networking building-blocks.

6.1 Information and knowledge management mechanisms

For further study in AFI.

Short description: all the mechanisms that allow capturing and exchanging information without manual intervention of the administrators, and taking into consideration dynamicity/variations of the system.

The aim of this procedure is to design mechanisms for information acquisition, learning and managing the associated knowledge, thus enabling an autonomic system to be self-aware. Specifically, the current mechanism offers a data exploitation of relevant information elements and the environment for information management in order to encompass sets of processes, interfaces, architectural and business requirements/guide-lines.

6.2 Cognitive mechanisms

For further study in AFI.

Short description: includes learning and reasoning; allows the system to self-describe; increases the self-awareness of the system; improves the decision-making process.

As "you go up the layers", the cognition mechanisms become more powerful. Degree of *Cognition* should increase as you go up from within a network element to the isolated overlay Knowledge Plane of the network. The embodiment of cognitive mechanisms in the system could be considered as an evolution towards "flat" network management.

The technological era is characterized by advances in the infrastructure, the applications/services supported, and the business model. These characteristics mean that the management of networks and algorithms should undergo evolutionary change too. Therefore, mechanisms for the embodiment of intelligence is required.

6.3 Service Models and Service Discovery Mechanisms

For further study in AFI.

Short description: Service Modelling is about the creation and representation of service models. Therefore, that autonomic elements of the network (i.e. Decision-making Elements), operating at certain levels within the vertical management framework, can refer and use the service models to understand the requirements of individual services such as QoE/QoS (Quality of Experience/Quality of Service) requirements, survivability requirements, etc. Hence, it allows to know how to dynamically provision and adapt the required resources so as to guarantee the requirements of individual services (e.g. availability, reliability, QoE/QoS). Service Discovery Mechanims play a role in the aspect of Autonomic Service Management presented later in the present document.

Added-value: It enables autonomic mechanisms of the network to know the requirements of individual services offered by the Services Stratum (e.g. in the NGN architecture), including QoE and QoS requirements, survivability requirements, etc, so that the Service Control Functions in the Services Stratum can adaptively control services in an autonomic fashion in collaboration with Transport Control Functions and associated Autonomic Functions in the Transport Stratum.

6.4 Network Governance Mechanisms

For further study in AFI.

Short description: The enabling notion of Governance is based on the fact that the autonomic network requires as input, the goals, objectives, profiles and policies defined by the Human Operator, in order for the network to operate in respect of the input provided. Therefore, the mechanisms should enable the Human Operator to define and validate policies, encapsulate policies and objectives and configuration data into network profiles describing services, and disseminate the profiles into the network. The other important principle of Network Governance is support for Profile Continuum and Policy Continuum.

The aim of this mechanism is to guarantee that the AFI Generic Autonomic Network Architecture Reference Model is able to achieve a manageable autonomicity in order to be able to guide network behaviour e.g. by way of facilitation for some degree of "human in the loop" interactions. Moreover, this procedure can be based on explicit policy management framework for guiding infrastructure and controlling the network entities. Both Policy- or Goal-based management apply.

6.5 Capability discovery mechanisms

For further study in AFI.

Nowadays, network technologies often run far ahead of the ability to manage them. This may result in complex networked systems requiring manual configuration and dedicated management provided by very expensive experts. Such a model is obviously cost ineffective and it cannot scale. One of the aspects that seems to be missing is any consideration of what is the cost of ownership of a given behaviour in the context of planning, fulfilling/configuring and assuring this behaviour in the end to end system. It is very easy to create a network feature that is bespoke and therefore difficult to integrate. From an OSS (Operating Support System) point of view the need would be to understand how the capabilities of network elements and network behaviours can be described in a common standardised way that makes them easy to integrate and describe to the overlying OSS processes. This involves the tasks of the self-description of the network elements, their self-discovery within the network and the given context(s), as well as the auto-discovery processes embodied in the network itself.

6.6 Embodiment Mechanisms

For further study in AFI.

In the Future Internet context there will be a constant evolution of networks, services and the ways of managing them, e.g. in terms of algorithms, interfaces, platforms. This fact necessitates the existence of embodiment mechanisms. The notion of embodiment refers to the capability of dynamically deploying and orchestrating management functionality, for instance related to context acquisition and reasoning, policies, profiles, knowledge development and sharing, algorithms for optimisation, negotiation and decision making.

Therefore, this enabler also includes Loading of Control Strategies, which can be achieved through loadable and Run-Time Executable Behavioural Models that can be loaded into the network such that the customized Behaviour Specification (executable at run-time i.e. interpretable and executable) can be used to guide the way a network element or the network as a whole should operate.

More details on how the enabler is taken into consideration are found in the technical description of the present document. This is also linked to Programmability as Enabler.

6.7 System Modelling

For further study in AFI.

Various methods exist for modelling the network which allow easier network description, definition and manipulation. This improved definition allows for the development of improved network management techniques.

Network Modelling can refer to a wide range of techniques including the ability to simulate networks using various software techniques such as Opnet [www.opnet.com], ns3 [www.nsnam.org], etc. More useful however, for the purposes of defining a Generic Autonomic Network Architecture, are network modelling techniques such as the description of the network into different functional components or layers as described in models such as the OSI model ISO/IEC 7498-1 [i.64] and the 4D model [i.25].

6.8 Modularity/Composability

For further study in AFI.

The management of complex systems strongly requires service composability in order to adapt to the resources and policies, provide for easy deployment but also easy update to tend towards the evolvability constraint. Components have thus to be designed according to modularity principles to enable the selection and assembling of components in various combinations to satisfy specific requirements. They are expected to be able to connect, interact, produce data or provide functionalities in a stateless manner or in a managed state by respecting standardized interfaces.

6.9 Platform-Independent Execution/Capability Model of a network element

For further study in AFI.

Each network device has different execution/(re)configuration capabilities that the available OSI layers support, and which could be used by an external self-management system for the individual adaptation of the behaviour of a network device or for the collective adaptation of a network domain. A common way to describe, to publish and to call an execution capability, in a platform agnostic manner, is required for the development of the adaptive property of an autonomic future Internet network system.

6.10 Cooperation Mechanisms

For further study in AFI.

In the context of an autonomic future Internet environment, network elements (homogeneous or heterogeneous) are called to address locally either a common or a contradictive goal (i.e. fault identification/optimization, cooperative routing/transmission etc). Well-specified interfaces that enable network devices collaboration for distributed monitoring and decision making processes are necessary. The formation of compartments among network devices could be used for various management problems, by extending individual network elements monitoring, situation awareness capability and decision making capacity through network elements collaboration. Standardized Interfaces that will allow the formation of compartments and the localized exchange of information among network elements is proposed. Cooperation mechanisms could support several characteristics/properties of the AFI Reference Model of a Generic Autonomic Network Architecture, as these are introduced in clause 5 of the present document (e.g. Automatic, Aware, Federative, Stable etc). Apart from that certain aspects of cooperative mechanisms will be provided to the AFI WI#3 Group Specification of Autonomic Ad hoc/Mesh/Sensor Networks.

6.11 (Cross-Layer) Monitoring Methods and Techniques

For further study in AFI.

An autonomic system, or entity of the system, may adapt its behaviour in response to changes in the system itself or in its environment. Monitoring is the process by which such change related information relevant to the self* properties of the system/entity is collected. This information is often designated as context and such a system is thus said to be context aware. Monitoring is performed by software or hardware elements called sensors. Monitoring may be passive i.e. "listening" or active i.e. specific actions may be taken to capture information. Monitoring information collected by an entity may be processed and propagated to other system entities. System wide montoring therefore entail the need for a monitoring infrastructure including data model, storage, processing and dissemination.

6.12 Fault-Detection, Fault-Diagnosis/Localization, Fault-Removal/Repair and Recovery Techniques and Mechanisms

For further study in AFI.

An autonomic application/system should be able to detect and recover from potential problems and continue to function smoothly. in terms of an uninterrupted service provision In order to self-heal a system, the autonomic system shall diagnose the potential failure in order to retrieve the cause and the location of a problem in the system and then infer the further implications from it to instantiate certain remediation actions. The diagnostics enabler is used to retrieve and select the different monitoring process, data processing functions and tests functions in order to retrieve the cause of a problem and disseminate it in an autonomic system.

6.13 Bootstrapping Mechanisms

For further study in AFI.

Bootstrapping is usually referred as a self-sustaining process that progresses without (or with limited) external help. In an autonomic networking environment, bootstrapping is related with the provision of initial configuration information to newly joined nodes or newly created networks, aiming to the successful initialization of the network mechanisms without any pre-configured data. Network-wide parameter estimation, data forwarding mechanisms, routing protocols, and security policies are some of the processes that need to be activated by the autonomic nodes without human support or pre-configured software. Decentralized bootstrapping, i.e. process instantiation without the need for special-purpose nodes, is also crucial in autonomic networks since it facilitates the design of functionalities and mechanisms in ad-hoc and resource constrained environments.

Bootstrapping may be realized in multiple phases. Initially, nodes in proximity (or neighbouring nodes) may establish point-to-point communication channels with each other. In a later stage, neighbouring nodes collaborate in order to establish end-to-end communication paths, e.g. by enabling proactive/reactive routing functionality. Service provisioning and fulfilment of any network-wide objectives are realized in later stages, e.g. by provisioning distributed repositories for data, information and knowledge sharing.

This is also related to capability discovery mechanisms.

6.14 Programmability

For further study in AFI.

Programmability, as an enabler refers to the provision of "Primitives/Operations" on the "Management Interfaces" of various types of Managed Entities (e.g. protocols, stacks and networking mechanisms) to enable Decision Logic that governs autonomic behaviour to "program" (i.e. start, pause, resume, terminate) the operation of a particular Managed Entity while at the same time supplying as input, parameter values, policies or behavioural specification, etc, required as parameters of the "Primitives/Operations". The Decision Logic (external to the Managed Entity) that governs autonomic behaviour may "re-program" a particular Managed Entity. In the technical specification of the Reference Model, models of various types of Managed Entities are presented as well as the notion of a "Management Interface" and "Primitives/Operations" of a Managed Entity through which programmability can be achieved by Decision Logic for autonomic control at a level higher than the Managed Entity.

7 Generic Autonomic Networking Architecture Reference Model

7.1 Principles & Motivations: Why it is required

The question for self-managing networks or autonomic networks is what is currently calling for a re-visitation of today's networking models, architectures and paradigms, thereby causing the networking research communities to be divided into two approaches: those who think of "network evolution" for self-manageability of networks versus those who that think of "network revolution/clean-slate" for self-manageability of networks. In evolutionary approaches we view Self-management, in its big picture; not only as "automation" through "scripting" but that it includes autonomicity (control-loops) as an enabler for achieving enriched and advanced self-manageability of nodes and networks by design. This can be achieved through instrumenting the nodes/devices with Autonomic Manager Components/Elements (referred in the present document as Decision-Making-Elements (DMEs) or simply Decision Elements (DEs)), which automate network operations by realizing (implementing) control loops. These control-loops operate on knowledge regarding events and state of network resources or functions, at different abstraction levels of functionality (defined later), and regulate the resources or functions of a system/network according to the goals of the network.

NOTE 1: AFI is taking an "evolutionary approach" to designing self-managing future networks while instantiating the GANA Model onto existing architectures, though the GANA Model can also be applied in designing Future Network Architectures that exhibit Self-Management Capabilities from the dawn of their design.

In the description of the sought Reference Model, the concept of DME is defined i.e. an autonomic manager element, as a concept that is associated with some concrete resource(s)/entity(ies) that the DME manages, and implements and drive its control loop based on its continuous learning cycle. Information or views continuously exposed by its managed resource(s)/entity(ies), together with information coming from other potentially required information suppliers of the DME, such as the environment in which the device hosting the DME is operating. Such information is used by the DME to change the behaviour of the managed resource(s)/ entity (entities) in order to achieve and maintain the goals known by the autonomic element (DME).

NOTE 2: In the description of the sought Reference Model, we adopt the concept of a Managed Entity (ME) to denote a managed resource or a managed automated-task in general, instead of a Managed Element-a term used in traditional network management terminology, in order to be more generic and to avoid the confusion that comes with the use of the term "Managed Element", which is normally associated with only meaning a physical Network Element (NE) and not some functional entity within a node/device such a protocol module or a component such as a monitoring component.

Below, a short review on the concepts of abstract autonomic system models and control loops is provided, starting with the well-known IBM-MAPE model [i.10], [i.38]. Later, clause 8.2.1, illustrates the nature of control loops that can be implemented by diverse functions of a node (micro-level), as well as by a particular networked system (node) or the network as a whole (macro-level). Past and recent autonomics related approaches suffer the following shortcoming when it comes to establishing evolvable architectures for the self-managing future networks. There have not been any attempts that start by holistically and generically specifying the required control loops for Autonomicity at micro-level and macro-level (including their interactions and relationships) for nodes and the network as a whole, including capturing the diversity of information suppliers that feed and drive a control-loop.

Because the area of autonomic networking is still evolving, all the current autonomics related projects/initiatives address different issues of relevance to the field of autonomics. What is currently missing is a generic architectural Reference Model of an autonomic networked system tailored to the broader domain of autonomic network engineering in particular, not to the domain of autonomic computing of which IBM is the one that first established an abstract Reference Model.

This clause points out the issues that are desirable to have been considered first before designing network architectures that are claimed to be autonomic, namely the creation of a Generic Reference Model of an autonomic network architecture that captures in a holistic way, such issues as the nature/architecture, multiplicity and diversity of autonomic manager elements (Decision-Making-Elements (DMEs)) of a node and a network as a whole; the nature, multiplicity and diversity of control-loops of DMEs, the Managed Entities(MEs) of nodes and a network, whose behaviours are managed/controlled/regulated by an associated DME, the relationships and interactions between DMEs within a single device or in a network. As will be clearly seen later in this clause and next clauses, the DMEs of a network considered autonomic self-managing should form an Architectural Reference Model for Autonomic Networking, Cognition and Self-Management. It is referred to as the GANA Model.

7.2 A review of today's best known approaches to Autonomic Networking

This clause provides a brief review of some of the best known approaches in the field of autonomic networking. The aim is not to describe each approach exhaustively but rather to summarize what is available in state-of-art and has been considered in the development of the GANA Model as a unifying framework for harmonized concepts from the different approaches, which are accommodated within the GANA Model. Besides, clause 9.12 (GANA as a Unifying Model) contains a table describing how concepts from the different viable best known approaches to autonomic networking have been selectively combined in a harmonized way or accommodated within a single unifying model referred to as the GANA Reference Model.

A review of today's well-known approaches to autonomic management and communication such as the IBM-MAPE Model [i.10], [i.38], 4D architecture [i.25], CONMan management model [i.8], FOCALE [i.38], [i.28], Knowledge Plane for the Internet [i.17], etc., shows that none of these well known approaches proposes a holistic Reference Model that defines and distinguishes between diverse Autonomic Elements/Managers and their associated Managed-Entities (MEs) and Control-Loops for different levels of abstractions within node/device architectures and network architecture as a whole. What is required is a Generic Autonomic Network Architecture Model that **defines the "Levels of abstractions of functionality"** at which to introduce **Autonomic Elements/Managers** in a **holistic way,** and accommodates and unifies concepts from today's well-known approaches to autonomic management and communication. This is because, each of these approaches presents concepts that can be combined in a harmonious way with concepts from the other approaches, thereby creating a single unified holistic model that can be applied in reasoning and designing autonomic components, their interfaces and relations. This has been one of the main goals behind the creation of the GANA Model from its very early stages when it was first introduced in [i.13] and also in [i.15]. The need for nested Hierarchical Control-Loops within node/device architectures and the network architecture as a whole, are well proven as described in the specification and description of the GANA Model and other sources in literature and cited in the GANA description-as given in more details in the subsequent clauses of the present document.

Most evolutionary approaches such as FOCALE [i.38] and [i.28] do not define a holistic framework that defines multiple levels where autonomic elements can be introduced, but settle only on a single distributed control loop outside of network elements/nodes without defining a framework of control loops and their interactions down to within individual autonomic node architectures. ANA [i.3] is a clean-slate project that focuses mainly on a framework for flexible protocol stacks and its architecture was not driven by a holistic capturing and specification of control loops and their interactions within individual autonomic nodes and the network as a whole, as a process initially required towards designing or deriving a suitable autonomic network architecture.

That is, the ANA architecture is not derived from a generic autonomic network architecture (because none has ever existed), and does not even attempt to look at how network heterogeneity and role-based systems design should influence the design of autonomic elements (DMEs) in nodes and the network as a whole. Haggle [i.4], like ANA, is an experimental clean-slate project focusing on issues such as eliminating layering above the data-link layer. Both 4D [i.25] and CONMan [i.8] were not designed to cater for autonomic functionality that may need to be designed for finer level of abstraction of network functionality within a node, with the possibility of having hierarchical control loops for autonomicity within a node - one of the principles GANA advocates for. GANA is considered generic to relax from the assumption of limited resource capabilities of devices/nodes, while at the same time allowing for the specification of the interfaces for which issues that require centralized decision making for the network such as in the case of 4D and CONMan can also be captured and specified. This means that some aspects of the 4D and CONMan should be taken into consideration in the design of the GANA architecture and its evolvability. Therefore, as described in [i.19] some of the concepts from today's approaches, for the Internet, as well as other approaches, can be inherited into the required GANA architectural reference model.

Therefore, to summarize: the evolving GANA Model **"Unifies" within a single holistic framework**, key concepts from approaches such as the IBM-MAPE Model, 4D architecture, CONMan management model, FOCALE, Knowledge Plane for the Internet, and GENI (as summarized in table 2 in clause 9.12. This table describes how concepts from the different viable approaches to autonomic networking have been selectively combined in a harmonized way or accommodated within a single unifying model referred to as the GANA Reference Model. More related details can be found in the GANA specification and description itself). Apart from accommodating and unifying concepts from the diverse approaches, GANA as a Hierarchical Autonomic Management and Control Architectural Framework, goes further to specify the following aspects (to mention a few, more aspects can be found in the GANA specification and description itself):

- How the framework, as a hybrid model, enables to combine Centralized and Distributed Decision-Making based Management and Control of Resources while pointing out the main limitations of decentralization and distributed decision-making in network management and control.

- The Hierarchical, Peering & Sibling Relations between autonomic manager components (referred to as "Decision Elements" (DEs) in the GANA Model) and the purposes of such relations.
- Methods and techniques for addressing Stability of Control-Loops in Hierarchical Autonomic Management and Control Architectural Frameworks such as GANA: i.e. Stability in GANA.
- Reference Points necessitated by the autonomic manager components and Information/Knowledge Sharing components.
- How cross-layer information may need to flow to all Decision Elements (DEs) at the Function-Level (Level-2 in GANA).
- How to incorporate Cognition in Decision Elements (DEs) and Cognition Levels in the GANA Decision Plane Hierarchy.
- Knowledge Plane as part of the GANA Decision Plane.
- How to accommodate Virtualization techniques.
- Mappings to existing management frameworks such as TMN-LLA (Logical Layer Architecture) and FCAPS framework, and how service management and network management as well as associated systems such as OSS's would need to evolve as necessitated by the GANA framework.
- Federation in GANA.
- Decision Notification in GANA for the "Human in the Loop" towards building Trust and Confidence in Autonomic behaviours.

7.3 Why a holistic dimension for Architectural Reference Model

Whether an evolutionary approach or revolutionary approach is taken towards designing future networks, there is a requirement for a holistic Generic Autonomic Network Architecture (GANA) that allows "standardizable" specifications of Self-* functions for diverse networking environments to be produced. These Self-* functions shall be testable and verifiable by using various test methods that help determistically infer the correctness of the functions. Moreover, such a Generic Autonomic Network Architecture shall allow to talk about self-managing/autonomic properties of network nodes and networks as a whole, at different levels of abstracted networking and system functions. It should also allow to reason about hierarchical levels of control loops and their associated Decision-making-Elements (DEs) for self-manageability/autonomicity. This should include peering relationships between Decision-Making Elements (DEs) that determine the autonomicity of a node(s) and also allow self-managing/autonomic behaviours to be achieved by the autonomic nodes in a distributed fashion.

By separating issues of concern for different, well-defined levels of autonomic decision-making processes, i.e. control loops, the GANA architecture would allow the production of standardizable specifications of autonomic behaviours including their coupling and interactions. Input to producing specifications of GANA autonomic behaviours for diverse networking environments should come from what has been achieved in isolated cases of autonomic networking for which information is currently scattered in scientific conferences, workshops, and journal papers, as well as in results obtained from some relevant research projects such as FOCALE, ANA, Haggle, 4D, CONMan, etc. as described in clause 9.12 where a summary Table is presented.

One of the key benefits of having a Conceptual Model like GANA that enables to separate specification and description aspects for design models from their implementation-oriented aspects, is to enable the production of generic specifications of autonomic elements and behaviours with some assumptions that the imaginary (i.e. *virtual*) autonomic nodes and the targeted network environments in which the autonomic nodes should operate will have enough implementation/run-time resources to satisfy the need of the specified (simulated and validated) autonomic elements. The roadmap from an imaginary (i.e. *virtual*) autonomic node captured by its autonomic behaviour specifications to its implementation, which then takes into account the required resources, the need to merge some of the components and interfaces from the specifications into some monolithic implementation or a modular implementation that strictly follows the specifications, all belong to the implementation issues and thus implementation issues should be left out from the generic specifications of the autonomic elements, interfaces and behaviours.

7.3.1 GANA properties

The GANA Architectural Reference Model is required to have the following properties:

- To be "**generic**": meaning that specification and description models of the key building blocks (DMEs) and Interfaces leave out the implementation-oriented details. Fundamental interfaces and operations of the DMEs shall be generic to support different types of data models used later in the actual implementation. The DMEs that can be instantiated by design, for a particular network device/node, are decided upon by the context and role the device/node can play in the network. There are some behaviours of DMEs that may be standardized to ensure interoperability, e.g. behaviours pertaining to what DMEs communicate on their interfaces (reference points) with other functional blocks they are required to communicate with.

- To be seen as a commonly shared reference model from which further architectural refinements for either clean-slate based realization of GANA **or** an evolutionary way of realizing the GANA can be achieved through the evolution of today's networking paradigms and protocols like IPv6. The aspect of being "commonly shared" also addresses the need for the concepts of the reference model to be agnostic to any specific technology or implementation issues.

- To be seen as a unified reference model in the sense that the types of functional entities (the so-called GANA Decision-Making-Elements) that govern the autonomic behaviours (i.e. Self-* features) of a node/network should be captured by the model, including their relationships and their interfaces - a result of consolidating different concepts from relevant approaches to autonomic networking, and harmonizing the understandings and even terminology into one single unifying framework.

- To be seen as a holistic Reference Model in the sense that the reference model shall capture and define in a holistic way the types and number of levels of abstractions of functionality, at which DMEs/DEs that govern autonomic behaviours (i.e. Self-* features) by way of driving hierarchical control-loops within a node architecture and the network architecture as a whole, need to be designed. The notions of "upper DE", "lower DE", "time-scaling and notions of fast control-loop versus slow control-loop", etc are created.

- To be seen as an evolvable Reference Model in the sense that the Reference Model shall incorporate design for evolvability principles that allow the DMEs to evolve as new requirements for autonomicity emerge, such as the need to improve their cognitive (learning and reasoning) properties, as well as the need to distinguish and fix their interfaces that shall remain constant (mandatory).

7.3.2 GANA Meta-Model

In addition to the properties above, the aspects and issues that require centralization of some of the autonomic decision-making processes should be captured by the Reference Model. The point is: by separating issues of concern for different, well-defined levels of autonomic decision-making processes i.e. control loops, the GANA architecture would allow the production of standardizable specifications of autonomic components and their behaviours including their coupling and interactions. With this understanding, the Generic Autonomic Network Architecture (GANA) would allow for Modelling and Validation of the captured (specified) Autonomic Behaviours using Formal Description Techniques (FDTs) such as the well-known and successful SDL [i.26]. Also, associated with the Generic Autonomic Network Architecture Reference Model, a Domain-Specific Meta-model for the Autonomic Networking Domain in particular, i.e. the GANA Meta-Model is required, that defines in a formal way, the concepts and semantics for autonomic network engineering, concepts such as control loops, Decision-Making-Elements and their associated Managed Entities, as well as other types of information suppliers, interfaces, etc, required by a Decision Element (DE). The GANA Meta-Model is formalized definition and description of GANA architectural Reference Model, and can be expressed in OMG's MOF language for example, or as a UML Profile. The GANA Meta-Model enables the development of a Model-Driven Methodology and Tool Chain that would allow for the formal specification of GANA Decision-Elements (DEs): their Control-Loops and interactions, the employment of policy-based control by DEs over their Managed Entities (MEs), as well as the simulation and validation of autonomic behaviours of DEs to address "Design for Stability Requirements". As known in Systems Engineering, Meta-Models can be standardized to help so called Tool-Vendors create Tools required by Network Equipment Manufacturers for designing, simulating, validating components and generating software-code. The GANA Meta-Model is required for developing the Tool-Chain that is useful for the developers of autonomic components, especially for the Network Equipment Manufacturer. The GANA Meta-Model itself is not included in the present document but will be provided as a separate document. Preliminary work on a GANA Meta-Model can be found in EFIPSANS deliverable D1.8b [i.5] and [i.35].

7.3.3 GANA principles relating to structural aspects

Therefore, GANA is an evolving architecture model that is meant to also address the following problems and issues:

1) Complexity - by defining some abstractions for autonomic/self-management functionalities at four basic hierarchical levels as described later.

2) How to ensure that the decision-making processes for autonomic behaviours are conflict-free and stable. This subject on how stability is addressed in GANA, is presented in clause 9.16.

3) How to incorporate design principles that enable "in-network management" or "network-intrinsic management" and define constraints and boundaries for in-network management - self-organization and self-management achieved by the network nodes themselves without the requirement and intervention of specially instrumented network management systems (NMS's) meant to co-ordinate the network. This subject is addressed in clause 9 (clause 9.8) and clause 11 (clause 11.10).

4) Capturing the kind of perspectives offered to end-users and operators of autonomic/self-managing networks, such as the interfaces that are meant to allow humans to define network-level objectives that govern the operation of an autonomic network under the control of an administrative domain i.e. Interfaces for Network Governance. This subject is addressed in clause 11.

8 Core Concepts of the GANA Reference Model

8.1 Approach taken to developing the concepts

GANA establishes some architectural principles, including the required abstractions to engineering autonomic Decision-Making-Elements of an autonomic network. This means, GANA sets the principles and guidelines that need to be followed when specifying and designing autonomic network components of a self-managing network. In GANA, Autonomicity - realized through control-loop structures operating within network nodes/devices and the network as a whole, is seen as an enabler for advanced and enriched self-manageability of network devices and networks.

These self-* functions instantiated in so-called context-aware Decision-Making-Elements (DMEs) need to potentially exhibit cognitive properties, designed for the management of multiple and diverse networking environments. As described earlier, a central concept of GANA is that of an autonomic Decision-Making-Element ("DME" or simply "DE" in short - for Decision Element). A Decision Element (DE) implements the logic that drives a control-loop over the "management interfaces" of its assigned Managed Entities (MEs). The Control-Loop is what determines autonomicity. As will be presented later, on the subject of learning, reasoning, planning and cognition, not every type of a DE/Control-Loop shall have the four properties. Therefore, in GANA, self-* functionalities are functionalities implemented by Decision Element(s).

As already discussed earlier, the "generic nature" of GANA Model includes the definition of the interfaces and their fundamental operations that need to be supported by a Decision Element, the interconnection and relations among the DEs within node and network architectures, as well as the assignment of the types of Managed Entities (MEs) that are managed by their associated DEs, including the fundamental operations that should be supported on the management-interfaces of the individual MEs, while leaving out implementation-oriented details.

From such a commonly shared, unified, and holistic Reference Model as the GANA, either clean-slate based architectural refinements (and implementations) **OR** incremental/evolutionary architectural refinements and implementations should then be derived. An evolutionary approach can be pursued for applying ("instantiating") GANA for autonomic management and control of today's protocols and networking mechanisms.

The vision of a Self-managing Future Networks i.e. Future Internet, adopted in GANA, involves the designing of the network nodes/elements in such a way that all the so-called traditional network management functions, defined by the FCAPS management framework [i.2], as well as the fundamental network functions such as routing and forwarding, are made to automatically feed each other with information such as goals and events, in order to effect feedback processes among the diverse functions. These feedback processes enable collaborative self-adaptation (reactions) of the various functions in the network and/or individual nodes, in order to achieve and maintain some defined network goals. In such an evolving environment, it is required that the network helps itself to detect, diagnose and repair failures, as well as to constantly adapt its configuration and optimize its performance. Autonomicity defined on the basis of control-loops and feed-back mechanisms [i.28], becomes an enabler for advanced self-manageability of networks beyond what can be achieved through scripting based automation techniques.

Following this trend in network evolution, the FCAPS functions become diffused within the node architectures, apart from implementing the Management Plane of an overall network architecture - whereby traditionally, a distinct management plane is engineered separately from the other functional planes of the network. Since even the management functions become inherent functions of the fundamental node/device architectures, it means that the functional planes of a self-managing network, would need to be (re)-defined and re-factored (see [i.25] and [i.13]). New concepts, functional entities and their associated architectural design principles that enable self-management at different levels of node and network functionality and abstractions, are thus required.

8.1.1 Implementation Use Case as an illustration

The EFIPSANS research project successfully performed "instantiations" (applications) and validation of GANA for autonomic management and control of different types of Managed Entities (Protocols, Stacks and Mechanisms at GANA' lowest level/layer" for diverse network environments (Fixed/Mobile/Wireless Networks) (for information on the subject, the reader is referred to EFIPSANS deliverables [i.5]).

GANA is a Model that allows for the production of standardizable specifications of autonomic elements and behaviour due to the fact that the key founding principle of GANA is: "clearly separate specification and design issues for autonomic elements and behaviour (that includes the structural and behavioural aspects of the associated Decision-making-Elements) from their implementation issues". GANA is meant to benefit both the "evolutionary" approaches and "revolutionary/clean-slate" approaches to Future Internet (Future Networks) design, in the long run, as GANA becomes adopted as the common reference model for autonomic network engineering, cognition and self-management. Evolvability of GANA can be assured in two ways:

1) Identifying the interfaces and concepts of GANA that shall remain constant to allow new components to be added to the architecture.

2) Application of model-based evolution of the elements of the architecture by ensuring that communication between architects and implementers of GANA should be based on model evolution of the specifications of all the elements.

8.2 GANA Reference Model: Structure, Core concepts and Principles

In GANA, the four basic levels of abstractions for which DEs, MEs and Control-Loops can be designed following what we call the GANA Hierarchical Control Loops (HCLs) framework briefly introduced below. The Levels are elaborated and described in more detail in the next clause. The reason as to why hierarchical control loops is also explained.

In GANA, the four basic levels of abstraction for which DEs, MEs and Control-Loops can be designed following what we call the GANA Hierarchical Control Loops (HCLs) framework are defined below.

NOTE: While we aim at specifying the GANA Reference Model we focus on firstly describing each abstraction-level at which autonomicity(control-loop) and self-management features can be designed into the associated functionality abstracted at each level, and then build a picture on how the different levels interact and relate to each other in the subsequent sections and clauses that elaborate the Reference Model. This means a single figure for the four levels is presented in later sections. Why Hierarchical Control Loops?

1) This allows us to separate the key issues of concern when trying to produce Autonomic Behaviour Specifications in which well defined information sets that drive the behaviour of a Decision-making Element and its associated Managed Entity (ies) and control-loop can be captured and specified.

2) Hierarchies allow for some decisions to be taken autonomously at different levels of control and complexity.

3) Hierarchies offer flexibility towards controlling/resolving decision conflicts and priorities as mentioned earlier.

GANA basic four- HCLs (Hierarchical Control-Loops) Levels

Level-1 (Protocol-Level): protocol-level (the lowest level) by which self-management is associated with a particular network protocol itself (whether monolithic or modular) that implements the notion of a control-loop. What are the protocol layers involved in level-1? Any protocol-layer (layer-1 to layer-7 of OSI) may be considered while introducing autonomicity into an individual protocol. However, later , why it is not desirable to introduce autonomicity into individual protocols will be treated.

Level-2 (Function-Level): the abstracted network functions-level (directly above the protocol(s)-level) that abstract some protocols and mechanisms associated with a particular network function e.g. routing, forwarding, mobility management, etc-whereby it becomes possible to reason about autonomic routing, autonomic forwarding, autonomic fault-management, autonomic configuration management. The term "Autonomic" is used because of having a specific Control-Loop over the protocols and mechanisms abstracted by the "function" in question that performs auto-configuration of the protocols and mechanisms, listens for events from the protocols and mechanisms, as well as the environment and performs adaptation actions accordingly.

Level-3 (Node-Level): the level of the node/device's overall functionality and behaviour i.e. a node or system as a whole is also considered as level of self-management functionality.

Level-4 (Network-Level): the level of the network's overall functionality and behaviour (the highest level).

A more detailed description of the four hierarchical levels is provided later in the present document.

The well known abstract model of an autonomic system defined by IBM [i.10] and [i.38] is an abstract model from which one can further derive control loops adapted to a particular type of system belonging to a concrete domain such as the domain of autonomic networking. Indeed, there is a need to derive a model specifically meant for autonomic networking, not autonomic computing or autonomic systems in general. As illustrated in the present document, an appropriate model for an autonomic networked system for the domain of autonomic networking would allow to reason and think of a particular autonomic networking functionality that implements the concept of a control loop. This means, having the ability to reason about autonomic networking functions such as autonomic routing, autonomic forwarding, autonomic fault-management, autonomic configuration management in the sense that the autonomic element (referred to as a Decision-Making Element in the present document) drives the specific control loop using information learnt from its required information suppliers (possibly multiple types) to control the behaviour of the functionality considered autonomic.

Figure 7 shows a model of an autonomic networked system and its associated control loop developed that is derived from IBM-MAPE model specifically for autonomic networking. Here, the model is still a generic model but illustrates the possible distributed nature of the information suppliers. GANA model adopts the concept of a Decision-Making-Element as the autonomic element, that plays a similar role as an Autonomic Manager in the IBM-MAPE model. It also adopts the concept of a Managed Entity (ME) to mean a managed resource or an automated task in general, instead of a Managed Element-a term used in traditional network management terminology, in order to be more generic and to avoid the confusion that comes with the use of the term "Managed Element", which is normally associated with only meaning a physical Network Element (NE) and not some functional entity within a node/device such a protocol module or a component such as a monitoring component.

Figure 7 also illustrates the fact that the behaviours or actions taken by the DE do not all necessarily have to do with triggering some behaviour or enforcing a policy on the Managed Resource(s) (i.e. changing its behaviour thereof) but that, some of the behaviours executed by the DE may have to do with communication between the DE and other entities e.g. other DMEs in the system or network. This is indicated by the extended span of the arrow "Downward Information or Communication flow" and the "Horizontal Information/Communication flow" to other DEs. The same goes for the fact that a DE also exposes views such as events to its "upper" DE and receives policies, goals and command statements from its "upper" DE.

For example, in such information or communication flows, the Decision-making Element (DE) may need to self-describe and self-advertise to other Decision-Making Elements (DE) in order to be discovered or to discover other Decision-Making Elements (DEs), in order to communicate or get "views"-knowledge known by the other Decision-Making Elements (DEs). In addition, the upward Information Supply also facilitates auto-discovery of other DEs or environment. What also needs to be noted is the fact that the nature of the Managed Entity(ME) may be of a physical nature e.g. a device, memory, or may have the nature of a an Automated Task (in an abstract/general sense) implemented by either a system function(s) or a networking function(s) that is implemented by a single protocol, or diverse protocols that may employ diverse algorithmic schemes or policies. Therefore, the Managed Entity (ME) may take the nature of Managed Resource (or a Managed Automated Task) such as a Networking Function e.g. Routing Function, that is implemented by a single protocol or diverse protocols abstracted by the Function. This means that the ME may employ diverse types of Algorithmic Schemes (i.e. behaviours) or Policies that may be requested (via a command) by the responsible DE for enforcement on the ME (depending on context changes or network state changes). A DE may be assigned to autonomically manage and control multiple MEs. The Sensor and Effectors parts of the "Management Interface" of a particular ME e.g. a protocol, that the binding DE uses to autonomically manage the ME, may be realized by a MIB (Management Information Base). The "Upward Information Supply" reveals the need for "cross-layering" in some cases in order to facilitate the decision-making self-adaptive behaviour implemented by a DE.

NOTE: The generic model described on the figure focuses on communication and information flow between the functional blocks than specifying the interfaces between the functional blocks per-se. The subject of the interfaces (reference points) is covered in the subsequent later sections/clauses that elaborate those aspects. A summary of the Reference Points is given later in clause 13. As illustrated later, DEs form interfaces with each other and communicate with each other through those interfaces. The so-called hierarchical, peering and sibling relationships between DEs (described in clause 9.8), the interface model of a DE and that of an ME (described in clause 9.11), the need for an Information/Knowledge Sharing Repository for sharing info/knowledge among functional entities (e.g. DEs) within a node (presented in Figure 19), all relate to the way DEs and MEs communicate with each other. Is there a control loop between peer DE(s)? A control-loop is a concept that binds an autonomic manager (i.e. a DE) and a Managed Entity (ME) (or multiple MEs). The "horizontal interaction" between peer DEs facilitates for realizing "distributed decision-making", and may be viewed as a control-loop when the interaction is such that one DE is playing a role of manager of the other DE, or may simply be called a kind of a distributed control-loop due to the mutual management and control among the DEs involved. More details on the subject are provided later in this clause and in clauses 9.8 and 11.10 that discuss DE-to-DE horizontal interactions (e.g. in the case of realizing "network-intrinsic management" i.e. "in-network management" discussed later).

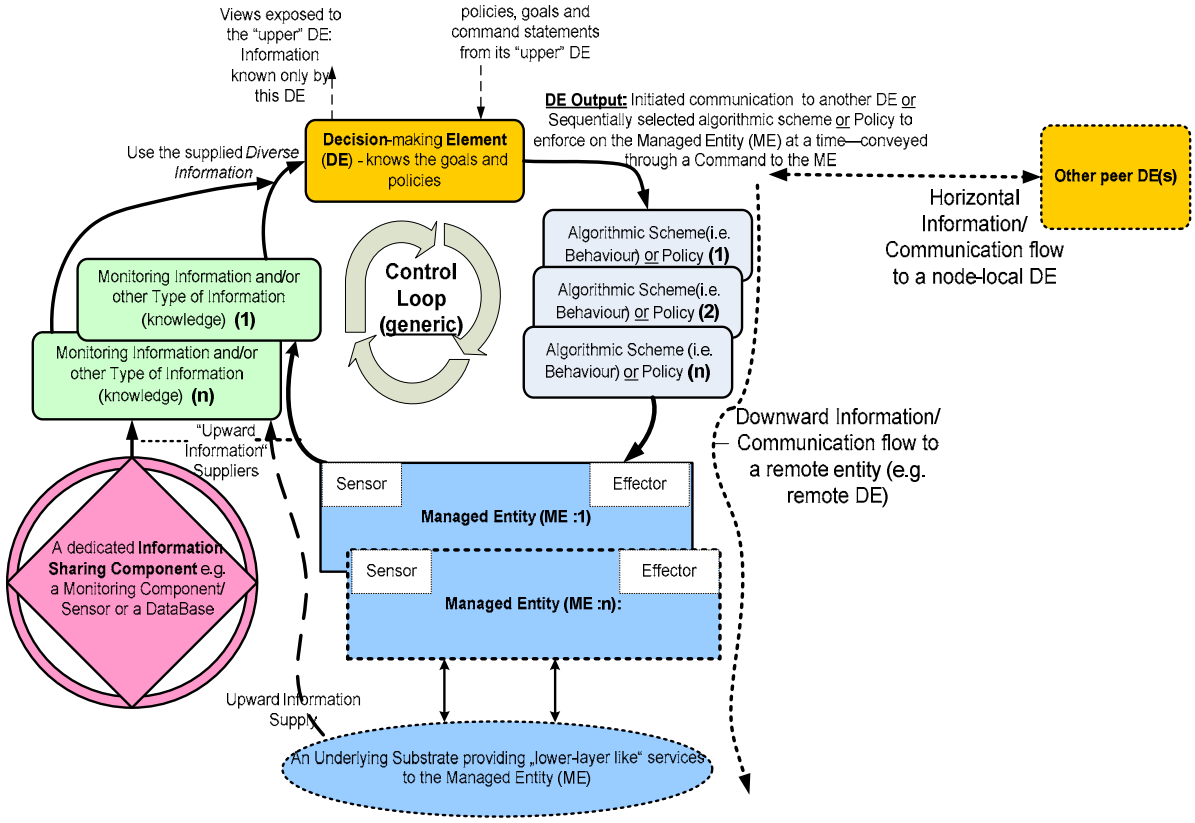


Figure 7: Generic model of an Abstract Autonomic Networked System

Clause 8.2.1 show this generic model of an abstract autonomic networked system can be applied at any of the "*four-basic levels of abstraction of functionality*" defined by GANA.

8.2.1 The four GANA basic abstraction levels and their associated types of Control Loops

In GANA, the four basic levels of abstraction for which DEs, MEs and Control-Loops can be designed following what we call the GANA Hierarchical Control Loops (HCLs) framework are defined below.

NOTE: While we aim at specifying the GANA Reference Model we focus on firstly describing each abstraction-level at which autonomicity(control-loop) and self-management features can be designed into the associated functionality abstracted at each level, and then build a picture on how the different levels interact and relate to echa other in the subsequent sections and clauses that elaborate the Reference Model. This means a single figure for the four levels is presented in later sections. Why Hierarchical Control Loops?

1) This allows us to separate the key issues of concern when trying to produce Autonomic Behaviour Specifications in which well defined information sets that drive the behaviour of a Decision-making Element and its associated Managed Entity (ies) and control-loop can be captured and specified.

2) Hierarchies allow for some decisions to be taken autonomously at different levels of control and complexity.

3) Hierarchies offer flexibility towards controlling/resolving decision conflicts and priorities as mentioned earlier.

First is described how the generic model of an abstract autonomic networked system presented in clause 9.2 can be applied at any of the "four- basic levels of abstraction of functionality" defined by GANA. Because the abstraction levels introduce the notion of a Decision Plane that consists of a Vertical View and a Horizontal View of interactions of Decision-Elements, elaborated on the Hierarchical, Peer and Sibling Relationships of DEs and DE-to-DE peer interfaces for realizing "network-intrinsic" management (in-network management) and control in the definition of Decision-Plane (in the subsequent section that defines the GANA Functional Planes).

8.2.1.1 GANA Level-1: the Protocol-Level

It is the lowest level, by which self-management is associated with a particular network protocol itself, whether monolithic or modular. What are the protocol layers involved in level-1? Any protocol-layer (layer-1 to layer-7 of OSI) may be considered while introducing autonomicity into an individual protocol. There is growing opinion, however, that future protocols need to be simpler, i.e. with no decision logic embedded, than today's protocols which have become too hard to manage due to intrinsic decision logic that may interact in an undesired way with the decision logic of other protocols. This means, as explained in 4D [i.25], CONMan [i.8] and in [i.44], there is a need to rather implement decision logic at a level higher, i.e. outside the individual protocols.

NOTE 1: For this reason, AFI supports the approach of focusing on introducing autonomicity outside of individual protocols i.e. at higher levels of abstractions defined by the Reference Model.

The concepts of a Control Loop, Decision-Making Element, Managed Resource or Managed Automated Task, as well as the related self-manageability issues may be associated with some implementation of a single network protocol (whether monolithic or modular). For example, protocols like OSPF and TCP are known to have the above concepts intrinsically implemented within the holistic behaviour of the protocol. Figure 8 shows the case of a protocol-intrinsic control loop. This permits to reason about a protocol being autonomic or having some degree of autonomicity. This level allows to have some individual protocols exhibiting some autonomic (control-loop) features. A Decision-Making-Element (DE) that constitutes the core logic of the protocol and drives a control-loop intrinsic to the protocol is referred to as a Protocol-Level DE (or GANA Level-1 DE).

NOTE 2: The bundling of the Protocol-Level-DE and the single functional entity that contains the managed protocol driver results in an autonomic protocol, in a somewhat similar way the OSPF protocol or TCP protocol is designed to operate.

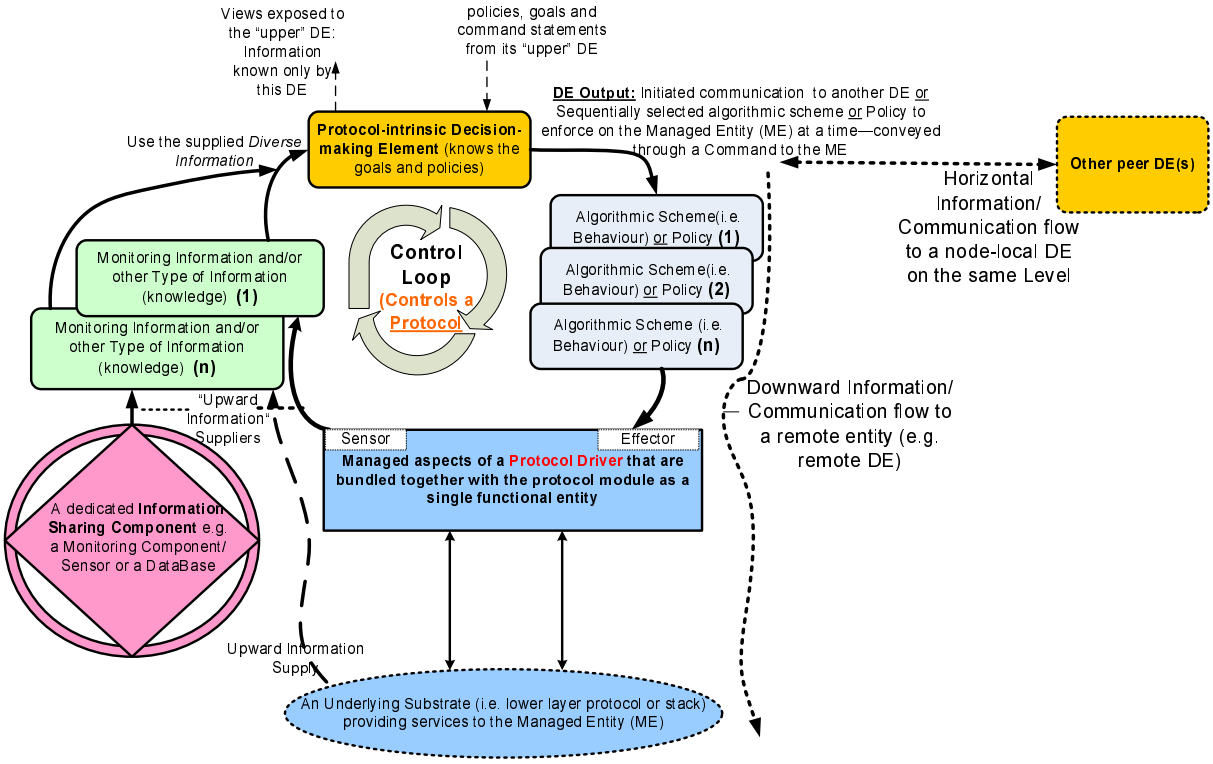


Figure 8: A protocol-intrinsic control-loop

8.2.1.2 GANA Level-2: the Function-Level

It is an abstracted network functions level, directly above the protocol(s) -level, that abstracts some protocols and mechanisms associated with a particular network function e.g. routing, forwarding, mobility management, etc., whereby it becomes possible to reason about autonomic routing, autonomic forwarding, autonomic fault-management, autonomic configuration management. Why call them "abstracted network functions"? Because "abstracting" i.e. coming up with a set of abstractions of functions is a process that should be exercised since there no standard set of "network functions". The present document, identify which functions and how many of those basic network functions are defined in the Reference Model. From management point of view, some of the abstracted functions are superior to the other basic networking functions, in terms of management hierarchy. The management functions that are rather superior to basic networking functions such as routing, forwarding, mobility management, QoS management, are placed on a higher-level (node-level) than at the Function-Level since they manage the inferior functions.

The concepts of a Control Loop, Decision-Making Element, Managed Resource or Managed Automated Task, as well as the related self-manageability issues may also be associated with a higher level of abstraction than a single protocol (see Figure 9). This means that the aspect of autonomicity may be addressed on the level of abstracted networking functions (or network functions) such as routing, forwarding, mobility management, QoS management, etc. At such level of abstraction, what is managed are a group of protocols and mechanisms that are collectively wrapped by what it may be called a Function Block, and are considered to belong to the functionality of the abstracted networking functions e.g. all routing protocols and mechanisms of a node become managed by a Decision-Making Element assigned and designed to manage only those protocols and mechanisms.

From an implementation point of view, the managed Function Block in this case, can be considered as a wrapper around all the related mechanisms and protocols of the abstracted networking function of the node, exposing their "views" to the Decision-Making Element (DE) or otherwise the DE may be considered to have direct access to the managed mechanisms and protocols and manages them directly. Alternatively, the managed Function Block may be the one that has direct access to the mechanisms and protocols and orchestrates them based on the decisions enforced by the DE. In that case, the Functional Block can simply be considered as a wrapper around the abstracted protocols and mechanisms. This level of abstraction allows to reason about autonomicity of self-managing properties at this particular level of abstraction e.g. autonomic routing in the node/network. A Decision-Making-Element (DE) that drives a control-loop over the Managed Entities assigned to be managed by the DE is referred to as a Function-Level DE (or GANA Level-2 DE).

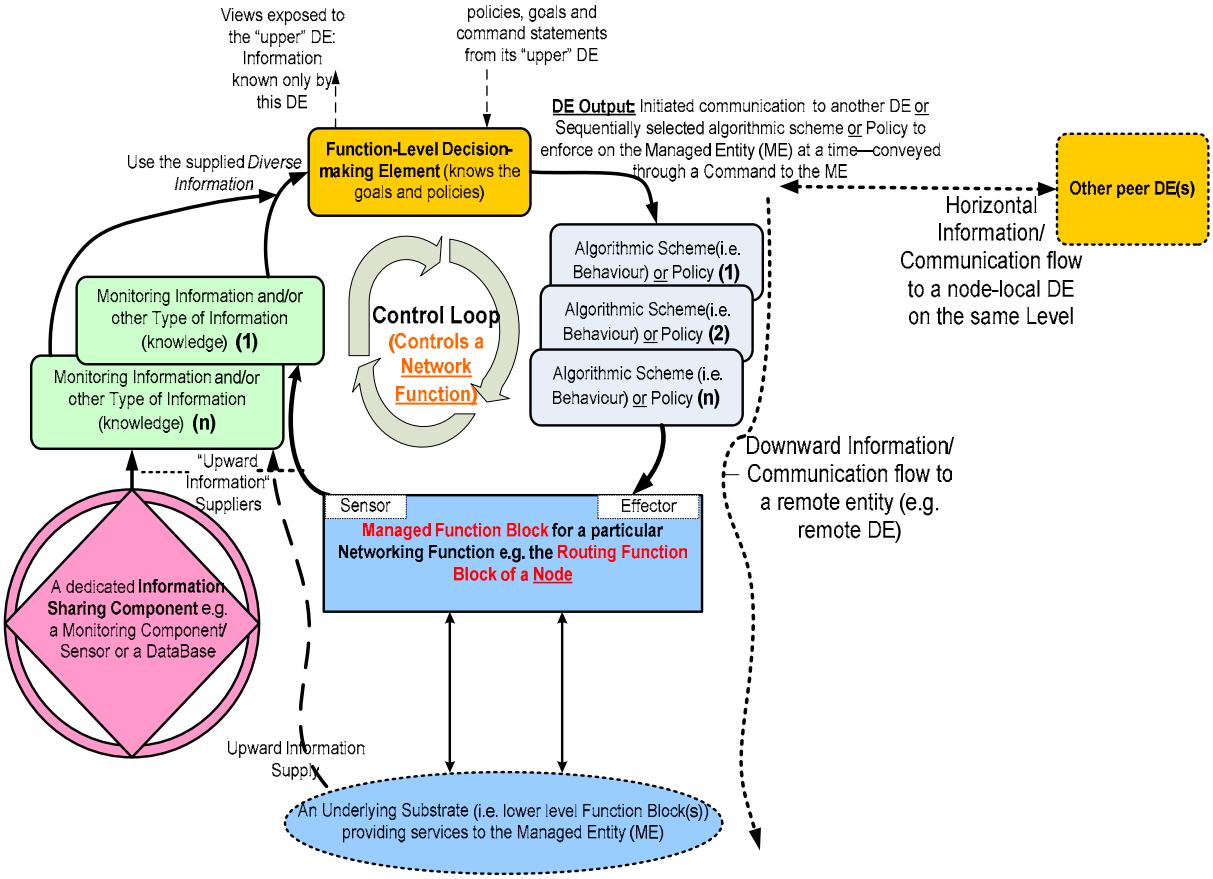


Figure 9: A Control Loop specific to an abstracted "Networking Function"

8.2.1.3 GANA Level-3: the Node-Level

It is the node/device's overall functionality and behaviour level i.e. a node or system as a whole is also considered as level of self-management functionality.

On a higher level of autonomic networking functionality than the level of abstracted networking functions of a node/network such as routing, the concepts of a Control Loop, Decision-Making Element, Managed Resource or Managed Automated Task, as well as the related self-manageability issues may be associated with a system (node) as a whole. Figure 10 illustrates that in this level of self-managing (autonomic) properties, the lower level Decision-Making Elements (DEs) operating on the level of abstracted networking functions (described earlier) become some of the Managed Automated Tasks (entities) of the main Decision-Making Element (DE) of the system(node). This means the node's main DE (referred to as the NODE-MAIN-DE (i.e. GANA Level-3 DE)) has access to the "views" exposed by the lower level DEs and uses its knowledge of the higher level (system and network level objectives/goals) to influence (enforce) the lower level DEs to take certain desired decisions, which may in turn further influence or enforce desired behaviours on their associated Managed Resources or Managed Automated Tasks, down to the lowest level of individual protocol behaviour (whether the protocol itself is autonomic or not).

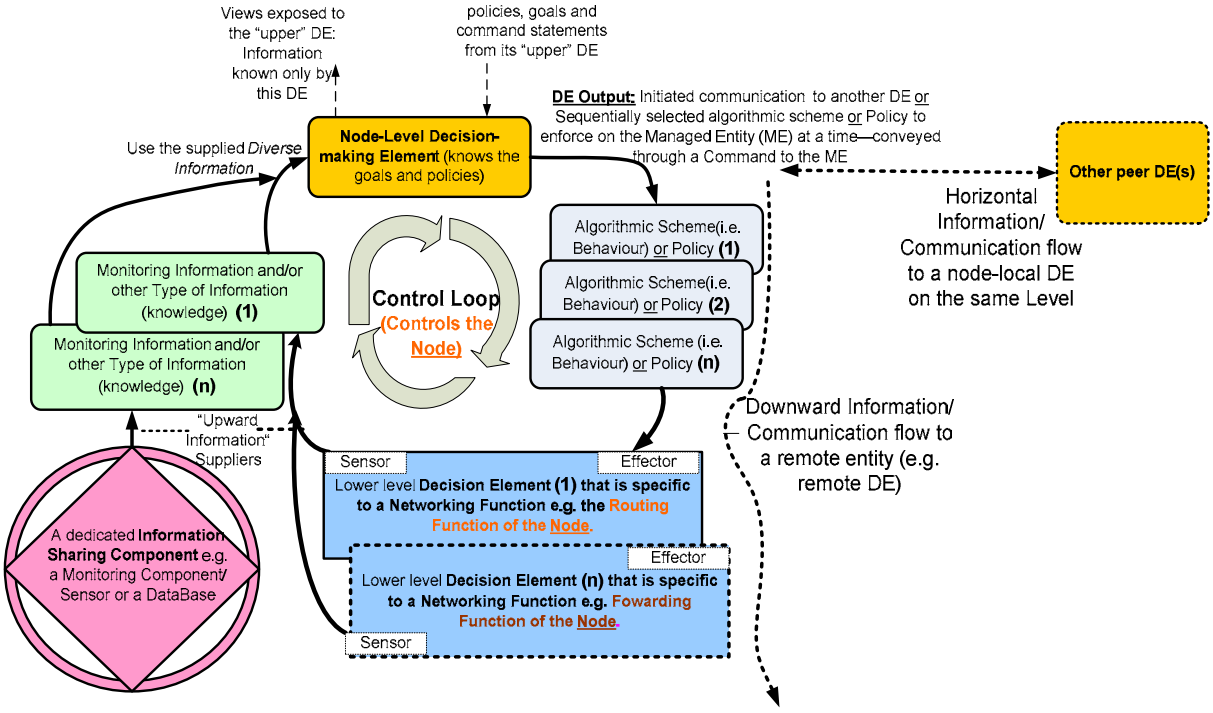


Figure 10: The Node's Main DME and main control-loop of an Autonomic Node

8.2.1.4 GANA Level-4: the network's overall functionality

The next higher level of self-manageability (autonomicity) after the node level described above is the network level.

Figure 11, illustrates that there may exist a logically centralized Decision Making Element or plane such as in the 4D network architecture [i.25] that knows (through some means) the objectives, goals or policies to be enforced by the whole network.

The objectives, goals or policies may actually require that the nodes' Main (top-level) DEs, belonging to the network covered by the centralized DE(s) export "views" such as events and state information to the centralized DE(s), in order to influence or enforce the DEs of the nodes to take certain desired decisions that may in turn have an effect of inductive decision changes on the lower level DEs of individual nodes i.e. down to protocol level decisions.

Figure 11 further illustrates how a distributed network-level Control-Loop may look like. Alternatively one could involve the main Decision-making-Elements of nodes working co-operatively to manage the network without the presence of a logically centralized DE(s). A Decision-making-Element (DE) that drives a network-level Control-Loop is referred to as a NETWORK-LEVEL-DE (or GANA Level-4 DE). The network-level-DEs may be distributed in what is referred to as the Knowledge Plane nodes which communicate together. This subject is covered in more detail in the clause 10.1.3 that describes the Knowledge Plane, Cognition and Information/Knowledge sharing. Reference Points (logical interfaces) related to interaction between Network-Level-DEs are also described in clause 12.

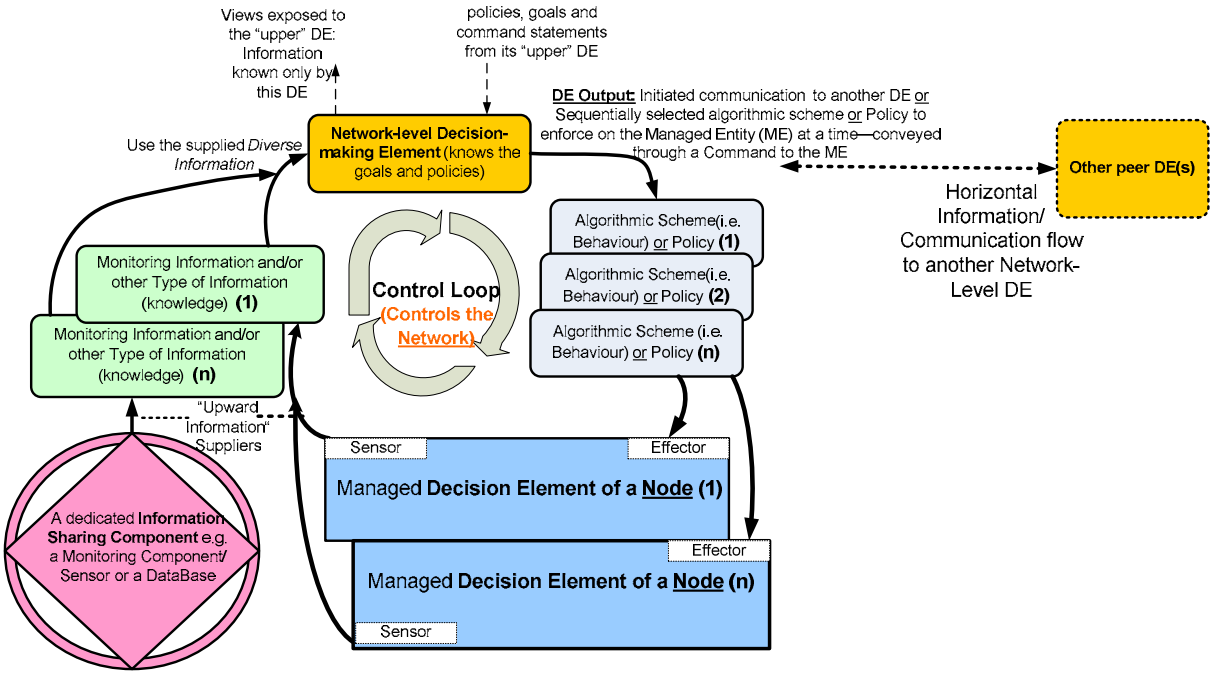


Figure 11: Managed "Main Decision-Making Elements" of Nodes

Therefore, a key design principle inherently incorporated into the GANA architectural Hierarchy of DEs is that of coordinated access-control to the lowest level Managed Entities (e.g. protocols) through synchronization among DEs and arbitration means in the intermediate levels in the hierarchy down to the Decision Elements at the Functions-Level that THEN directly access and enforce changes to the GANA lowest level Managed Entities (protocols and mechanisms). Up the GANA Hierarchy of Control-Loops, the slower the Control-Loops become - all of which shall be designed following the following guidelines to designing stable control systems that stem from concepts from the field of control theory:

1) Establishing well-defined "valid operating regions" of particular control-loops.

2) Decoupling control systems by ensuring that they control different independent outputs based on independent inputs and if this is not possible, then tuning them so that they impose control at very different timescales can help to decouple systems that would otherwise couple [i.34].

The subject of Stability in GANA is addressed in more detail in clause 9.16.

In GANA:

- Lower level DEs expose "views" up the Decision Plane (see definition and elaboration provided later), allowing the upper ("slower Control-Loops-with some planning capabilities" at Network-Level) control loops to control the lower level (faster) control-loops (lower level DEs).

- Changes computed in the upper DEs implementing slower Control-Loops are propagated down the DE hierarchy to the Functions-Level DE(s) implementing the faster control-loops that then arbitrate and enforce the changes to the lowest level Managed Entities (protocols and mechanisms).

GANA exhibits the characteristics of a hierarchical control system described in [i.1] though in the evolution of GANA, some of the characteristics of such a hierarchical control system may need to be modified or constrained depending on the instantiation and implementation of GANA for particular network devices and environments.

8.2.2 GANA Functional Planes

GANA, like 4D, first take the position that the functional planes known in today's world of networking can be compressed (with merging and re-factoring some of the planes) into four functional planes that could still be called the Decision Plane, the Dissemination Plane, the Discovery plane and the Data plane, GANA adopts from the 4D but we redefine them as in the clauses below.

8.2.2.1 The Decision Plane

The **Decision Plane** makes all decisions driving a node's behaviour (including the behaviour of all managed entities of the node) and network-wide control, including reachability, load balancing, access control, security, and interface configuration. Replacing today's Management Plane, the decision plane operates in real time on a network-wide view of the topology, the traffic, events, context and context changes, network objectives/goals/policies, and the capabilities and resource limitations of the nodes and devices of a network of some scope (Definition adopted but with modification, from the 4D architecture). GANA principles consider that the elements of the Decision plane are autonomic elements i.e. the Decision-Making Elements (DEs).

GANA, considers that Decision-making Elements that drive Control Loops of an autonomic node/network form a Hierarchy consisting of four basic levels of Autonomicity - Control-Loop Structures, a framework of Hierarchical Control-Loops (HCLs) called the GANA HCLs framework (see Figure 17 and Figure 20) that was described earlier. The Vertical view of the GANA Decision Plane as well as the Horizontal View are further elaborated in the clauses 11.1.1.1 and 11.1.1.2 that reflect on Vertical DE relationships and Horizontal DE relationships across network nodes/devices.

NOTE: The paragraphs below further elaborate on the four basic levels of abractions described in clause 8.2.1, focusing on the Vertical View and a Horizontal View of interactions of Decision-Elements.

GANA Level -1 DEs: Decision-making Elements (DEs) intrinsic in some protocols (existing protocols or new ones), e.g. TCP or OSPF (the lowest level in the Decision Plane hierarchy). This level allows to have some individual protocols exhibiting some autonomic (control-loop) features. It refers to DEs introduced at protocol level "**Protocol-level DEs**". A Protocol-level DE simply means the protocol is intrinsically made to implement a control loop. There is growing opinion, however, that future protocols need to be simpler (i.e. with no decision logic embedded) than today's protocols which have become too hard to manage due to intrinsic decision logic which may interact in an undesired way with decision logic of other protocols. This means, as explained in [i.25], [i.8] and other references such as [i.44], there is a need to rather implement decision logic at a level higher (i.e. outside the individual protocols).

GANA Level- 2 DEs: Decision-making Element(s) (DE(s)) specifically designed for concrete Function Block(s) that abstracts a particular Networking Function(s) and its associated mechanisms e.g. the Routing protocol(s) and mechanisms. A DE of this kind manages protocols and mechanisms (whether autonomic in their own capacity i.e. they have protocol intrinsic control loops, or not) abstracted by the Function Block. The DEs operating at this level are called the "**Function-Level DEs**". For more examples of abstracted network functions see Figure 17.

GANA Level -3 DEs: Decision-making Element (NODE_MAIN_DE) managing the overall behaviour of a Node by managing lower level DEs. At this level of self-management (autonomic) properties ("**Node-Level**"), the lower level Decision-Making-Elements operating at the level of abstracted networking functions become the Managed-Entities of the main DE (NODE_MAIN_DE) of the system (node). This means the node's NODE_MAIN_DE has access to the "views" exposed by the lower level DEs and uses its overall knowledge to influence (enforce) the lower level DEs to take certain desired decisions, which may in turn further influence or enforce desired behaviours on their associated Managed-Entities (MEs), inductively down to the lowest level of individual protocol behaviour.

GANA Level-4 DEs: Decision-making Element(s) operating on managing the behaviour of a network of some scope and either having the ability to control/influence the main Decision-making Elements of nodes covered by the scope (see Figure 17 and Figure 20 for hierarchies) or that the main (top-level) Decision Elements of the nodes co-operatively interact in order to self-organize and manage the network (see the top peer relationships on Figure 20). From GANA perspective, these DEs operating at this level, are called the "**Network-Level DEs**".

Therefore, the Decision Plane consists of a hierarchy of Decision-making Elements (DEs) that have the ability to form peers with other DEs (see Figure 17 and Figure 20). Our concept of a DE is not the same as the Decision Element (DE) concept of the 4D architecture, in the sense that in GANA, a DE is an autonomic element that exists in any node or device in the network and has properties determined by the properties of the Decision Plane, which also differs from the Decision Plane defined by the 4D. However, Decision Elements in 4D contribute to functional properties of Network-Level-DEs in GANA (more details are provided in clause 9.12). In the statements that follow below, the properties of the DEs of the GANA architecture, such as hierarchical, peer and sibling relations among DEs are described.

The properties required of the Decision Plane by autonomicity are that the plane supports the following Principles:

- Property P1: Hierarchical Relationships between Decision Elements (DEs) (see Figure 17, Figure 20 and Figure 11): Hierarchical Relationships also means that a lower level DE(s) is managed by the immediate upper level DE.
- Property P2: Peering Relationships between Decision Elements (DEs) (see Figure 20): A peer relationship is about the facilitation of some communication between DEs for exchanging views, negotiations pertaining to (re)configurations of managed entities, or requesting each other for some service(s).
- Property P3: Sibling Relationships between Decision Elements. Sibling relationship simply means that the entities are created or managed by the same upper level Decision Making Element (DE). This means that the entities having a sibling relation can form peer relationship within the autonomic node or with other entities hosted by other nodes in the network, according to the protocol defined for their means to communicate with other DEs (see Figure 17 and Figure 20).

Figure 12 shows the "relative distribution and interactions views" of Decision Plane elements (DEs). Figure 20 illustrates in more detail the Hierarchical, Peering, Sibling Relations and Interfaces of DEs in GANA.

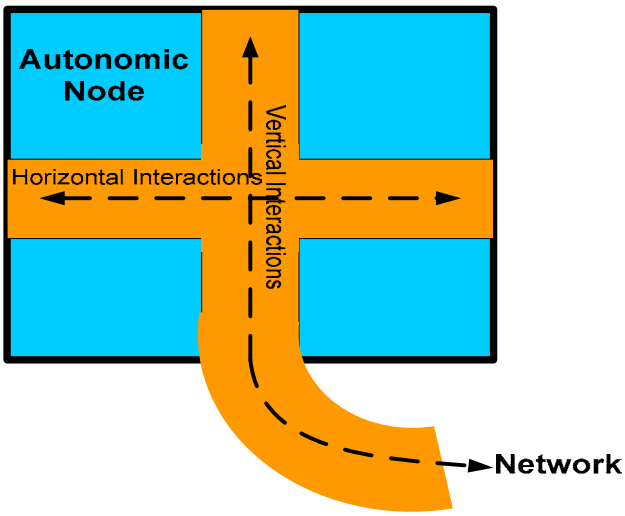


Figure 12: The "relative distribution and interactions views" of Decision Plane elements (DEs)

8.2.2.2 Dissemination Plane

The **Dissemination Plane** consists of mechanisms and protocols that provide a robust and efficient communication substrate that is used to exchange control information as well as any special type of information (or knowledge) that is not considered as the actual user data e.g. monitoring data, among entities inside a single box/node e.g. DEs, and among nodes and/or devices, via push or pull models for information retrieval. (Definition adopted but with modification, from the 4D architecture). In [i.14] the concept of the "Spinal Cord of an Autonomic Network (SCAN)" is defined, which describes dissemination associated functions and resilient secure communication structures that would be considered to belong to the Dissemination Plane and can be used for conveying the following types of information or knowledge: Signalling information; Monitoring Data, including change of State Info; Other types of Control information that need to be exchanged between Decision Elements (DEs); Incidents Info e.g. faults, errors, failures, alarms, etc. Example elements of the Dissemination Plane are protocols or mechanisms that can be considered to belong to this Plane are: ICMPv6, MLD, DHCPv6, SNMP, IPFIX, NetFlow, IPC mechanisms and more.

Decision Elements (DEs) of the Decision plane **use the services** of the Dissemination Plane to communicate with each other. The "relative distribution and interactions views" of Dissemination Plane elements is similar to the case of the relative distribution and interaction views of the Decision Plane elements (Figure 12).

8.2.2.3 Discovery Plane

The **Discovery Plane** consists of protocols or mechanisms responsible for discovering what entities make up the network or a service and creating logical identities to represent those entities. The discovery plane defines the scope and persistence of the identities, and carries out the automatic discovery and management of the relationships between them. This includes box-level discovery (e.g. what interfaces are on this node? How many FIB entries can this node hold? Neighbour-discovery - including "Capabilities" of nodes of interest e.g. routers that are members of the all-routers multicast address of which this node is member, network discovery, service-discovery, etc. (Definition adopted but with modification, from the 4D architecture). Protocols and mechanisms for self-description and self-advertisement of capabilities of entities should be at the heart of this plane. Example elements of the Discovery Plane are: IPv6 Neighbour Discovery and, IPv6 SEND, Service-Discovery Protocols/mechanisms, Topology-Discovery Protocols, and more. The "relative distribution and interactions views" of Discovery Plane elements is similar to the case of the relative distribution and interaction views of the Decision Plane elements.

8.2.2.4 Data Plane

The **Data Plane** consists of protocols and mechanisms that handle individual packets (extending up to the traditional layer 4 protocols such as TCP and UDP) based on the state that is output by the Decision Plane (i.e. the Data Plane Management-DE). This state includes the forwarding tables, packet filters, link-scheduling weights, and queue management parameters, as well as tunnels and network address translation mappings (Definition adopted but with modification, from the 4D architecture).

Some of the elements of the Discovery Plane may use the services of the Data Plane i.e. they run on top of the Data plane or they may use the Dissemination plane. Elements of the Dissemination Plane need not use the services or run on top of the Data plane. The two planes may be made independent of each other. In today's networks, some protocols that would be qualified as elements of the Dissemination Plane run on top of the Data plane. However, allowing flexibility to dynamically switch between using and not using the Data plane for resilience purposes in the event of adverse conditions or failures is an issue worthy consideration. The properties required of the Data Plane by autonomicity are: within an autonomic node, the view of the Data Plane is that it shall have an associated Decision Element (DE) or specialized instances of DEs that manage the protocols and mechanisms of the plane (see Figure 17). Figure 13 shows the "relative distribution and interactions views" of Data Plane elements. Example elements of the Data Plane i.e. protocols or mechanisms belonging to this plane are: IP Forwarding, Layer2.5-Fowarding, Layer2-Fowarding, Layer3-Switching, Layer2-Switching, etc.

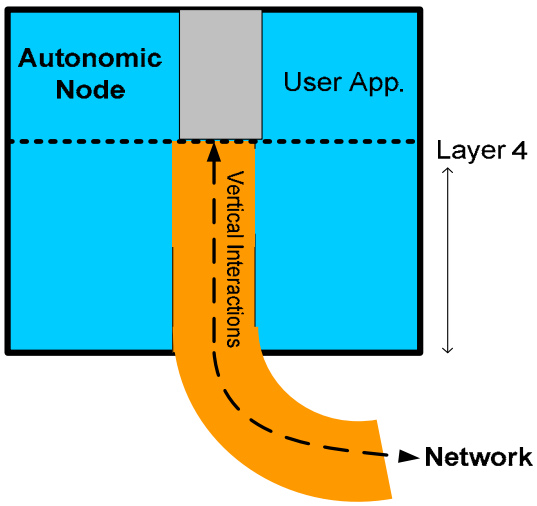


Figure 13: The "relative distribution and interactions views" of Data Plane elements

9 Architectural Principles for specificating Autonomic Behaviours (ABs) of Decision-Elements

This clause describes some key issues that need to be addressed when designing architectural and behavioural features of autonomic nodes and network-wide autonomic behaviours for diverse networking environments, following the outlined GANA architectural principles. The hierarchical levels of control loops and their associated elements, the required interfaces and behaviours, as well as the interactions between control loops need to be specified according to the guidelines for designing control loops of an autonomic system proposed in [i.34] to ensure that the control loops and their interactions are testable and unambiguous. Therefore, the perspectives outlined in this clause are those that should be considered for the production of standardizable specifications of autonomic behaviours of DEs.

9.1 Definition of an Autonomic Behaviour (AB)

In GANA, an **Autonomic Behaviour** (AB), *observable and verifiable on interfaces of an autonomic manager* element (i.e. a DE) is defined as a behaviour or action that may consist of a set of sub-behaviours or sub-actions triggered by a Decision-Element (DE) in an attempt to achieve the goal defined by how the Decision-Making-Element manages a Managed Entity (Entities)-ME(s) under its control (within a Control-Loop Structure). The autonomic behaviour is considered as behaviour of a DE, triggered as a result of reception of information from its information suppliers such as its associated Managed Entity (ies), in an attempt to regulate or reconfigure the behaviour of the Managed Entity (Entities), OR starts as behaviour spontaneously triggered by the DE. A behaviour triggered spontaneously by a DE is simply a spontaneous transition in the Finite-State-Machine describing the overall behaviours of the DE. An example of an autonomic behaviour is: self-description and self-advertisement, self-healing, self-configuration, all triggered by a DE. The decision logic implemented by a DE may have learning, cognitive and planning behavioural characteristics, depending on the nature of the control-loop (whether "fast-control loops" versus "slow types of control loops" that are associated with the need for planning and sophisticated decision making for the long term operation of the system/network).

Therefore, it is important to note that an autonomic behaviour is bound to a DE, and possibly (though not necessarily) to information supply parts of the control loop implemented by the DE together with the Managed Entity (ies) under the control of the DE. In this context, when talking about autonomic behaviour specifications, it means a description of the architecture and functionalities of a DE, which may mean formal specifications of the Finite-State-Machine of the DE.

Designing a DE for a node/device that can operate or assume different roles in diverse networking environments means that the diversity of the network environments and the roles the node can play have direct impact on the way the DE is designed to switch context depending on the environment. On the other hand, diverse networking environments means the information and events specific to the dynamics of the concrete environment also impact on the way a DE and its control loop shall be designed.

9.1.1 High-Level State Transitions of a Decision-Element (DE)

Figure 14 provides high-level state transitions view of a DE. A DE, after it has been started enters a state where it continuously listens for the following types of Signals:

1) Policies or COMMANDs from the Upper-DE (or from a human if this is a Network-Level-DE). Some of the COMMANDS may not be necessarily conveying policies.

2) Events from the Managed Entities (MEs) of this DE.

3) Information from peer and sibling DEs (on the same level in GANA) and from other types of information sources in the environment.

The following are the high-level state transitions involved.

- When the DE receives a COMMAND from the Upper-DE (or from a human if this is a Network-Level-DE):
  - *Its ACTION: {Perform the required Operation e.g. change the behaviour of the Managed Entities (MEs) associated with this DE, re-configure an ME, etc.}*

- When the DE receives Policies from the Upper-DE (or from a human if this is a Network-Level-DE):
  - *Its ACTION: {Apply the policies to the Managed Entities (MEs) associated with this DE}*
- When the DE receives Events from the Managed Entities (MEs) of this DE:
  - *Its ACTION: {(1) Decide how to react to the events and whether an Action should be performed on the MEs, and act accordingly; (2) Decide whether the Events and local Plan of Actions need to be communicated to the upper DE or to peer and sibling DEs (on the same level in GANA) and act accordingly}*
- When the DE receives Information from other Suppliers such as peer and sibling DEs (on the same level in GANA) and from other types of information sources in the environment:
  - *Its ACTION: {Decide whether to change the behaviour of the Managed Entities (MEs) associated with this DE and act accordingly}*

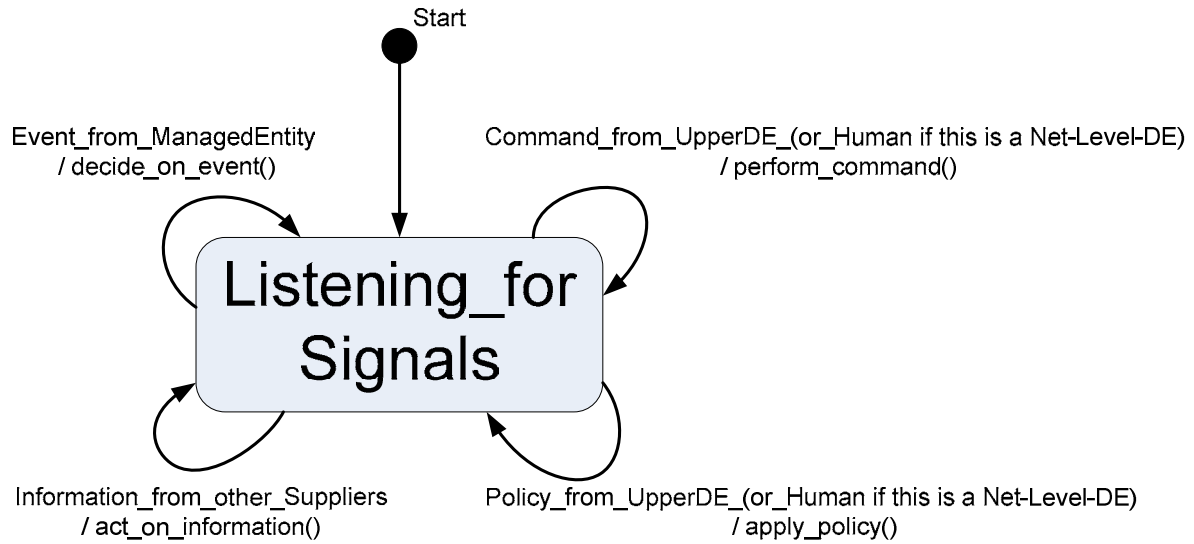


Figure 14: High-Level State Transitions of a Decision- Element

9.2 GANA Control Loops

The separation between the concepts: Decision Element (DE) and its associated Managed Entity(ies) - ME(s) allows to capture and specify the behaviours of the two without looking into implementation issues. The separation does not necessarily mean that the DE and its associated ME(s) shall be implemented as separate run-time entities i.e. separate processes or threads. Both may be implemented as a combined single functional entity, like in the case of some of today's protocols that implement some notion of a control loop such as TCP and OSPF. The DE and its ME(s) may be implemented as two sets of libraries, which can then be linked as a single executable. It should be noted that the complexity of a DE depends on the roles the autonomic node can potentially play, meaning that by design, the DE may implement "specialized braches of behaviours" according to the roles the DE can play i.e. in order to allow for role-based self-management of an autonomic node.

9.3 Protocol-intrinsic Control Loop and its associated Decision Element (DE)

There are some protocols that do implement a *control-loop* within the behaviour of the protocol (Figure 15). In this case, the DE is simply the core reasoning functions of the protocol while the ME is simply the manageable functions (aspects) of the protocol's behaviour. What would be considered when enhancing some existing protocol to make it "autonomic" is enriching some "decision" aspects of the protocol e.g. an IPv6 protocol(s) by either extending the protocol's functionality or allowing the protocol to access more information from potential information suppliers in order to enrich the protocol's intrinsic control loop. This would include adopting such concepts such as *primitives, protocol module abstractions, self-description of capabilities by protocol modules* proposed by the CONMan, that improve or extend the *management interfaces* of protocols (see [i.8],[i.19]). What this means is that some protocol that is considered to be an "autonomic protocol" can be designed in such a modular way that it clearly has a "distinct" separation between its protocol-intrinsic DE and the regulated (managed) functions(aspects) of the protocol that are managed/regulated by the associated protocol-intrinsic DE.

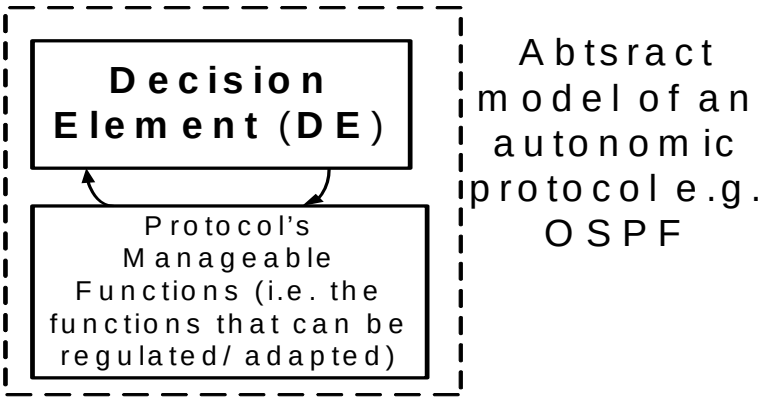


Figure 15: A protocol-intrinsic Control-Loop

9.4 Example of Decision Element (DE)

Figure 16 is an example of a control loop operating on the level of an ***abstracted network function*** namely QoS Management (see Figure 17 for other types of abstracted network functions). The figure shows the issues to be considered towards the production of Autonomic Behaviour Specification(s).

Some of the issues to be considered for adoption in the GANA Specifications regarding Information Suppliers of a DE are: Information Models and information presentation mechanisms (dissemination) to the DE, as well as information formats, with possible adoption and extensions of concepts/ideas from FOCALE, DEN-ng, XML, RDF, SMI (refer to SNMP standards), etc.

NOTE: When it comes to issues related to Managed Entities (MEs), according to state-of-art, a lot has already been achieved regarding protocols and mechanisms that would constitute an ME, such as the one presented for the QoS-Management Control loop above. What remains however, is how to bring all the vital manageable(managed) aspects (e.g. configurable parameters and orchestrations) of protocols and mechanisms into being managed through some Management Interface by a corresponding Decision Element that drives the control loop in order to enable autonomicity.

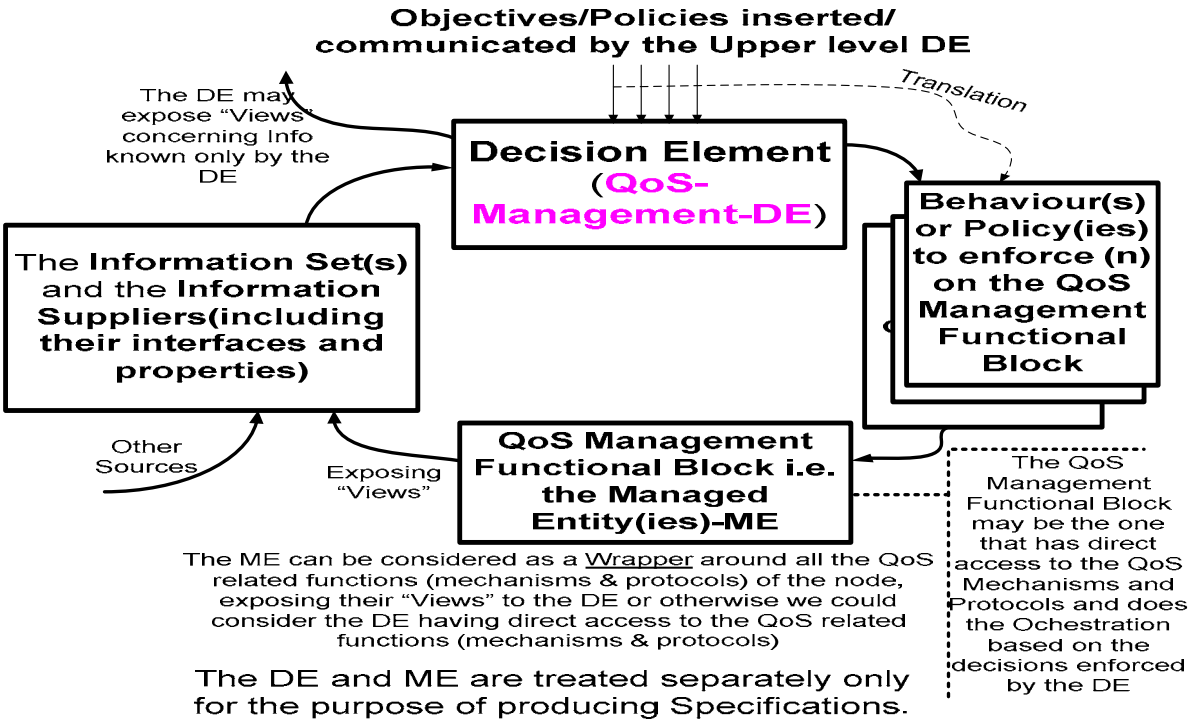


Figure 16: A Control-Loop specific to an abstracted "Networking Function" - QoS Management, and some of the key items that call for "Design Specifications"

9.5 Structure of a GANA node and related DEs Hierarchy

This clause describes the structure of a GANA Node. A GANA Node does not prescribe an implementation though the structure can inspire a derivation of an implementation architecture. A GANA Node is a "virtual node", meaning that the actual types of DEs that are instantiated in a specific type of a device architecture e.g. an mobile terminal, a fixed end-system, a router or a switch, vary for each type of device (since it depends on the types of Managed Entities (MEs) supported and the role the device can play in the network, etc.).

Figure 17 and Figure 20 show the hierarchical relationships and sibling relationships among Decision Elements inside an autonomic node. The abstracted functions whose Decision Elements are named on the figure, represent some of the levels of autonomicity (control loops) for which specifications can be produced. Multiple instances/specializations of each of the DEs below the Node-Main-DE may be specified and designed according to the need to separate issues of concern, and narrowing complexity by modularization. Other types of DEs, apart from the ones defined on Figure 17, such as "Security-Management-DE" need also to be defined for the autonomic node.

Figure 18 and Figure 19 present the structure of a node/device devised in accordance to GANA, showing the layers at which Autonomicity/Control-Loops can be introduced. The figures also reveal that the Decision Elements that need to take decisions at the level of the node as a whole need to be considered, and thus designed, as sub-DEs of the Node-Main-DE. Such DEs are Fault-Management-DE and the Resilience-and-Survivability-DE, Security-Management-DE, and AutoDiscoveryAndAutoConfiguration-DE.

NOTE 1: The core functionalities (extensible) of some various DEs can be found in deliverables from the EC funded FP7 EFIPSANS project (downloadable from http://www.efipsans.org/), in which the specification and validation of the DEs were documented. Of course, the community can extend the functionalities. Example references include: [i.43], [i.44], [i.60], [i.61] and [i.62].

Referring to Figure 18 and Figure 19, the place-holders for internal control-loops (inside a Network Element) depicted by the Reference Model enable to design and embed "node-local" Self-Management behaviours/algorithms, including node-local Self-Optimization, i.e. some degree of network element intelligence through the internal Decision Elements (DEs) that realize the internal control-loops. Example node-scoped Self-* behaviours that do not necessarily require collaboration/negotiation with other network elements include: Plug-n-Play; Energy Savings through autonomic functions; Autonomic Security Management (self-protection and self-defending behaviour); Autonomic Fault-Management and Resilience (proactively and reactively), etc.

NOTE 2: The management aspects and Control-Loops that are realized by the sub-DEs of the Node-Main-DE are "superior" to those at Level-2 (Function-Level) and below, because what happens on the Node-Level is supposed to globally affect the whole node. The figure also shows the Service Layer view in GANA with the incorporation of the Service-Management-DE. Therefore, on the Function-level, the following DEs are required, with each DE autonomically managing its assigned Managed Entities (MEs): Service-Management-DE, Monitoring-DE, Mobility-Management-DE, QoS-Management-DE, DataPlane-and-Forwarding-Management-DE and the Routing-Management-DE. The core functionalities (extensible) of some various DEs presented in the figures can be found in deliverables from the EC funded FP7 EFIPSANS project (downloadable from http://www.efipsans.org/), in which the specification and validation of the DEs were documented. Of course, the community can extend the functionalities. Example references include: [i.43], [i.44], [i.60], [i.61] and [i.62].

NOTE 3: The interfaces between the DEs are omitted for the purposes of presenting a simplified picture. The DEs in the node may communicate using Information/Knowledge Sharing Repository for sharing info/knowledge among functional entities (e.g. DEs) within a node (presented in Figure 19). DEs may also communicate through the Interfaces defined for a model of a Decision Element that is presented later in Figure 23.

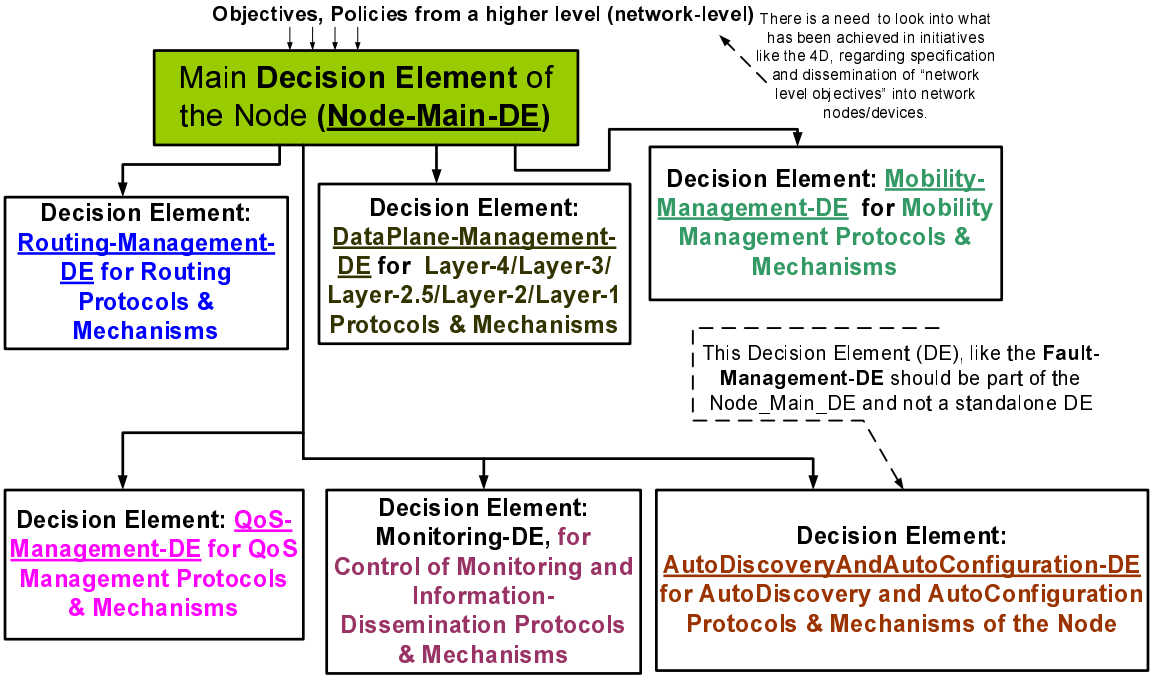


Figure 17: Example GANA Decision Elements (DEs) of "abstracted networking functions" and Function-Level-DEs as forming sibling relationships with each other-with Node-Main-DE as common "parent"

NOTE 4: According to the concept of "ownership", a protocol-level-DE (i.e. a protocol with an intrinsic control-loop structure) communicates (in terms of management and control) with the Function-Level-DE that manages it together with other protocols and mechanisms abstracted by the "Function", as the Managed Entities (MEs) of the particular Function-Level-DE, and not with another Function-Level-DE. This subject is covered in the clause 9.11.5 that addresses Assignment of Managed Entities (MEs) and their configurable and controllable Parameters to specific Decision Elements (DEs) in GANA i.e. the concept of "ownership" in GANA.

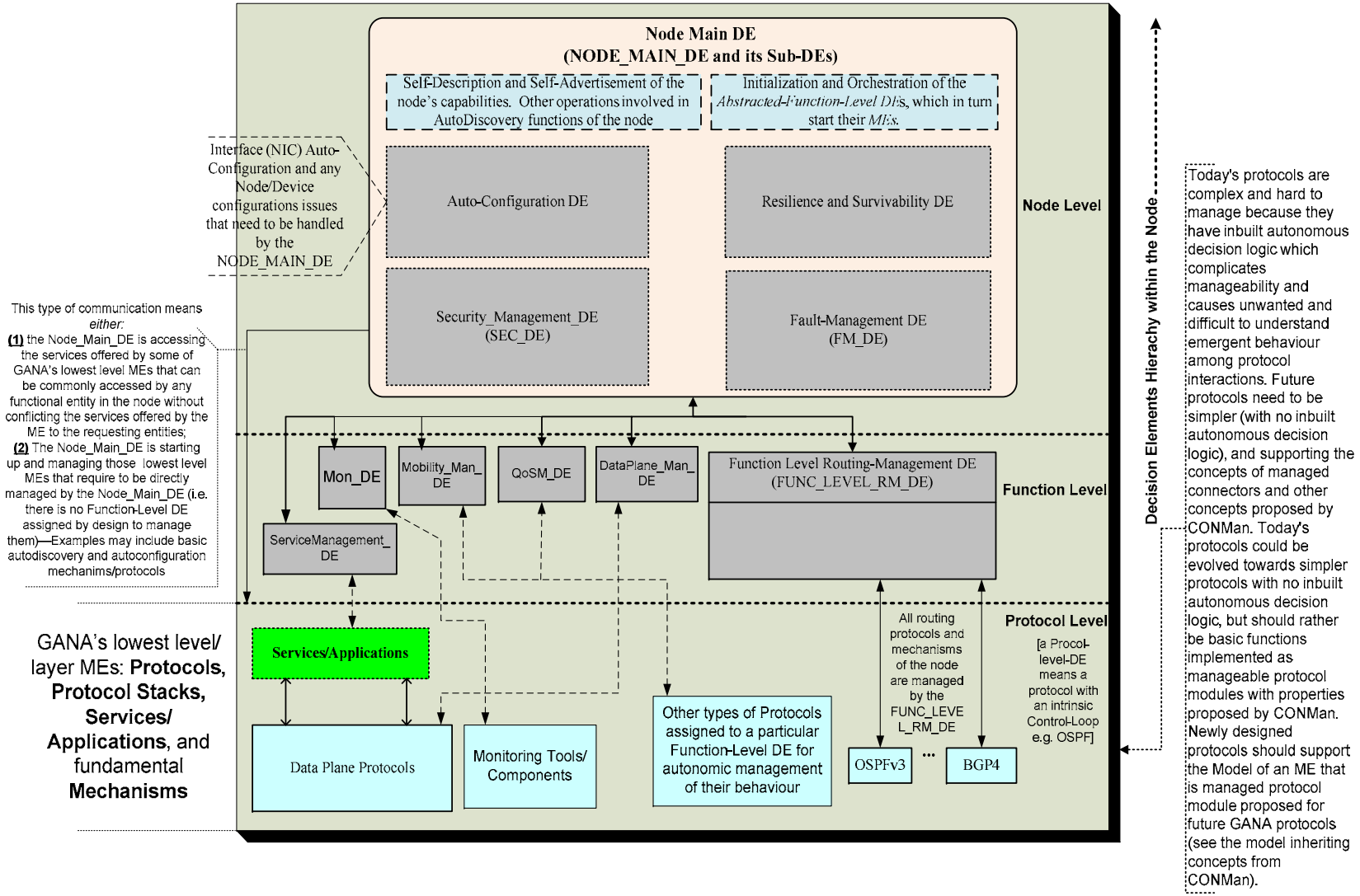


Figure 18: Structure and Composition of a GANA Node and an illustration on how to instantiate GANA for Autonomic Routing in IPv6 Network

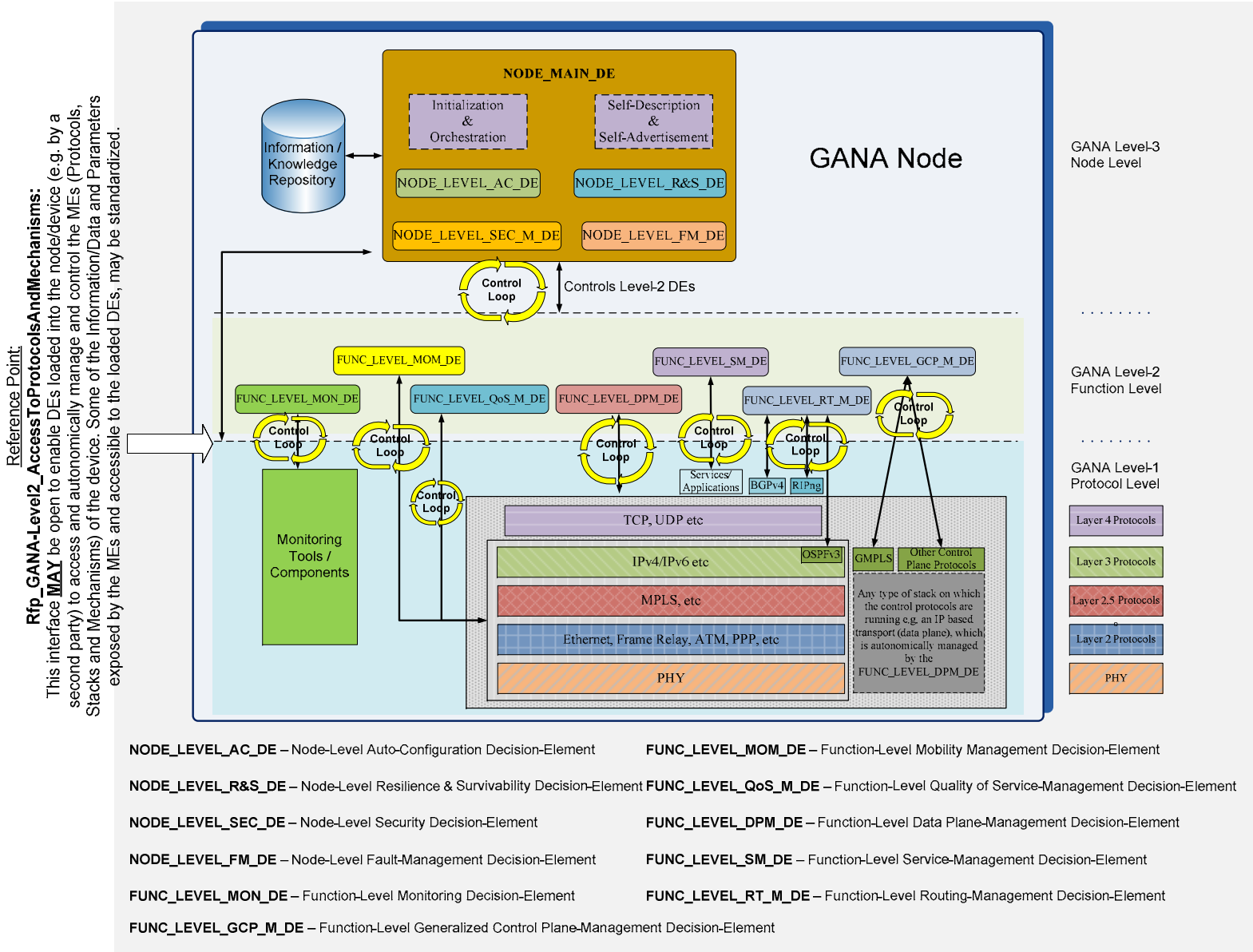


Figure 19: Expanded view of the GANA node structure, the Decision Plane and Control Plane views and example assignments of DEs to some protocols, stacks and mechanisms as Managed Entities (MEs)

9.6 The Internal Interface/Reference Point within a GANA Node

9.6.1 Specification (High level guidance)

Decision Elements (DEs) in GANA are containers of Algorithms that determine autonomic behaviour i.e. Self-* properties realized by a DE (see definition of Autonomic Behaviour in GANA) and such Algorithms per-se cannot be standardized. The algorithms require information/data to be communicated to and from the entities that communicate with a DE. Different types of algorithms require different types of data to flow on the interfaces of a DE. The GANA Model offers a clear framework by which DEs from a second party can be loaded into a device (if the vendor supports this in a somewhat similar fashion to the emerging Software Defined Networking (SDN) Paradigm). At the same time, the DE Model supports the loading of a control strategy that can be expressed as a Run-Time Executable Behavioural Model (specified and provided as input) i.e. an algorithm in some sense, the DE can interpret and execute at initialization and during its operation. For example, there may be different types of algorithms that can determine the logic and behaviour of say the DataPlane-and ForwardingManagement-DE. This requires that the interface to the protocols and mechanisms of a device should be open to allow Function-Level-DEs to access and autonomically manage and control protocols and mechanisms of a device. This means at any time during the operation of a device, DEs may be loaded and unloaded by the operator. Practically, in terms of the communication between a Function-Level-DE and protocols and mechanisms, CLI or SNMP based methods could be used or a new type of a unified API could emerge and get standardized.

The internal Reference Point within GANA node is indicated and further described in Figure 19.

AFI does not mandate the implementation of the internal structure as captured by the concept of a GANA Node Structure. The internal structure serves primarily to offer some guidelines on how device-internal hierarchical control-loops (i.e. nested control-loops) can be designed. At the same time, it is aimed at showing that, potentially a customized "decision-making-logic i.e. behavioural-model" that governs the behaviour of some Decision-Element supposed to operate inside a device(network element) could be loaded during deployment and network operation time. Such an loaded logic would implement a control-loop that dynamically adapts the behaviour of its specifically assigned Managed Entities (e.g. protocols, stacks, mechanisms).

The GANA Node structure does not prescribe an implementation per-se, though the structure can inspire a derivation of an implementation architecture. Such a structure is primarily aimed at helping separate "specification and description aspects for design models of autonomic elements (i.e. DEs)" from their actual "implementation-oriented aspects", by employing "abstractions" and a "modular design pattern"-thereby presenting a somewhat imaginary (i.e. virtual) autonomic node model. This helps a designer (e.g. AFI or designers within an equipment manufacturer organization) to produce generic specifications of autonomic elements (DEs) and their behaviours, and simulate and validate them without bothering about the actual implementation strategy that can then be taken as the second step. One would perceive the aspect of moving from an imaginary (i.e. virtual) autonomic node captured by its "implementation-agnostic" autonomic behaviour specifications of internal DEs, to its actual implementation as a second step after simulating and validating modular design specifications.

9.6.2 Implementation (high level guidance)

The implementation stage for DEs then takes into account the required resources, the possible need to merge some of the autonomic elements (i.e. internal DEs) and interfaces from the modular "implementation-agnostic" specifications, into some monolithic implementation or a modular implementation that strictly follows the GANA Node structure model (i.e. the simulated/validated specifications). All these issues belong to "implementation issues" and thus implementation issues should be left out from the generic design specifications of the autonomic elements (i.e. internal DEs), interfaces and behaviours. For example, during the implementation of the design models of DEs, one may implement them in a one-to-one mapping such that the DEs run as separate parallel processes or threads, or one may merge the design models of some DEs (e.g. the Finite-State-Machines describing individual DEs) and implement the resultant merged design models to run as monolithic single process or thread instances.

Thanks to the concept of GANA Node Structure, AFI is seeking to produce some selected specifications and design of some internal control-loops associated with some DEs, for the purpose of demonstrating some node-scoped control-loops (fast control-loops) in contrast to the outer slower control-loops operating on the network-level i.e. those realized by the Knowledge Plane of which AFI will also design some viable outer control-loops for demonstration. For example, AFI may consider designing some internal control-loops for the autonomic management and control related to the following "partitioned" aspects: "Autonomic-QoS-Management", "Autonomic-DataPlaneAndForwarding-Management" and "Autonomic-Security-Management". Other internal control-loops aimed at putting some degree of "intelligence" that may also be considered are for autonomic management and control of the ControlPlane Protocols and Mechanisms, for autonomic management and control of Monitoring Mechanisms of a network element, etc. All this will serve to help manufacturers identify the type of autonomic behaviours that can be embedded into the network element that make the network element react "autonomously" without requiring the influence of the Network-Level DEs of the Knowledge Plane.

NOTE: However, AFI will give more focus on the outer control-loops realized by the Knowledge Plane (Network Level).

As already mentioned earlier Decision Elements (DEs) in GANA are containers of Algorithms that determine autonomic behaviour i.e. Self-* properties realized by a DE, and different types of algorithms require different types of data to flow on the interfaces of a DE. For example, some optimization algorithms may only be developed for a specific topology that a network operator has finally chosen to build i.e. the algorithms are customized to a particular emergent topology.

There are two ways the equipment manufacturer may facilitate the loading of "customized DE algorithms", but again, AFI does not mandate this since this is up to the manufacturer (AFI is only communicating the possibilities that can be considered for today's evolving networks, the future network evolutions, or future network architectures in general).

1) Supporting Loadable DEs: The GANA Model offers a clear framework by which level-2 and level-3 DEs (if not hard-wired into a network element by the manufacturer), from a second/third party, may be loaded into a device.

2) Manufacturer-Hardwired DEs that support Run-Time Loadable Executable Behavioural Models for interpretation and execution by the DEs: A DE Model (see the model in clause 9.11) supports the loading of a control strategy that can be expressed as a Run-Time Executable Behavioural Model i.e. an algorithm in some sense, which can be interpreted and executed by a particular DE hard-wired by the manufacturer, at initialization time of the device(DE) and during its operation. The loaded Behavioural Model would govern the behaviour of a particular DE. Loadable Run-Time Executable Behavioural Models can be specified using some formalism of choice such as Finite-State-Machines or State-Charts (refer to the UML language). This means, the internal DEs hardwired by the manufacturer can be designed to also function as "interpreter" of loadable Run-Time Executable Behavioural Models (specified and supplied as input for execution). A DE, as an interpreter, could achieve this say in a somewhat similar way a Java VM interprets and executes byte-code). As described in the present document, Behavioural Models can be used for more advanced control/programmability strategy that complements Policy-Based Control (management).

For example, there may be different types of customized algorithms that can determine the decision-logic and behaviour of say the "DataPlane-and-ForwardingManagement-DE" that autonomically configures and adaptively controls the DataPlane and Forwarding protocols and mechanisms. This requires that the interface to the protocols, stacks and mechanisms of a device may be open to allow Function-Level-DEs (GANA Level-2 DEs) to access and autonomically manage and control protocols and mechanisms of a device. This is what is referred to by the Internal Reference Point that could be introduced by manufacturers in the future. This means at any time during the operation of a device, DEs (if not hard-wired by the manufacturer) or alternatively Behavioural Models may be loaded and unloaded by the equipment operator. Practically, in terms of the communication between an internal Function-Level-DE and "protocols, stacks and mechanisms", existing CLI or SNMP based methods could be used by the Level-2 DEs, or a new type of a unified API could emerge in the future and get standardized.

In addition to these aspects, according to the Network Governance Interface used by the Operator to interact with an Autonomic Network (described in clause 11), the concepts of Network Profile, Sub-Profiles, Encapsulation of Policies, Profile Continuum and Policy Continuum, all apply.

9.7 Cross-Layering in GANA

1) Cross-Layer Information should be communicated to ALL the DEs at the Functions-Level to enable individual decisions and self-adaptive behaviours in the DEs.

2) Since Function-Level DEs are assigned to specific Managed Entities (MEs) and are supposed to ALL know the goals/objectives and policies that apply at their level in GANA, the DEs influence each other in configuring or adjusting the configuration of their respective MEs with respect to fulfilling Cross-Layer communications and requirements. If this approach that involves Level-2 DE collaborations is not preferred then their co-ordination can be achieved at the upper level i.e. at the NODE-LEVEL.

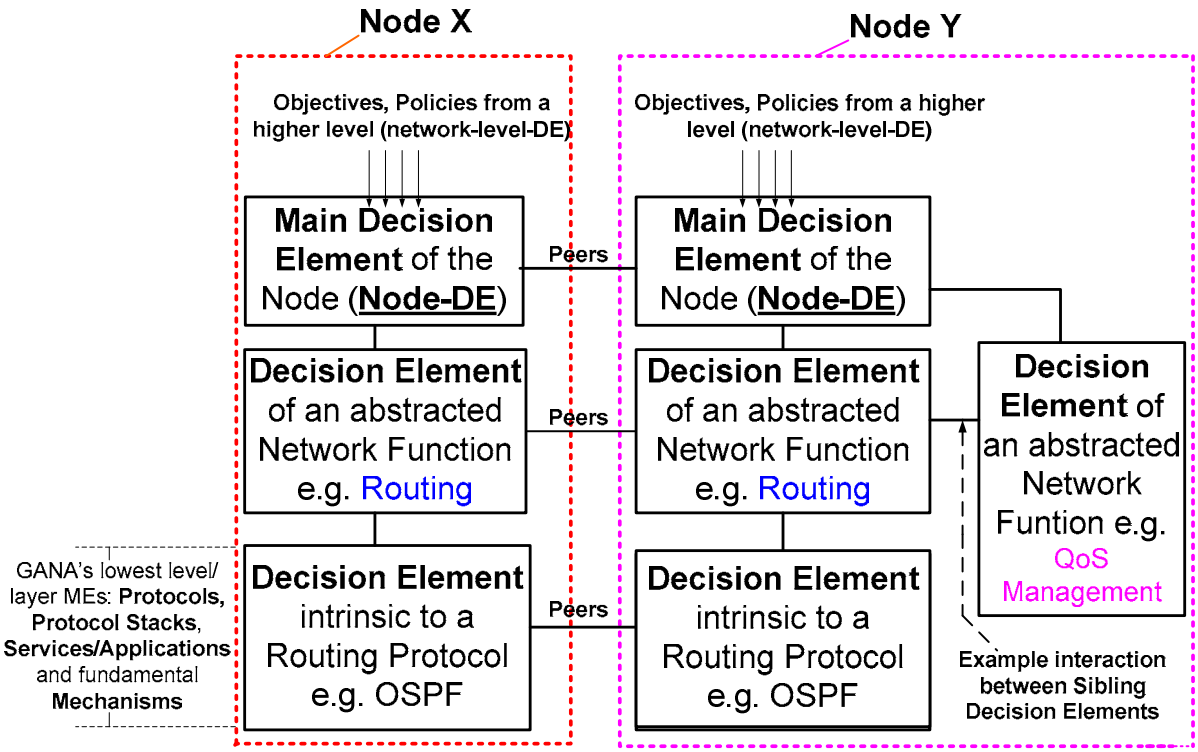
9.8 Combining Centralized and Distributed Decision-Making based Management in GANA

9.8.1 Interfaces specification

This clause describes the guidelines on how to combine decentralized autonomic management and centralized management, and point the "hooks" required to achieve this within the GANA framework, while pointing out the main limitations of decentralization in network management. Main limitations elaborated later in this clause, include: resource limitations in network nodes/devices which limits their ability to perform extensive data mining and information analysis while expected to route and forward traffic at the same, and the problem of longer convergence time associated with some distributed decision-making processes. Clause 8 contrasts classical approaches to network management (non-autonomic), to the GANA based autonomic management and control, and provides a deep insight into the evolution of classical network management approaches and frameworks as impacted by the GANA based autonomic management and control paradigm.

Figure 20 depicts the hierarchical, peering, sibling relations and interfaces of Decision Elements that are calling for specifications-for Function-Level DEs, Node-Main DE and Network-Level DEs. It shows how the Decision Plane inside a node/device can be navigated from lowest level type of DEs in GANA, namely autonomic protocols-protocols with intrinsic control-loops.

As already described in Types of Control Loops in GANA (clause 8.2.1), the next level of self-manageability (autonomicity) after the "node level", described above, is the "network level". The "network-level" represents the last level (top-level) of autonomicity i.e. self-management. There may exist a logically centralized network-level Decision-Making-Element(s) or the kind of DEs proposed in the 4D network architecture [i.25] and [i.23] which are referred to as Network-Level DEs in GANA, belonging to an isolated "decision cloud" i.e. outside the nodes that knows (through some means) the objectives, goals or policies to be enforced by the whole network. The objectives, goals or policies may actually require that the main (top-level) DMEs/DEs of the nodes of the network that are covered by the centralized network-level DE(s) of the "decision cloud", export "views" such as events and state information to the centralized DE(s). This may happen in order for the centralized DE to influence or enforce the DEs of the nodes to take certain desired decisions following specific network policies that may in turn have an effect of inductive decision changes on the lower level DEs of individual nodes i.e. down to protocol level decisions.



NOTE: All the Types of DE Interfaces depicted illustrate the need for "node/device-intrinsic management" and "network-intrinsic management or in-network management" in Self-Managing Future Networks.

Figure 20: Hierarchical, Peering, Sibling Relations and Interfaces of DEs in GANA

The following bullet points describe the various ways of combining both centralized control and distributed control so as to have the complementary benefits of both approaches in designing control-loops and their interactions (via the DEs realizing the control-loops and autonomic management and control of their assigned Managed Entities(MEs)):

- A distributed network-level control-loop may be implemented following the above set-up (DE-to-DE peer exchange of information and negotiations on how to adapt the behaviours of the Managed Entities (MEs) of the collaborating DEs, in collaborative self-optimization or self-healing behaviour), while another case of implementing a distributed control-loop would involve the main Decision-Making Elements of nodes (Node-Main-DEs) working co-operatively to self-organize and manage the network without the presence of a "specially instrumented" logically centralized network-level DME(s) operating in an isolated "decision cloud, namely the Knowledge Plane".

- A logically centralized network-level DME(s) (i.e. in the Knowledge Plane) would be meant to manage the whole network and should be hosted in a special machine(s) whose resources are only dedicated to network state data analysis and management operations (in harmony with the self-management happening at node-level and below, for the fundamental (i.e. normal/conventional) network nodes).

- "in-network" or "network-intrinsic" management. This second case implies that the fundamental network nodes need to be designed in such a way as to have the possibility for the nodes themselves to self-organize and perform "in-network" or "network-intrinsic" management-which can only be achieved to a certain extent due to resource limitations of the nodes and the problem of longer convergence time with some distributed decision-making algorithms. The subject of "network-intrinsic" i.e. "in-network" management is covered in more detail in clause 11.10.

9.8.2 Illustration of interaction between fast control-loops in an NE and the slower outer loops in the Knowledge Plane, using "routing function" case

In a fixed/wired network, two kinds of "mirror-DEs" and their associated Control-Loops for a given abstracted network function such as Routing-Management, Forwarding-Management or QoS-Management are required in order to fully implement the autonomicity of the function under consideration This approach (set-up) is illustrated with an example applied to the case of routing described below, which paints a picture on how the Routing Function in individual node architectures, and for the network architecture as a whole, can be made autonomic using two kinds of Routing-Management DEs and their associated Control-Loops, which mirror each other and work co-operatively.

The Routing Functionality (Function) of nodes in an IPv6 based fixed network and the network as whole can be made autonomic by making diverse Routing Schemes and Routing Protocol Parameters employed and altered based on network-objectives, changes to the network's context and the dynamic network views in terms of events, topology changes, etc. Figure 21 depicts how the routing behaviour of a node/device and the network as a whole can be made autonomic. The figure also shows that the Network-Level-DEs are part of what is called the Knowledge Plane (for more information on the subject refer to the later clause: Cognitive Networking and Knowledge Plane as part of the GANA Decision Plane, and Information/Knowledge Sharing).

Two types of Control-Loops are required for managing/controlling the routing behaviour. The first type is a node-local control loop that consists of a Function-Level Routing-Management DE embedded inside an autonomic node e.g. a router. The local Function-Level Routing-Management-DE is meant to process only that kind of information that is required to enable the node to react autonomically and autonomously (according to some goals) by adjusting or changing the behaviour of the individual Routing protocols and mechanisms required to be running on the node. The Function-Level Routing-Management DE reacts to "views", such as "events or incidents" exposed by its Managed Entities (MEs) i.e. the Routing protocols or mechanisms.

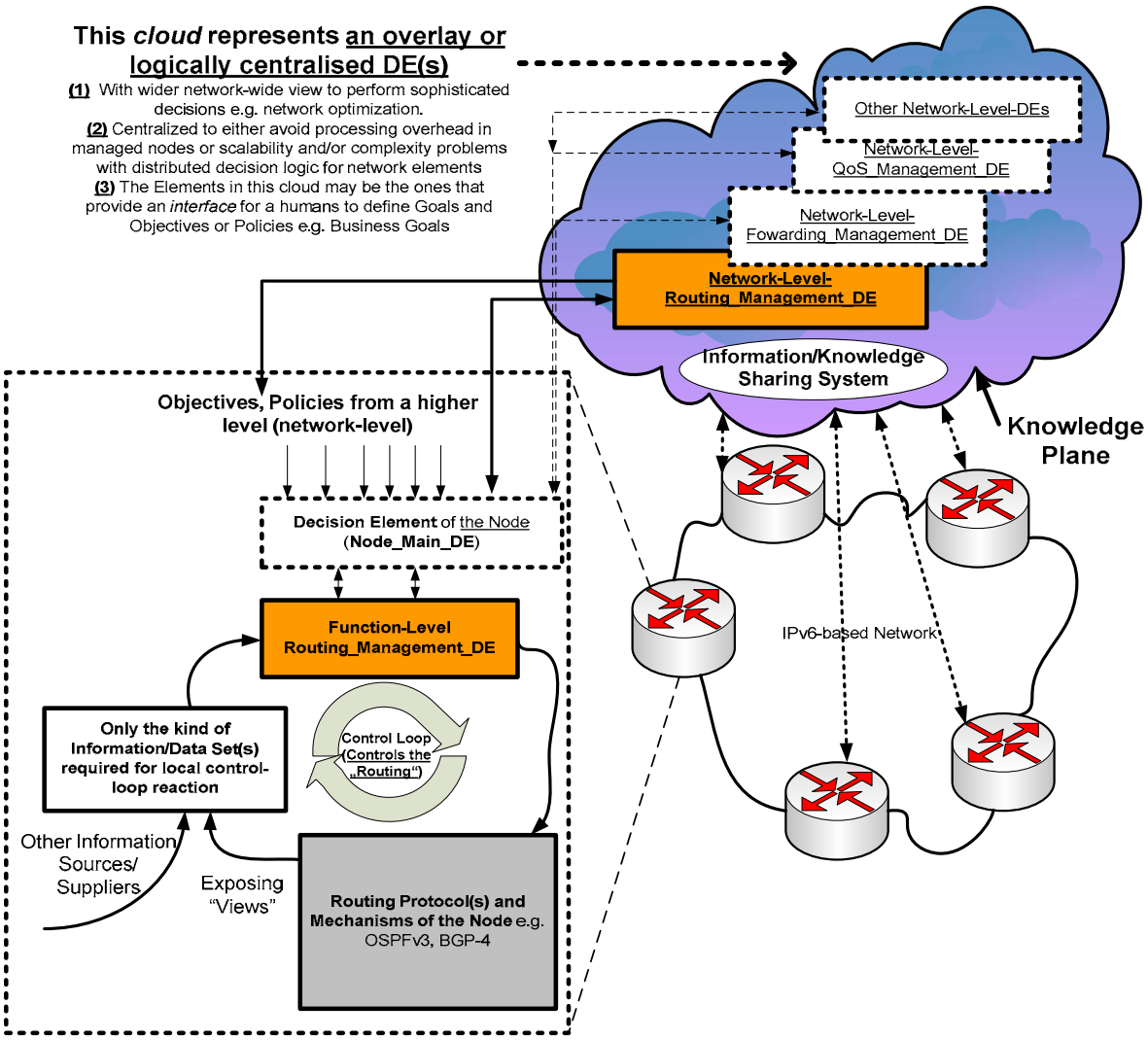


Figure 21: Autonomicity as a feature in Routing Functionality in a IPv6 based network

Therefore, the Routing-Management-DE implements *self-configuration and dynamic reconfiguration* features specific to the routing functionality of the autonomic node. It is important to note that due to scalability, overhead and complexity problems that arise with attempting to make a Routing-Management DE of a node process huge information/data for the control loop, a logically centralised Decision Element(s), may be required, in order to relieve the burden. In such a case, a network-wide *slower Control Loop* is required in addition to the *faster* node-local control-loop both types of loops working together in controlling/managing the routing behaviour in an autonomic way.

Therefore, both types of control loops need to work together in parallel via the interaction of their associated Routing-Management DEs (one in the node and another in the realm of the logically centralised network overlay decision making elements). The node-scoped (node-local) Routing-Management-DE focuses on addressing those limited routing control/management issues for which the node needs to react fast (the faster control loop). At the same time, it listens for control from the network-level Routing-Management DE of the outer slower control loop, which has wider network-views and dedicated computational power, and thus is able to compute routing specific policies and new parameter values to be used by individual routing protocols of the node based on the wider network-views it knows. The Network-level Routing-Management DE disseminates the computed values/parameters to multiple node-scoped Function-Level Routing-Management DEs of the network-domain that then directly influence the behaviour of the targeted Managed Entities (MEs) - the routing protocols and mechanisms within a routing node. The interaction between the two types of Routing-Management DEs is achieved through the NODE_MAIN_DE of a node which verifies those interactions against the overall security policies of the node. The node-scoped Routing-Management DE also relays some "views" such as "events or incidents" to the network-level Routing-Management DE for further reasoning.

The hierarchies of DEs in GANA imply that the higher we go up the hierarchy, the broader the global knowledge required by a DE to take decisions on managing/controlling its associated MEs (which may in turn inductively trigger actions on lower level MEs down to the level of protocols and mechanisms).

As discussed in the present document (see clause 9.13), in GANA, DEs at the Function-Level (level-2), Node-Level (level-3), up to the Network-Level (level-4) are the ones that should have the Cognitive Properties. Also, the notion of "planning", seems to make sense in a Node-Level-DE and a Network-Level-DE. However, having said that, at the Function-Level: Cognition *should be very simple and limited if it is to be allowed, because the Control-Loops at this level need to be "fast control loops" in contrast to "slower upper loops"*.

Moreover, as will be described in more detail in the clause 11.5, it is important that GANA incorporates the principles of reducing the load on upper management layers (DEs) and the data sent northbound to upper DEs, since some degree of self-management and control should be realized at lower GANA Levels of self-management within node/device architecture.

9.9 Reference Points (Logical Interfaces) within Network-Level-DEs interactions

Figure 22 describes the Reference Point between Network-Level-DEs, as well as some aspects that need to be understood regarding the deployment of Network-Level-DEs, such as redundancy for resilience, etc.

The interfaces between the DEs (see the common type of **Reference Point** on Figure 22) are logical interfaces. Their realization is an implementation issue: Network-Level DEs may be hosted by the same physical machine, but they need to be "**logically centralized not physically centralized**" i.e. they need to be replicated in different hardware for redundancy and resilience. What is important is to capture and specify is the kind of information/data (i.e. Characteristic Information) that need to be exchanged between DEs. Examples include events coming from the individual Protocols and Mechanisms (i.e. MEs) assigned to specific GANA-Level-2 (Function-Level) DEs in nodes (performing autonomic management and control by the faster control-loops). Other Network-Level-DEs missing on the figure include: **Network-Level-Mobility-Management; Network-Level-Fault_Management_DE;** and others that may need to be defined.

The DEs need to communicate the following type of Characteristic Information (using some protocol/mechanism):

1) "Views" such as Policy changes by the human operator; challenges to the network's operation from the perspective of a particular DE e.g. events, detected faults, threats, etc.; "views" communicated from lower - Level DEs in nodes that require Net-Level DEs to know and share.

2) Negotiations and Synchronization of Actions and Policies.

A summary of all relevant Reference Points and Characteristic Information exchanged in the GANA Model is given in clause 13.

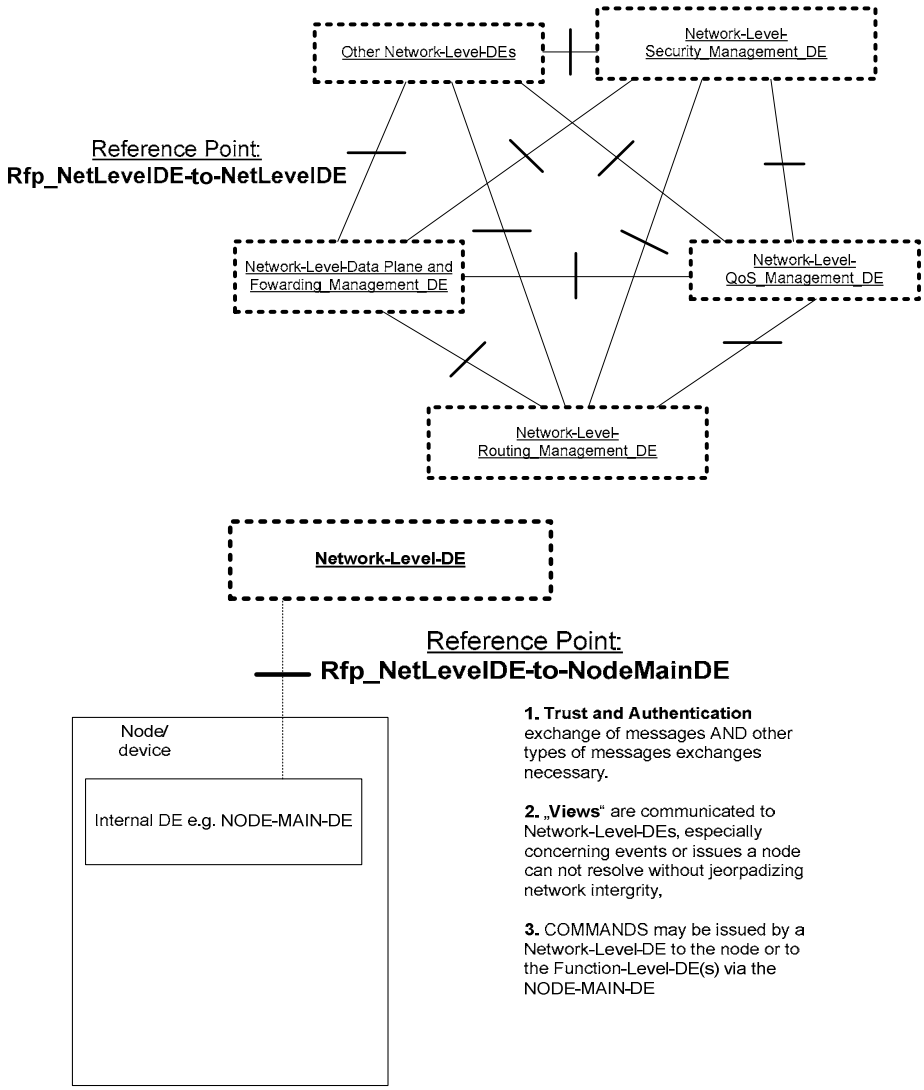


Figure 22: Reference Points/logical interfaces of DEs external to a GANA-Node

9.10 What needs to be standardized in the Autonomic Behaviours (ABs'): implementation guide

This clause describes the aspects and items to consider when producing **"Standardizable" Specifications in the scope of AFI Work Item#3** (i.e. not in the present document). When talking about standardizable "autonomic behaviours" we refer to the fundamental behaviours of the Functional Blocks for autonomicity and self-management during the process of Auto-Discovery and Self-Configuration of network elements in a Plug-n-Play fashion. That includes how the DEs discover network entities and the network objectives/goals, profiles, policies and configuration-data they need for the configuration of the network elements and the network as a whole. The clause on Network Governance covers the subject. The specification also refer to "behaviours" of Autonomics Functional Blocks such as DEs in their participation in some "in-network" collaborative behaviours through DE-to-DE Peer communications that enhance the Control-Plane with exchanging some information (e.g. protocol or network related events and statistics) or negotiation messages that enable the participating network elements/nodes (possibly Hop-by-Hop along an E2E path or not necessarily hop-by-hop i.e. not on-link neighbours) to perform some collaborative network optimization (i.e. possibly the "minimum" required of a network by the manufacturers as some "agreed basic behaviour"). In some way, this can be viewed as realization of distributed control-loops spanning some network elements within the E2E transport network. Enabling to realize distributed control-loops and possibly some basic optimization behaviours would need to be discussed by equipment manufacturers to see what sort of collaborative behaviours or enablers should be standardized. The types of distributed control-loops and the types of participating network elements should be discussed and established. All these aspects described above are a subject coverered in AFI WI#3 (out of scope of the present document). Having said all this, there are DE Algorithms that would provide Vendor Differentiation.

9.10.1 Elements of a Control Loop

This clause describes a DE and its corresponding Information Suppliers, Interfaces and their Properties:

- The Decision Element (DE) and its Behaviours. The specification considers DEs to form a hierarchy, starting from a protocol(s) that intrinsically implements the concept of a control loop (lowest level) (see Figure 15 and Figure 20).
- The Monitoring Information & Other Types of Information required by a DE to operate on, as well as the information suppliers (see Figure 7 through Figure 10 and Figure 11) for types of potential information suppliers.
- The Managed Entity (ME)-managed by a DE (the key focus is about manageability aspects e.g. towards composable stacks/services-"on-demand stacks"), support for policy-injection/enforcement, export of "views" i.e. information such as internal state-info or knowledge acquired or learnt from interaction with other entities, all exposed to the DE managing the ME.
- Interfaces between the DE and Information suppliers other than its associated ME(s).
- Interfaces between the DE and the associated ME(s) i.e. the sensory and effectors parts of the "management interface" of the ME.
- Interface and interaction between the DE and another DE above, across or below it i.e. a lower-level DE(s) is viewed as an ME by the upper DE, otherwise it is a sibling of the considered DE i.e. both share an upper DE as the superior DE. Therefore every managed DE, according to DE-hierarchies (see Figure 20), like every ME, shall support manageability principles i.e. support for policy-injection/enforcement, export of "views" i.e. information such as internal state-info or knowledge acquired or learnt from interaction with other entities, to the DE managing it.
- Interaction between peer DEs within a node or across the network.
- Interaction between a DE and an ME that is not directly under its control i.e. the case when the ME is considered by the DE simply as an information supplier and not being its own ME or that the DE uses some service(s) offered by such an ME that falls under the control of another DE.
- What about the interaction between MEs associated with different DEs? Obviously, MEs associated with different DEs interact with each other as may be already defined by today's static/fixed (non-composable) protocol stacks. However, where flexibility (composability support) is possible e.g. on-demand stacks/tunnels the DEs may need to interact with each other in order to influence (decide) the interaction between their associated MEs, including stacking/tunnelling and bindings.
- Distributed Control-Loops are achieved through the interactions between Distributed Decision-Elements (DEs) OR via a logically centralized overlay of Decision Elements operating on the network-level (i.e. in the Knowledge Plane, see later) and managing the nodes of some network scope like in the case of the 4D or FOCALE. Note that in the first case the decision-making process is distributed while in the second case (the case of 4D or FOCALE), the decision making process is somewhat centralized but in both cases what would be considered Managed Entities are all distributed entities.

NOTE: In order to have a common understanding on how to engineer autonomic behaviours for autonomic networks a Meta-Model for Engineering (designing) Autonomic Networks that captures the concepts such as control loops, Decision Elements, interfaces, relationships, properties and constraints among concepts is therefore required, and currently none exists. In order to address this issue, a GANA Meta-Model was designed [i.20], that can be further evolved by AFI.

9.10.2 Grouping the Autonomic Behaviour Specifications

This clause describes the way to group behaviour specifications into "Groups of Behaviours for the autonomic node". Grouping helps in structuring the Specifications of the Autonomic Behaviours according to say "phases" in the lifetime of the autonomic node operation, e.g. from boot-up time (phase), to operation and self-optimization phases. The grouping would also be helpful during the time of writing Test Suites for Testing the Autonomic Behaviours and understanding the order of execution of Test Cases. Grouping of Autonomic Behaviours may be done as follows:

- Behaviours executed by a node's main Decision Element (NODE_MAIN_DE) and its lower level DEs when an Autonomic Node is initializing (booting up) i.e. behaviours of all DEs and other Functional Blocks during the process of Auto-Discovery and Self-Configuration of network elements in a Plug-n-Play fashion.
- Behaviours executed by DEs when an Autonomic Node is shutting down its service offering components.
- Other types of Grouping would need to be defined.

9.10.3 Specification aspects for Control-Loops

This clause considers the types of Control-Loops that need to be considered for specifications.

1) Control Loops considered intrinsic within certain protocols (for both, existing protocols and any newly designed protocols considered autonomic according to the definition of a Decision Element provided within the GANA architectural framework). As said earlier, there is growing opinion, however, that future protocols need to be simpler (i.e. with no decision logic embedded) than today's protocols which have become too hard to manage due to intrinsic decision logic which may interact in an undesired way with decision logic of other protocols. This means, as explained in [i.25], [i.8] and other references such as [i.44], there is a need to rather implement decision logic at a level higher (i.e. outside the individual protocols).

2) Control Loops that should be implemented by some "Abstracted Network Functions" such as Routing-Management, DataPlaneManagement-and Forwarding-Management, QoS-Management, Mobility-Management, Fault-Management, etc. (see Figure 17).

3) A node's Main Control-Loop that controls the behaviour of the node as a whole.

4) The Decision Element (NODE_MAIN_DE) governing the overall behaviour of a node and shall have the ability to access the "views" maintained by all low-level DEs and their associated MEs and to manage lower level DEs directly below it (see NODE_MAIN_DE in Figure 18 and Figure 19).

9.11 Models of a Decision Element and a Managed Entity (ME)

This clause presents the Model of a Decision Element (DE), Models of Managed Entities (MEs) at GANA's lowest layer (the level where protocols, stacks and mechanisms), and Interfaces. On the interfaces of the Models, some of the Operations/Primitives that facilitate for Adaptive Control, Programmability, Policy Control and Loadability of Control Logic by an upper manager entity (i.e. an upper DE) are presented.

NOTE: The behaviour of the Operations/Primitives, the inclusion of more Operations/Primitives that may be required, is a subject for further elaboration/specification. This further elaboration may be done in the next release of the present document, and also some of the details may be completed during an actual implementation phase.

The clauses below illustrate the models. The DE Model is further enhanced later in the specification, in the clause on Knowledge Plane and also in the clause on the internal structure of a DE.

9.11.1 The Model of a Decision Element (DE), Models of Managed Entities (MEs) at GANA's lowest

The DE Model shows the Interface Names and what is happening on each interface, as well as the Primitives that should be supported by a particular DE Interface. A Management Interface of the DE is presented with its sub-interfaces and the Management Primitives that should be supported. The Primitives and their behavioural characteristics will be further elaborated in the next release of the specification (based on some initial work described in [i.43] and [i.5]).

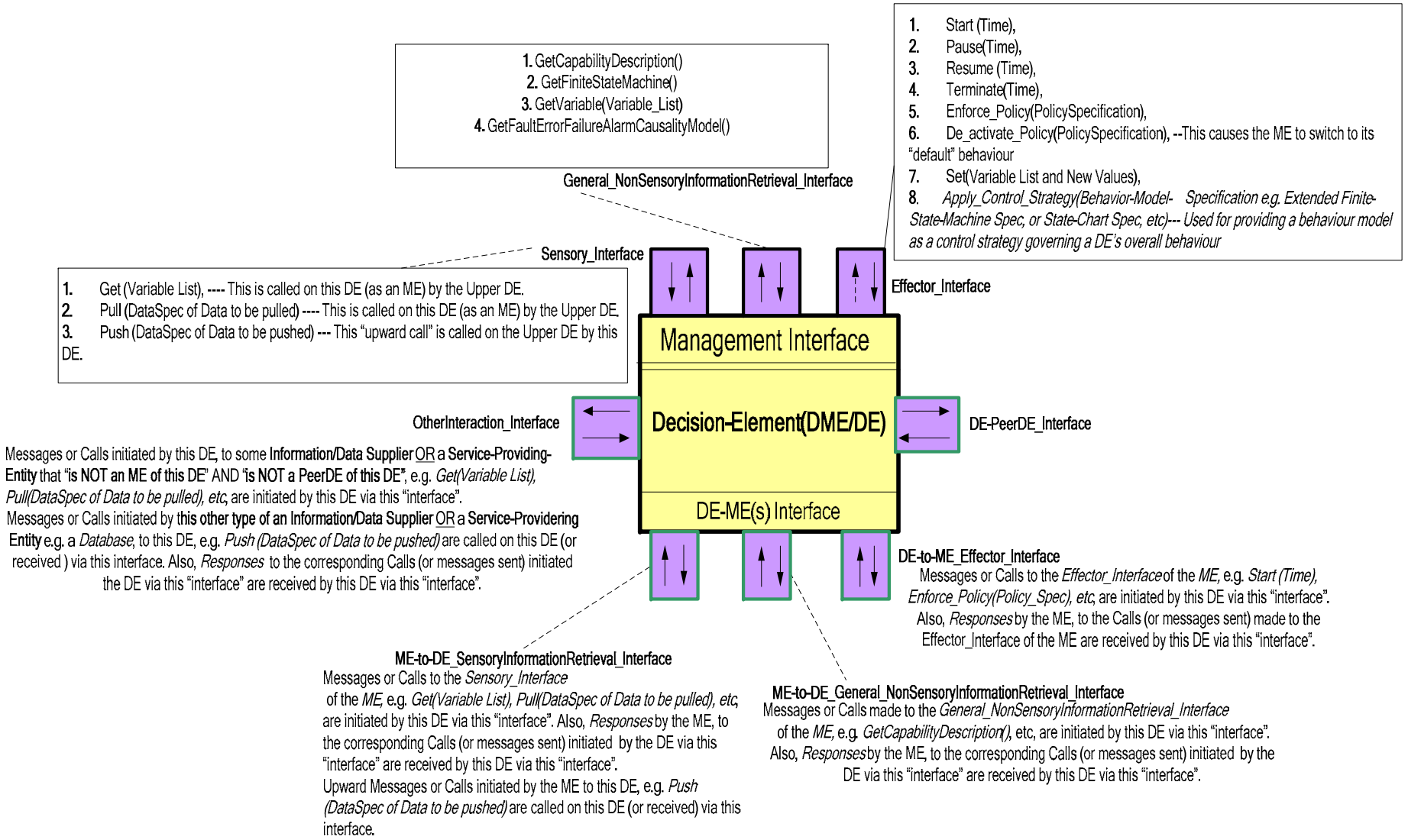


Figure 23: Interfaces of a Decision-Element (DE) and required Primitives

9.11.2 A Managed Entity (ME) at GANA's lowest layer (Variant -A)

Figure 24 shows an ideal model of a Managed Entity (ME) at GANA's lowest level MEs like protocols modules, components, etc, that has an "ServiceProviding_Interface" through which the services offered by the ME are requested for by some "user-entity" (contrast that to the interface with the upper-layer in the OSI Model); "ServiceRequesting_Interface" through which the ME requests for services provided by another entity (contrast that to the interface with lower-layer in the OSI Model). A Management Interface of the ME is presented with its sub-interfaces and the Management Primitives that ought to be supported (this is not really the case today, but only a partial fulfilment of the model can be traced in implementation and management of individual networking protocols). The Primitives and their behavioural characteristics will be further elaborated in the next release of the specification (based on some initial work described in [i.43] and [i.5]).

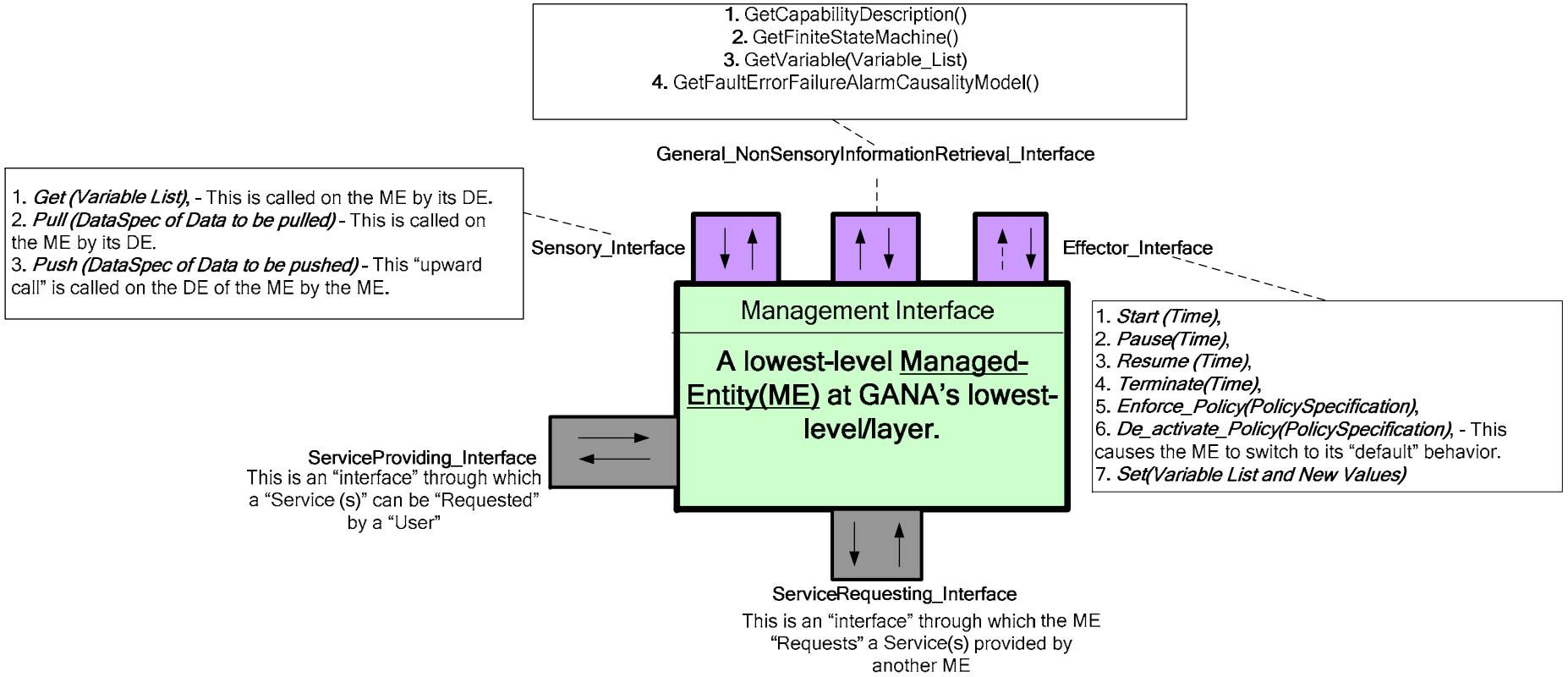


Figure 24: A Managed Entity (ME) at GANA's lowest layer (Variant -A)

9.11.3 A Managed Entity (ME) that is an "evolved Protocol" or a Future Protocol Model in GANA-at GANA's lowest layer (Variant - B)

There are ideas that recently emerged on how to design future protocols and the management operations/primitives that they shall support in order to support dynamic protocol-stack composition and re-composition as driven by some goals and context of operation and environment in which a device finds itself. Figure 25 illustrates such futuristic protocol models. Some of the ideas presented here for evolved or future protocols are taken from CONMan [i.8]. The EFIPSANS project elaborated and added additional primitives to the ones defined in CONMan [i.8]. The aspects will be further elaborated in the subsequent releases of the present document.

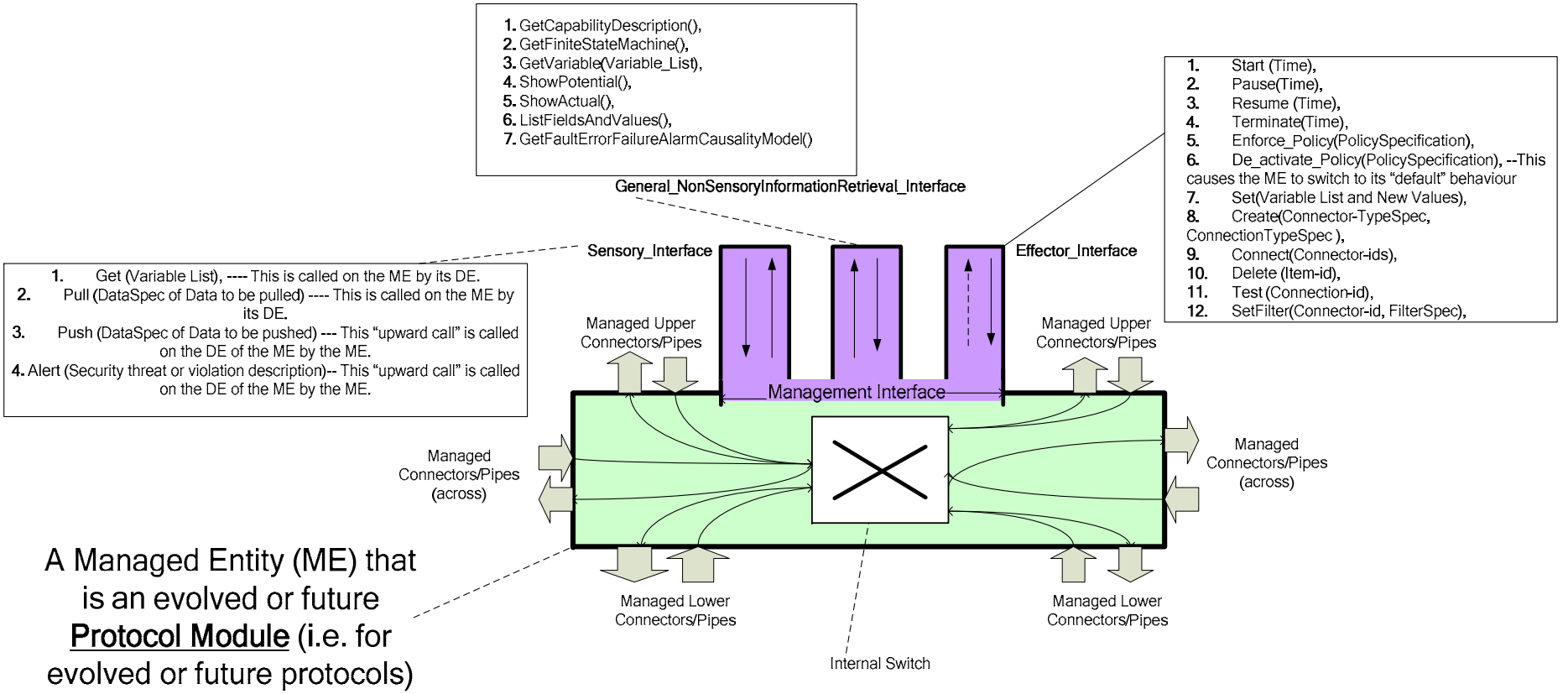


Figure 25: A Managed Entity (ME) that is an "evolved Protocol" or a Future Protocol Model in GANA - at GANA's lowest layer (Variant - B)

Additional models of what would be considered as lowest level MEs are possible. What is noted is that many types of components, tools integrated into systems today, do not support the notion of a "Management Interface" as depicted on the models above, rendering such entities "unmanageable". To be fully manageable an ME should support the notion of a "Management Interface". The figures in Appendix C and Appendix D of either of EFIPSANS deliverables D1.3 or D1.7 [i.43] depict the situation we have today regarding the "would be MEs" that do not fully support manageability (if at all).

9.11.4 Enabling Programmability within MEs and DEs

9.11.4.1 Programmability: what it is about

Programmability, as an enabler refers to the provision of "Primitives/Operations" on the "Management Interfaces" of various types of Managed Entities (e.g. protocols, stacks and networking mechanisms) to enable Decision Logic that governs autonomic behaviour to "program" (i.e. start, pause, resume, terminate) the operation of a particular Managed Entity while at the same time supplying as input, parameter values, policies or behavioural specification, etc, required as parameters of the "Primitives/Operations".

The following points characterize desired properties of Managed Entities (MEs) and Decision Elements (DEs) with respect to Primitives that should be supported on interfaces (particularly on their Management Interface), as well as other desired properties. Some of the properties are missing in today's management paradigms and architectural principles applied to designing protocols and other types of managed resources, modules/components. That means the desired properties can be applied in designing Future Network architectures and networking modules such as protocol modules. The behavioural aspects of the "Primitives" will be further elaborated in the next versions of the GANA Specification.

1) MEs that are protocols MAY support the protocol module abstractions (concepts, properties, capabilities) proposed by CONMan/Deep-4D [i.8].

   - Managed connectors/Pipes for the kind of protocols modules that can be made to support managed connectors proposed in CONMan.
   - Switches.
   - Filters.
   - Performance.
   - Security.

2) MEs that are protocols MAY support "Functions of the CONMan architecture" **BUT** leaving out the "ConveyMessage".

NOTE 1: The "**NM**" in CONMan maps to a *Network_Level_DE* in GANA. The "**MA**" maps to the "**Node_Main_DE/Node_DE**" in GANA.

3) A "Node_Main_DE/Node_DE" SHOULD also support the CONMan Functions **BUT** the functions shall have different behaviours when invoked on a "Node_Main_DE".

4) Similarly, GANA's "Functions_Level_DEs" SHOULD also support the CONMan Functions **BUT** the functions shall have different behaviours when invoked on a "Function_Level_DE" e.g. "Routing_Management_DE".

5) **Other Primitives** that SHOULD be supported on the "***Management_Interface***" of a ME and/or DE.

NOTE 2: The behaviour of the programmability-related operations/primitives, the inclusion of more primitives that may be required, is a subject for further elaboration:

- Start (Time).
- Pause(Time).
- Resume (Time).
- Terminate(Time).
- Enforce_Policy(PolicySpecification).
- De_activate_Policy(PolicySpecification): This causes the ME to switch to its "default" behaviour.
- Get (Variable List).
- Set(Variable List and New Values).
- Pull (Data to be pulled): This is called on the ME by a DE.
- Push (Data to be pushed): This is called on the DE by any Information Supplier to the DE, which could be the ME under its control.
- GetCapabilityDescription(): This returns the aggregate capability model description of the ME.
- GetFiniteStateMachine(): This is required by a DE in order to know the state transitions of the ME.
- GetFaultErrorFailureAlarmCausalityModel(): This is required for autonomic Fault-Management.
- GetFaiulreModesDescription(): This is required for autonomic Fault-Management if not covered by the implementation of GetFaultErrorFailureAlarmCausalityModel().

6) A DE may have dynamic *Roles*, designed as alternative selectable FSMs that can be executed by the DE depending on the *Role* a node is supposed to play (according to say, its support for context-awareness, or when network-objectives defined by a Human requires it to play some role) in the network, provided that the potential dynamic *Roles* of the node would mean that the DE in question should also be designed with the need for it to potentially support playing different *Roles*. A Role could mean access-router, border-router, relay, etc.

7) If a Protocol, Tool, or Component can be used directly by any functional entity (a DE or an ME of some sort) and by a number of functional entities simultaneously, without conflicts in the services it offers to the requesting functional entities, then it can be used directly by any functional entity. However, in such a case, a DE assigned to be the manager for the Protocol, Tool, Component, may need to maintain knowledge of the functional entities using it, possibly including statistics, etc.

8) If direct use of a Protocol, Tool, or Component by multiple entities simultaneously WOULD result in conflicts in the services it offers, then REQUESTS from functional entities intending to use it, SHALL go through the DE acting as manager for the Protocol, Tool, or Component, so that REQUESTS may be accepted or rejected, and conflicts resolved by the DE (manager) responsible. DEs need to discover each other.

9) DEs communicate with each other (within nodes or across nodes) in order to discover services available from their associated MEs by learning the MEs that have been initialized by DEs and are ready for providing services.

10) Filters may be applied on interfaces of DEs and on Managed Connectors of protocol level MEs.

11) DEs shall know the policies being enforced by other DEs on their corresponding MEs.

9.11.5 Assignment of Managed Entities (MEs) and their Configurable and Controllable Parameters to specific Decision Elements (DEs) in GANA ("*ME-Param*"-mapped to-"*1-DE*")

9.11.5.1 Concept of "ownership" in GANA

The "Concept of Ownership" is a feature of the intrinsic stability attributes that shall be considered for an autonomic network architecture (as is also defined in GANA). This concept requires that every ME is managed by a single DE, i.e. no two DEs (i.e. control loops) can control the same ME (i.e. functionality, resource, etc.) at any given point of time in the network. This is important from system's stability point of view since it relieves the burden of "conflicts resolution". Specifically, if an ME is controlled by two or more DEs at the same time, then, contrasting, conflicting and at times repetitive policies, objectives and reconfiguration requests, etc, originating from different DEs would lead to an unstable ME and thus, to an unstable autonomic network. Through the "Concept of Ownership", GANA ensures that this instability is avoided. An ME Parameter is assigned to and owned by exactly one DE, i.e. from an ME Parameter point of view (because a DE may own more than one ME parameter). In modelling terms, e.g. in UML, a "DE-to-ME_Parameter Relationship" would have multiplicity of "1" at the DE end point and multiplicity of "n" at the ME_Parameter end point. This clause illustrates how a DE relates in communication and interfaces to its own MEs, to other DEs, and to MEs owned by other DEs. At the end of this clause there is a Table that maps specific types of MEs and their configurable and controllable Parameters to specific type of DEs.

Figure 26, Figure 27, Figure 28 and Figure 29 illustrate various types of relationships that can be established and designed between DEs and MEs and the corresponding interfaces that can be used in DE-to-DE communications and in DE-to-ME communications.

If a Protocol, Tool, or Component can be used directly by any functional entity (a DE or an ME of some sort) and by a number of functional entities simultaneously without conflicts in the services it offers to the requesting functional entities, then it can be used directly by any functional entity. However, in such a case, a DE assigned to be the manager for the Protocol, Tool, Component, may need to maintain knowledge of the functional entities using it, possibly including statistics, etc.

In Figure 27, Figure 28 and Figure 29: if direct use of a Protocol, Tool, or Component by multiple entities simultaneously WOULD result in conflicts in the services it offers, then REQUESTS from functional entities intending to use it SHALL go through the DE acting the role of manager for the Protocol, Tool, or Component, so that REQUESTS may be accepted or rejected, and conflicts resolved by the DE (manager) responsible.

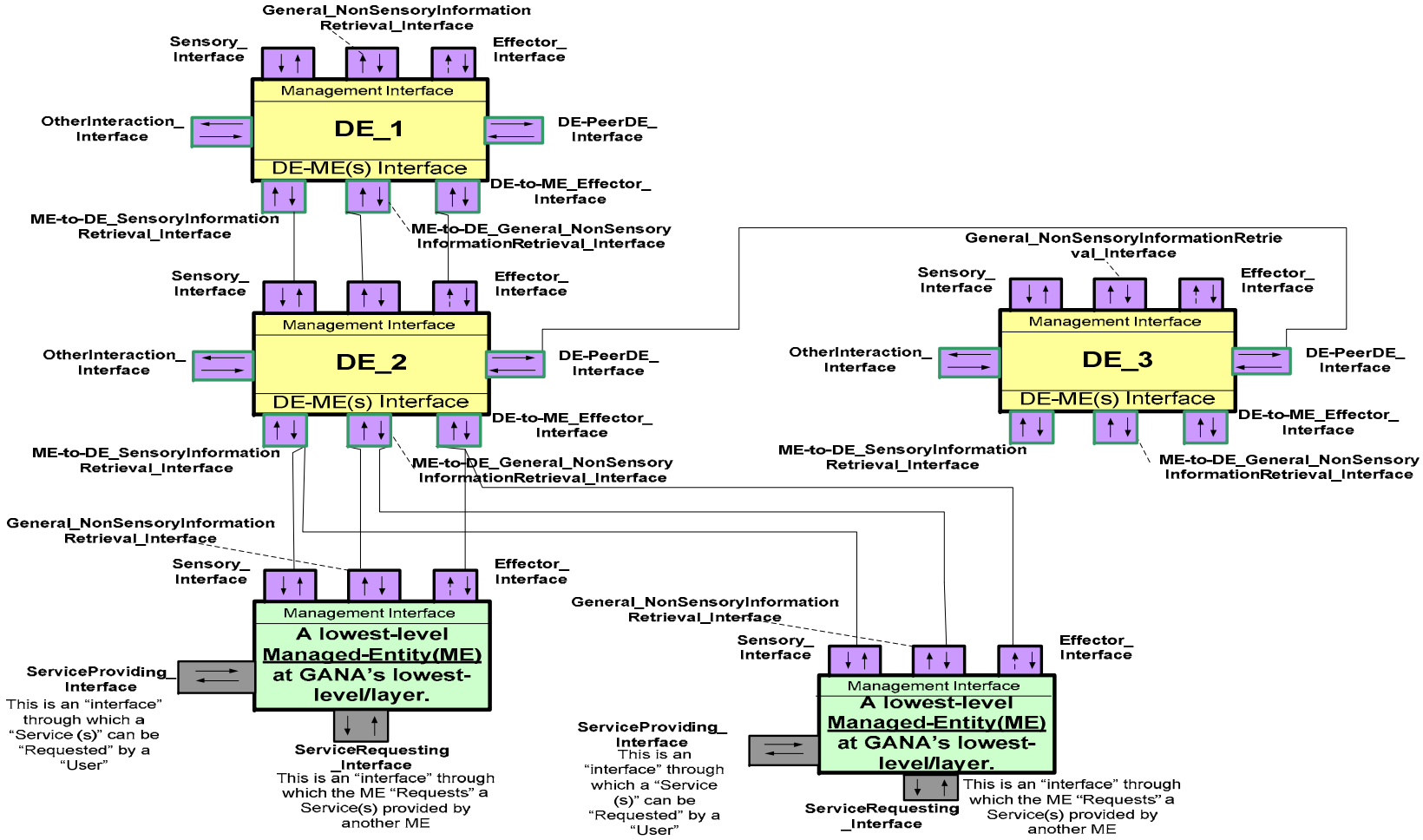


Figure 26: A DE with multiple MEs and interfacing with an Upper and a PeerDE (which could be a sibling DE since a Sibling relation is a special-type of a Peer Relation)

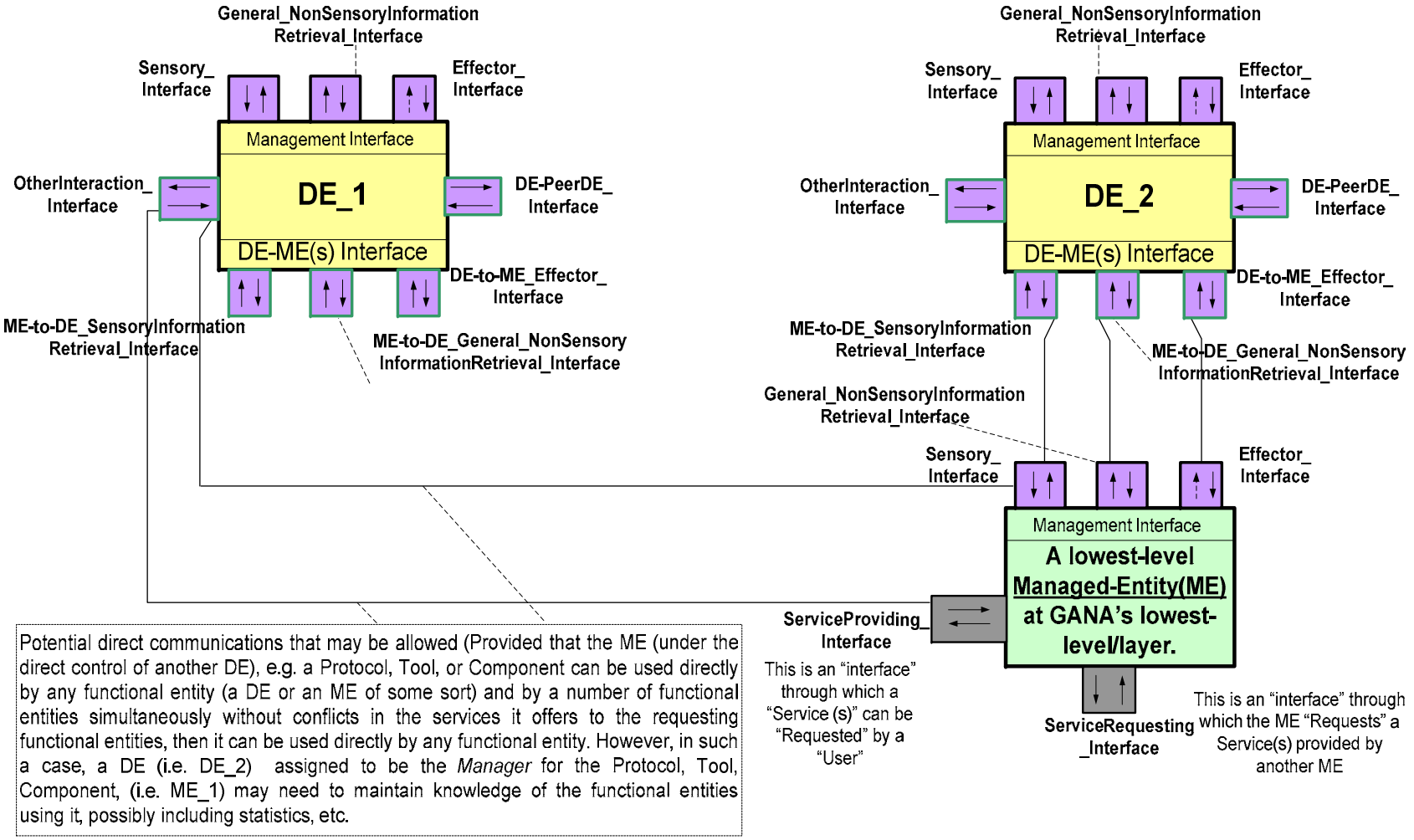


Figure 27: A situation under which a DE is allowed to interact directly with an ME that is not its own

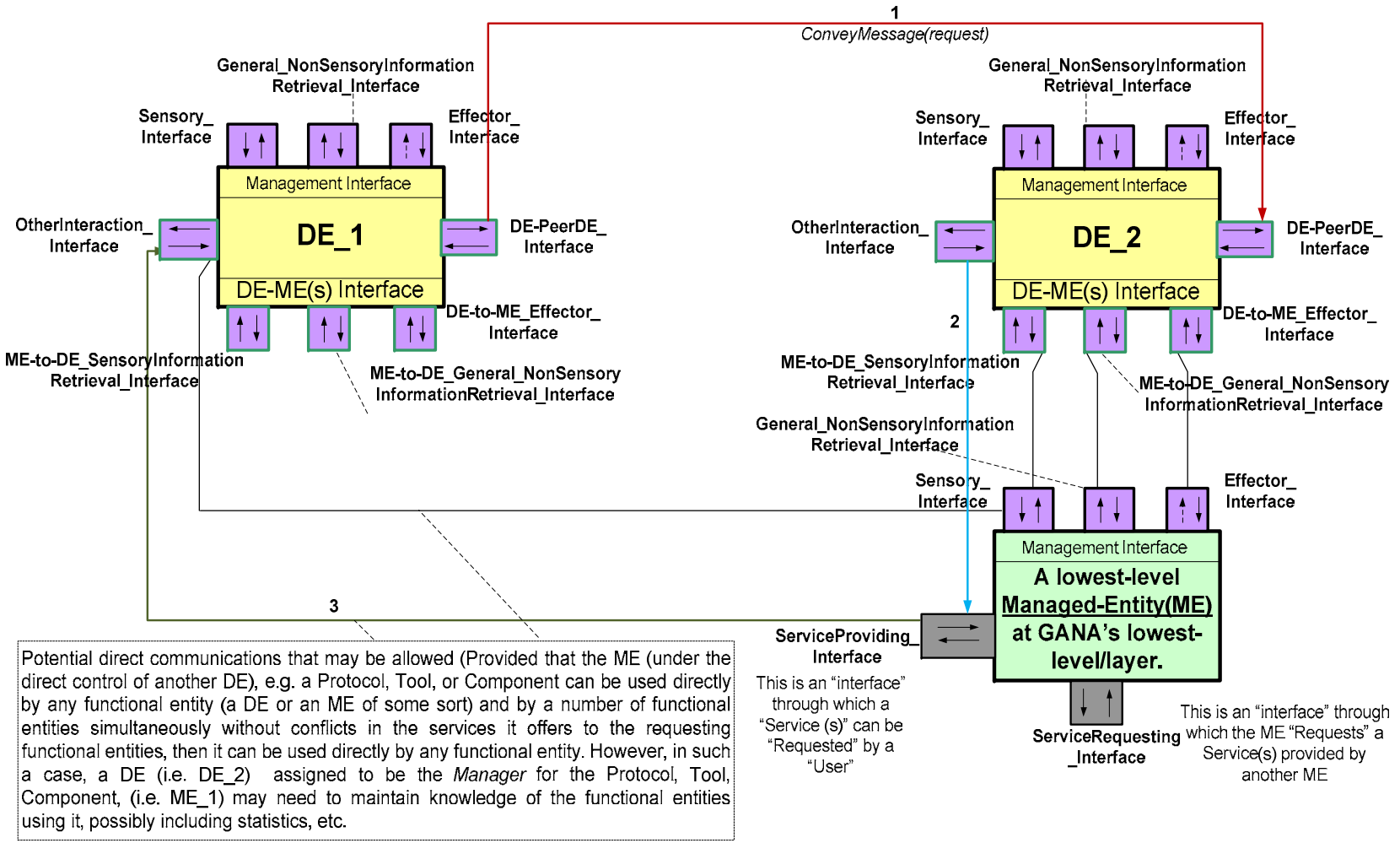


Figure 28: A case in which the DE may need to communicate with an ME that is not its own, via the DE of the ME

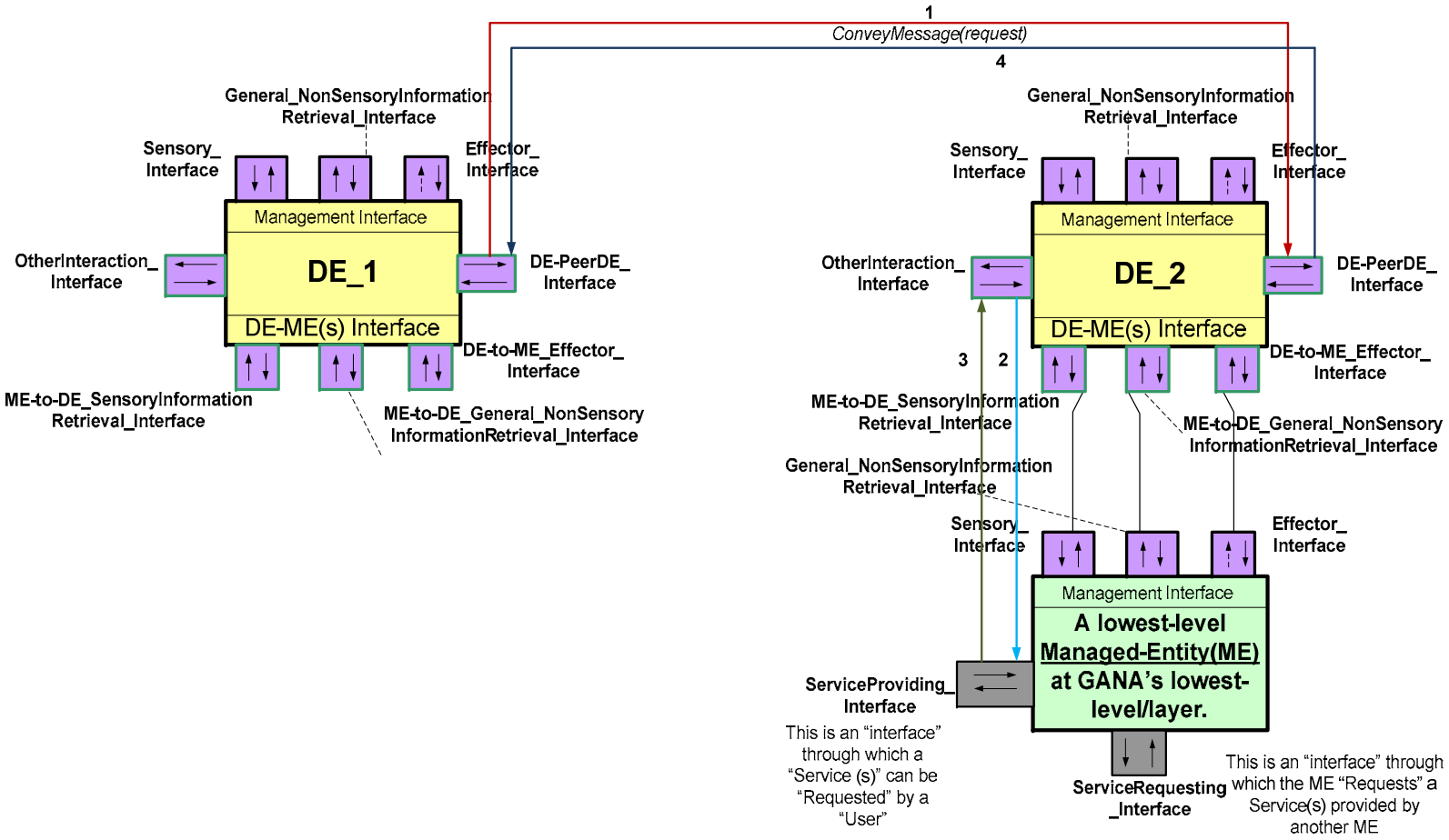


Figure 29: A case in which the DE may need to communicate with an ME that is not its own, via the DE of the ME

9.11.6 GANA Hierarchy - Mapping of Managed Entities (MEs) and their Configurable and Controllable Parameters to specific DEs (*ME-Param* -to- *1-DE* Mapping)

The following table presents a mapping of DEs from the Network-Level down to the GANA lowest layer/level Managed Entities (MEs).

NOTE 1: From Left to Right: It means viewing from the Network into a Network Element down to Protocols, Stacks and Mechanisms assigned to be managed by the assigned DEs as Managed Entities (MEs). Configurable and Controllable Parameters of MEs are assigned to specific DEs in a "ME-Param" -to- "1-DE" Mapping, i.e. from an ME Parameter point of view (because a DE may own more than one ME parameters). The **Table is divided into two Parts and spread on two pages** such that the second part is a continuation of reading from top levels going down to the individual Types of Managed Entities (MEs) mapped to the DE stack. The last column on the first part is repeated on the second part of the Table.

NOTE 2: The core functionalities (extensible) of some various DEs can be found in deliverables from the EC funded FP7 EFIPSANS project (downloadable from http://www.efipsans.org/), in which the specification and validation of the DEs were documented. Of course, the community can extend the functionalities. Example references include: [i.43], [i.44], [i.60], [i.61] and [i.62].

NOTE 3: The mappings of DEs to specific MEs and their configurable and controllable Parameters get refined and more specific when DEs are being "instantiated" onto a target implementation Reference Architecture (refer to AFI WI#3 work). And, some DEs on the network level that "mirror" specific DEs on Level-2 (Function-Level) may be missing in the Table, and so, the decision on whether it is necessary to have such DEs also on the network-level could be left to the AFI WI#3 work on the various Instantiations of the Reference Model onto various architecture types and environments to enable autonomicity, cognition and self-management.

Table 1

Network-Level DEs	Node-Level DEs	Function-Level DEs	Protocols and Mechanisms as Managed-Entities (MEs)	Examples of protocols and Mechanisms that are mapped as MEs
	GANA NODE			
NET_LEVEL_SEC_M_DE	NODE_LEVEL_SEC_M_DE		**Security Protocols, Algorithms and Mechanisms**	Certificates/Passwords Algorithms, Hash Algorithms, Encryption Algorithms, Access Control Mechanisms, Trust Mechanisms, Denial of Service (DoS) Detection/Prevention algorithms/mechanisms, Signature based intrusion detection mechanisms, etc.
NET_LEVEL_FM_DE	NODE_LEVEL_FM_DE		Fault Detection Mechanisms, Fault Isolation/Localization/Diagnosis Mechanisms, Fault Removal Mechanisms	Active Probing mechanisms, Bi-Directional Forwarding Detection (BFD protocol) for link failure detection, Self-test/diagnose functions, rebooting, reloading, automated module replacement mechanisms, etc.
NET_LEVEL_RS_DE	NODE_LEVEL_RS_DE		Proactive and Reactive Resilience Mechanisms, Survivability Strategies and Algorithms, Restoration and Protection Mechanisms	Node Resilience mechanisms, and Network Resilience mechanisms, etc.
	NODE_LEVEL_AC_DE		Neighbour Discovery Protocols/Mechanisms and Network Discovery Mechanisms	Neighbour Discovery Protocol (NDP), Secure Neighbour Discovery Protocol (SEND), etc.
NET_LEVEL_RM_DE		FUNC_LEVEL_RM_DE	Routing Protocols and Mechanisms	OSPF, BGP, RIP, ISIS, etc.
NET_LEVEL_FWD_M_DE		FUNC_LEVEL_FWD_M_DE	Layer-3 Forwarding Protocols and Mechanisms, Layer-2.5-Fowarding, Layer-2-Fowarding, Layer-3-Switching, Layer-2-Switching, etc.	IPv4/IPv6 Forwarding Engine, MultiProtocol Label Switching (MPLS), etc.
NET_LEVEL_QoS_M_DE		FUNC_LEVEL_QoS_M_DE	QoS Protocols and Mechanisms	Packet classifier, Packet Marker, Queue Management, Queue Scheduler, RSVP, etc.
NET_LEVEL_MOM_DE		FUNC_LEVEL_MOM_DE	Mobility Management Protocols and Mechanisms	Mobility Support in Internet Protocol Version 6 (MIPv6), Datagram Congestion Control Protocol, Mobile Stream Control Transmission Protocol, Site Multi-homing by IPv6 Intermediation, Proxy-Mobile-IP, Mobility-Management User-Equipment Managed-Entity, Measurement-Report-Function Managed-Entity, Candidate-Access-Router-Discovery mechanism, Fast Handover Scheme, Policy Control and Charging Rules Function mechanism, etc.
NET_LEVEL_MON_DE	NODE_MAIN_DE	FUNC_LEVEL_MON_DE	Monitoring Protocols, Mechanisms and Tools	IPFIX data collection and dissemination mechanisms, SNMP data collection and dissemination mechanisms, NETFLOW data collection and dissemination mechanisms, Protocol Analysers, Packet Trace creation and dissemination mechanisms. Effective and Available Bandwidth Estimation mechanisms, IPv6 hop-by-hop options for intrinsic monitoring, etc.
		FUNC_LEVEL_SM_DE	Services and Applications	Orchestration of services, service-discovery, interpretation of service and application requirements at run-time and requesting the network layer to behave in a service/application-aware manner, realizing a control-loop over the services/applications as its Managed Entities (MEs), collaboration with other DEs of responsible of autonomic management of the network layer protocols in order to realize collaborative self-adaptation on both the service-layer and the network-layer.

9.12 GANA as a Unifying Model

This clause provides how the GANA Model unifies, incorporates/accommodates concepts from best known approaches to autonomic networking, cognitive networking and self-management.

It also provides some perspectives on how concepts from different initiatives are incorporated or accommodated within the evolvable GANA Model as a single unifying model.

Table 2 describes how concepts from the different viable approaches to autonomic networking have been selectively combined in a harmonized way or accommodated within GANA Reference Model as a single unifying model.

Table 2: GANA Reference Model as a single Unifying Model that incorporates other models

Approach	How the concepts of the approach are accommodated in the GANA Reference Model
IBM -MAPE Model [i.10], [i.38]	The MAPE Model applies at each level in the GANA Decision Plane Hierarchy. However, the notion of "Planning" may need to be applied only at those Levels in GANA where Control-Loops need not be "fast control-loops"-i.e. likely to be the Node-Level and the Network-Level only. Control-Loops at the Function-Level and those that can be intrinsically introduced at the Protocol-Level need to be "fast control loops" since they directly affect the very lowest level Managed Entities (MEs) i.e. protocols, protocol stacks and mechanisms. More details on this subject can be found in the related sections of the GANA specification and description itself.
4D architecture [i.25]	The Decision Elements (DEs) in 4D are not necessarily the same as DEs in GANA, since some DEs in GANA are diffused into architectures of nodes/devices and the network as a whole, and also, DEs in GANA implement control loops at different levels of GANA's Hierarchical Control Loops (HCLs) framework. However, DEs in 4D correspond to Network-Level-DEs in GANA, meaning that Network-Level DEs in GANA can inherit and extend by design, the functional features of DEs of the 4D architecture. The Decision Plane in GANA, in contrast to the Decision Plane in 4D, has both a "Vertical View" through Decision Elements Hierarchical Relations, and a "Horizontal View" through DE-to-DE Peer and Sibling Relationships. The "Horizontal View of the GANA Decision Plane" is for realizing intelligence in the network devices and the network itself by introducing some autonomicity that takes a "horizontal view" to implement in a distributed fashion, device and network intrinsic self-management. More details on this subject can be found in the related sections of the GANA specification and description itself.
CONMan [i.8]	CONMan proposes protocol module abstractions (concepts, properties, capabilities) that enable manageability of future protocols in a "complexity-obvious way" and allow for dynamic protocol stack composition on the fly, and should be considered for the design of future protocols. The GANA adopts CONMan functions by adopting and extending the set of CONMan functions in the definition of the "*Management Interface*" of a Managed Entity (ME) model that is considered as a "Future Protocol model". CONMan advocates for *simplifying protocol design to enable manageability*: Protocol Engineering practices for future protocols that can be "pluggable" into networks will need to shift towards designing simpler protocols that are easy to manage. As

Approach	How the concepts of the approach are accommodated in the GANA Reference Model
	protocol engineering practices shift towards the approach advocated by CONMan, such types of Future Protocols supporting CONMan functions can be managed in an autonomic way by their associated Decision Elements (DEs). The "**NM**" in CONMan maps to and can be realized by a *Network_Level_DE* in GANA. The "MA" maps to and can be realized by a "**Node_Main_DE/Node_DE**" in GANA. More details on this subject can be found in the related sections of the GANA specification and description itself.
Knowledge Plane for the Internet [i.17]	Knowledge Plane is considered as a construct above the nodes/devices and even above the traditional Management Plane. **GANA Network-Level-DEs**, their cognitive engines, modules or services, form part of the Knowledge Plane. Also considered as a part of the Knowledge Plane is a Model-Based Translation Service (MBTS) that translates commands (/responses) from (/to) the Network-Level-DEs into (/from) a target command syntax and semantic formulation acceptable to the type of a target node/device being managed by the Network-Level-DEs. Network-Level-DEs such as the ones listed below, *form the "domains" of the Knowledge Plane, executing "K-application(s)" specific to a DE*: • *Net-Level-QoS-Management-DE;* • *Net-Level-Security-Management-DE;* • *Net-Level-Routing-Management-DE;* • *Net-Level-Fault-Management-DE;* • *Net-Level-DataPlaneAndForwarding-Management-DE;* • *Other type of Net-Level-DE,* However, some control-loops, cognition, and intelligence can be introduced into the node/device i.e. as provided for by the Node-Level (NODE-MAIN-DE) and Function-Level, assuming that we discourage Protocol-Level control-loops and cognition for reasons explained by 4D [i.25], CONMan [i.8] and other initiatives]. The reasons discussed include the need to simplify protocols by avoiding hardwiring complex decision-logic into protocols such as we now have in today's complex network protocols like OSPF that are now known to create undesired "emergent behaviours" when interacting with each other. More details on this subject can be found in the related sections of the GANA specification and description itself. Regarding cognition in GANA, DEs at the Function-Level, Node-Level, up to the Network-Level could be the ones that should require having *Cognitive Properties*. DEs at the bottom of GANA are meant to implement "fast control loops". Therefore, the notion of "planning", makes sense in a Node-Level-DE and a Network-Level-DE. However, having said that, at the Function-Level: Cognition *should be very simple and limited if it is to be allowed, because the Control-Loops at this level need to be "fast control loops"*. More details regarding Cognition in GANA can be found in clause 9.13.

Approach	How the concepts of the approach are accommodated in the GANA Reference Model
FOCALE [i.38], [i.28]	The FOCALE mainly focuses on a Network-Level types of a Control-Loop. In the FOCALE approach, a Managed Entity (ME) would have a *Finite State Machine* that models its State transitions specified and made known (by embedding) to the Autonomic Computing Element (which maps to a DE in GANA). The FOCALE also presents the concept of a Model Based Translation Layer that can be used to interpret and communicate vendor-neutral commands issued in a single unified form from an Autonomic Computing Element to a form that is understood by a targeted managed resource (a network element). The same applies to the translation of responses from the managed resources. Therefore, some ideas from FOCALE can be directly adopted when designing a DE, e.g. the internals of a DE can consists of elements defined in the FOCALE Model such as "*observe", "compare", "policy-server", "action", and "foundation"*. However, the Information Model used would need to be extended with GANA concepts, relations and constrains as discussed later in the present document. Therefore, the Autonomic Computing Element (in FOCALE), would be considered as a Network-Level DE in GANA because its associated type of a Managed Resource, is actually a node/device such as a router i.e. a Network Element (NE). Another additional important difference is that in GANA a Network level DE, apart from managing nodes and devices, can also manage lower DEs in nodes e.g. those at the GANA Level-2 (Function-Level). More details on this subject can be found in the related sections of the GANA specification and description itself.
GENI [i.46]	There are a number of concepts (if not all) in GENI that can be directly incorporated into the GANA Reference Model. For example, GENI Management Core should be considered as functionality of a Network-Level-DE in GANA while the CM functionality should be considered part of a Node-Main-DE in GANA. The aspects related to *Virtualization,* as well as the concept of *Slices* in GENI apply very well to the interface between "Function-Level-DE" and GANA's lowest layer/level Managed Entities (MEs) such as Protocols, Protocols Stacks and Mechanisms of a node/device-since these MEs could be virtual instances exposed by a *virtualization-layer* introduced between Function-Level-DEs and physical resources of the node/device. The DEs themselves, especially the Function-Level DEs could also be virtualized to allow multiple instances of the same type of a DE to run on the physical device. More details on this subject can be found in the related sections of the GANA specification and description itself.

9.13 Cognition and Knowledge Plane as part of GANA Decision Plane

9.13.1 Overview and Basic Definitions

This clause addresses Cognitive Networking and Knowledge Plane as part of the GANA Decision Plane as well as Information/Knowledge Sharing. It also aims to position cognitive networking in the context of GANA. This context has both a functional/situational aspect and a non-functional/infrastructural aspect. The functional aspect relates to the particular situation or problem to be addressed e.g. virtual networking, network governance, etc. This depends on the actual DE's that are instantiated in a particular situation as well as the individual cognitive approach taken. Such aspects are described elsewhere in the relevant parts of the GANA Specification. Infrastructural aspects relate to the support mechanism that GANA shall provide to enable cognitive networking. These aspects are described in this clause. The clause begins with a description of cognitive networking, followed by a description of the impacts and requirements on GANA. The clause relates to a number of enablers such as cognitive networking and information modelling.

Information can be defined as Processed Data or a Model. The Model could be a result of processing data and representing it by following and respecting a Data Model, or the Model can be one created by humans e.g. an Information Model, a System-Model, etc.

[i.36] Simplifies things by simply using one term "Information" to inclusively describe both Data and Information, hence the so-called "Information-layer". This seems to be a good approach.

Knowledge can be defined as: **Information** correlated according to a **Model** that is theoretically considered as a valid instance of a **Meta-Space** that defines the correlation among the abstract concepts of the **Information Elements**. Why "theoretically"?, because it may be difficult to achieve an ideal case/situation whereby all the elements required of correlated information are instantiated, due to the fact that some data or information from which to derive Knowledge, may be corrupted, incomplete or invalid. Therefore, an attempt to derive knowledge may result in **Partial or Incomplete knowledge**.

Therefore, **Knowledge** should be represented by a **Model** and its associated **Meta-Space** of which the Model is an instance of. The Model shall conform to the Meta-Space that describes valid instances of the Meta-Space. The Meta-Space can be a **Meta-Model** or could be some **Ontology and associated Schemas**, and shall be known by the cognitive process that claims to be able to operate on the knowledge and do something with it e.g. make some decision(s).

Cognitive processes may modify the Meta-Space and the elements of its instances (Models) during a learning process, by adding new "concepts" to the Meta-Space OR manipulating existing instances (Models) of the Meta-Space that are already known (i.e. are in the Knowledge Base) OR adding new instances (Models).

Cognition = Learning + Reasoning (Clark, Partridge, Ramming & Wroclawski [i.17] do define these terms).

9.13.2 What is Cognitive Networking?

The concept of a cognitive network encompasses the application of cognitive techniques allowing networks to perceive their conditions and then planning, learning, and acting according to end-to-end goals. The ideas of a cognitive network derive from work by Clarke [i.17], and Mitola [i.52]. The cognitive aspects are similar to that ones used to describe cognitive radio and broadly encompasses many simple models of cognition and learning.

It should be noted anyway that goals in a cognitive network are based on end-to-end network performance, whereas cognitive radio goals are localized only to the radio's user. End-to-end goals are derived across the network from operators, users, applications, and resource requirements. This difference enables the cognitive network to operate across all layers of the protocol stack. On the other hand cognitive networks targets are far beyond the scope of cross-layer designs. The cognitive network shall support trade-offs between multiple goals whilst cross-layer designs perform independent optimizations that do not account for the network-wide performance goals.

In summary, the ambition is to exploit autonomous, intelligent self-managing networks with consequent improvements in network efficiency and performance, reduced OPEX and CAPEX and increased profitability. A cognitive network can be seen as an extension of autonomic management whereby the network and management systems are embodied with the capacity to "think, learn and remember" [i.53].

Cognitive networking is a new and evolving field without a defined and agreed terminology, architecture, etc. The application of cognitive techniques in the networking domain will be therefore gradual. There will be incremental application of aspects of knowledge representation and reasoning occurring in selected parts of the network operations and management over time. Aspects that will influence this evolution include.

9.13.2.1 Cognitive process

A cognitive process can be broadly defined as a machine learning process, which is based on algorithm that "improves its performance through experience gained over a period of time without complete information about the environment in which it operates" [i.40]. Underneath this definition, many different kinds of artificial intelligence, decision making, and adaptive algorithms can be placed, giving cognitive networks a wide scope of possible mechanisms to use for learning.
There is a wide spectrum of approaches to represent knowledge in intelligent systems - ranging from simple ECA type rules, through various forms of logic to representing uncertainty through probability based mechanisms - with an associated range of reasoning mechanisms over these knowledge forms. Equally there are a variety of approaches to machine learning that are applicable to the process. Taken together these mechanisms define a very large space from which potential solutions to operational and management problems in the network domain can be applied.

9.13.2.2 Cognitive Architecture

There will be a combination of centralized and decentralized approaches to the application of cognitive networking. This reflects both the abilities and constraints of the cognition approaches as well as the inherent structure of the network domain itself.

A commonly shared model of cognition is the three-level theory [i.31]. Model is summarized as consisting of behavioural, functional, and physical layers. The behavioural level determines what observable actions the system produces, the functional layer determines how the system processes the information provided to it, and the physical layer comprises the neuro-physiology of the system.

Many researchers have adopted a three layer framework to describe the application of cognition to self-managing systems [i.56], [i.54], [i.57] and [i.55].

Figure 30 reports as an example three classes (strategic, tactical and classical) of cognition approaches characterized by different features. These approaches have different time scales and make use of different type of data (symbolic and numeric).

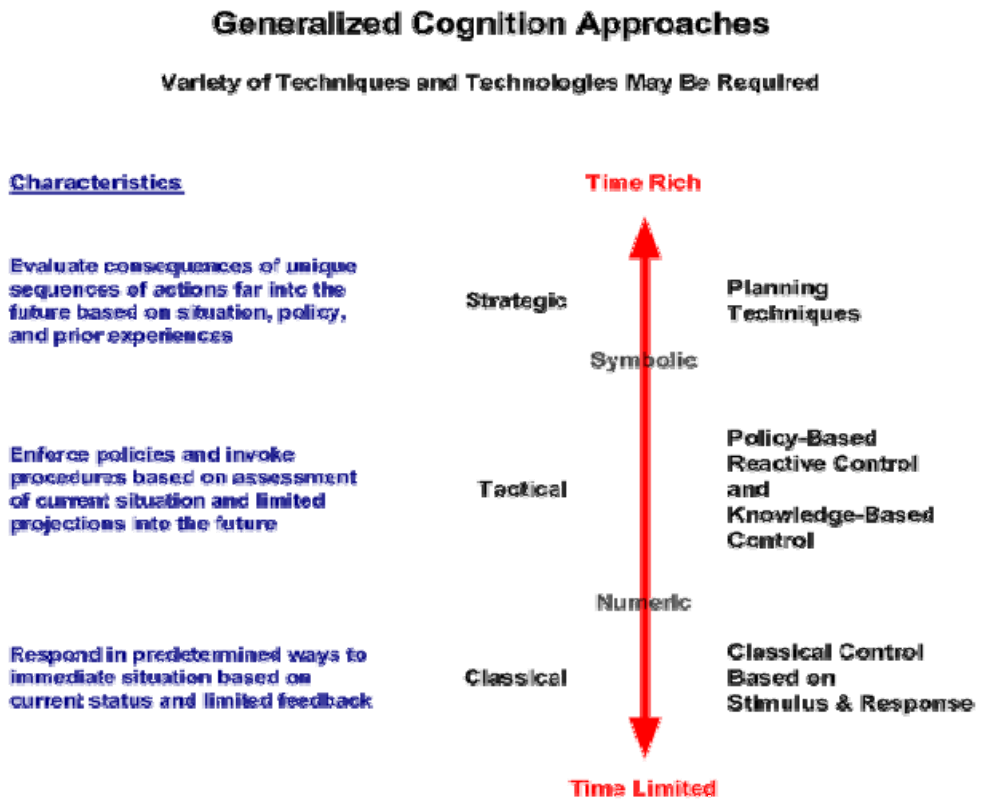


Figure 30: Three classes of cognition approaches characterized by different features

As mention it appears efficient to consider a proper trade-off between centralized and decentralized mechanisms. In general decentralized approaches are more likely to apply to time-limited aspects of network operation i.e. to real-time resource management and traffic handling; centralized approaches applying to more time consuming activities higher up the management stack. Decentralized cognitive approaches relate in part to topics such as self-organization, emergence, network intrinsic management and autonomic communications (e.g. through the use of multi-agent systems (MAS) and distributed autonomic components).

Another dimension that will influence the deployment of cognitive networking is deciding to what aspects of the network operation and management such solutions should be applied. A general criterion is that cognitive techniques are best used where the network is most dynamic and where adaptation to changing conditions is most needed. (Thus we see a great focus on cognitive radio to manage dynamic spectrum selection). However what is regarded as "non-dynamic" today may change tomorrow. For example it has been suggested [i.57] that fixed access networks are reasonably static in operation and therefore not an obvious candidate for cognitive networking. However as discussed in AFI, this is likely to change in the medium term in NGN and that cognitive techniques will be needed to manage this part of the network.

9.13.2.3 Relationship to existing Networking Planes

Traditionally networking activities have been split into three planes i.e. data/user plane (forwarding, transmission, etc.), control plane (routing, signalling, etc.) and management (monitoring, configuration etc). Although generally accepted there tends to be ambiguity about the precise meaning of the various planes especially the control and management planes. This arises from an overlap in some areas between the control and management planes e.g. configuration and also to the differing points of view from various sections of the (diverse) networking community.

It is not possible to cleanly map cognitive networking to a single existing plane, partly due to the inherent ambiguity of definition of these planes and partly due the different interpretations in the literature as to what precisely constitutes a knowledge plane. For example some authors separate knowledge representation from control (inference and learning) and to regard the modelling and representation of knowledge as the knowledge plane while others include both control and representation in a knowledge plane that exists alongside and complements the existing network planes. Further the situation is compounded by the various interpretations of what is meant by data, information and knowledge and where the boundaries lie between these terms.

From a GANA view point it is suggested to avoid this dilemma. Cognitive networking represents a family of technologies and approaches that can substantially augment the control, operation and management of networks over the current state of the art. These approaches are applied as required in particular situations to different parts of the existing planes to yield improved solutions. In effect the cognitive network merges into and is subsumed by the existing network planes. Because the deployment of cognitive networking will be gradual we will see both evolutionary and clean slate approaches applied concurrently. In some cases cognitive techniques will be used to augment existing, legacy, solutions as for example in the use of ontology's to give more semantics to current data and information models or to incrementally introduce improved management operation by improved semantics in policy management. In other cases at the same time there will be a more radical introduction in cognitive techniques, as for example in the cognitive radio space.

9.13.3 Impacts on the GANA Model

The impacts of cognitive networking on GANA are a combination of functional/situational and non-functional /infrastructural. Functional refers to the particular network operational or management situation to be addressed e.g. network governance, virtualization, OSS etc. This aspect of the application of cognitive networking is described elsewhere in the relevant parts of the present document. Infrastructural impacts are discussed here. Infrastructural impacts can be considered under the following headings.

9.13.3.1 Relationship to GANA Model (functional aspect)

The functional aspect of cognitive networking is realized by the particular set of DE's that are instantiated to solve particular aspects of network operation and management. Cognitive networking in this view is thus subsumed into the GANA Decision Plane. The non-functional aspects are realized through a combination of the Decision Plane and the other GANA planes.

At a conceptual level the generalized cognitive model described in Figure 31 maps very naturally to the GANA Decision Plane Control-Loop Hierarchy. The categorization of approaches serves as template to allocate cognitive approaches to the various level of GANA DE, as in Figure 31.

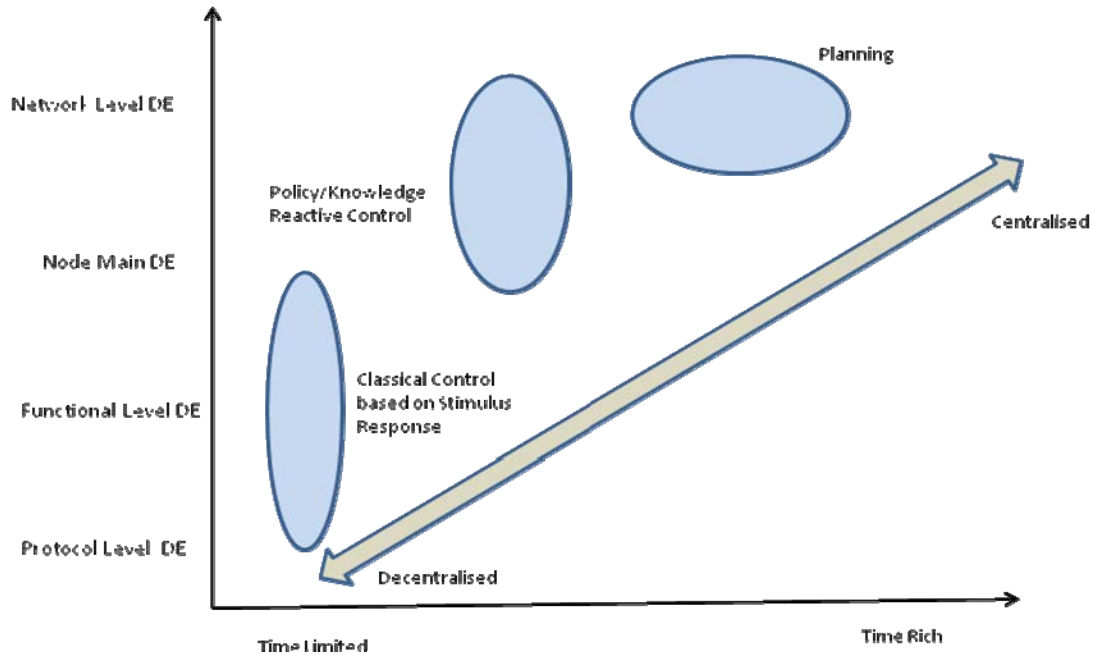


Figure 31: Mapping of GANA Control Loops to Cognition Approaches

9.13.3.2 Support for Decentralization

Support for decentralized approaches based on e.g. multi-agent systems requires communication and dissemination infrastructure (middleware and protocols). This relates to Network Intrinsic management (though that subject encompasses approaches which are non-cognitive). How to separate/combine the descriptions is ffs (for further study).

9.13.3.3 Implementation in DE's

A simplified model of a cognitive based DE can be obtained from a consideration of the generic description of a GANA DE and Russels and Norveg's (R&N) representation of an intelligent agent as depicted on Figure 32. There are two main elements represented in Figure 32.

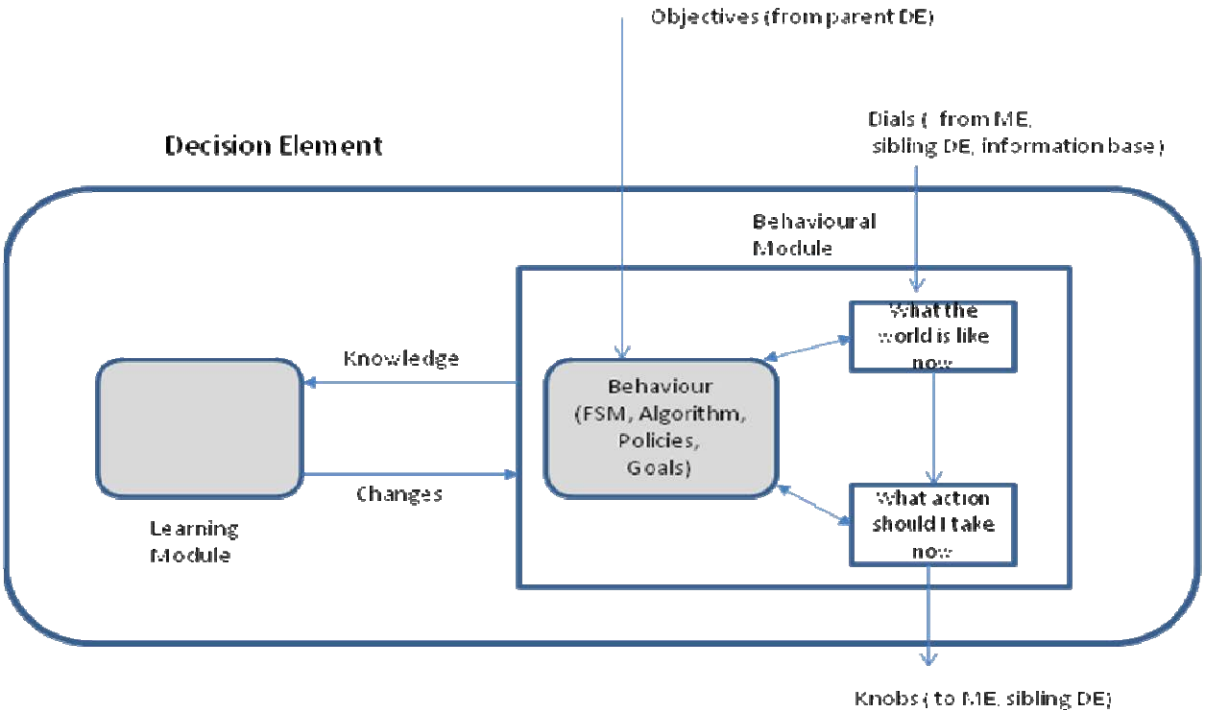


Figure 32: Model of a Cognitive DE

- Behavioural Module - this is the intrinsic functionality of the DE and can be considered as an intelligent agent in R&N terminology. The complexity of the module may vary according to the scope of the DE (e.g. function level vs. network level) and type formalism used to represent behaviour. According to the GANA design principles the behaviour may be specified dynamically i.e. changed at run-time.

- Learning Module - This module takes information from the behaviour module performance and determines how the behaviour module should be modified to do better in the future. The design of the learning module depends very much on the design of the behaviour module. Depending on the situation the learning may be continuous (online) or the training may be done beforehand (offline). There may be impacts on the learning module for a DE whose behaviour can be modified at run-time. The learning module may be optional (depends on what whether we consider learning as an *essential* part of cognition).

9.13.4 The need for an Information/Knowledge Sharing Overlay Network

In an autonomic network, interconnected entities need to collaborate in order to fulfil the global goals. For efficient collaboration to be established, data (e.g. capabilities description, configuration data, events, alarms, measurements), information (e.g. filtered data, consolidated measurements, etc) and knowledge (e.g. new information produced by using reasoning on existing information, transformation and symbolic representation of information, etc.) shall be gathered, elaborated and shared between the different entities in the network.

Any data (in numeric, symbolic, or semantic format) can be gathered and transformed into information; by "information" we mean the result of pre-elaboration and correlation of raw data. A further step is policy-based filtering and transformation (for example, by reasoning and learning techniques) of this information into knowledge, which is necessary to exploit the autonomic and/cognitive control loops.

A powerful system for information/knowledge exchange shall be established in an autonomic network. The information/knowledge exchange system enables more advanced autonomic/cognitive functions like self-configuration, auto-discovery, self-adaptation, self-optimization or other self-* functionalities.

The information exchange system shall have the usual requirements for a commercial system, i.e. it shall provide Stability, Scalability, Robustness, Security, Extendibility (data storage and functionality). Further, it shall be: easy to install, easy to maintain, easy to manage, easy to use. Since different types of information cannot be stored in one place and can be distributed in different information servers/repositories/sources throughout the network and even across network domains, ideally such a system can be realized as a distributed system that forms an Overlay Network for Information eXchange (ONIX).

Information categories

Before describing the requirements of this information/knowledge exchange system, the information to be exchanged shall be defined. AFI characterizes the information using the following criteria:

1) The type of raw data and information to be gathered and exchanged.

2) The type of knowledge, for example numeric knowledge and semantic knowledge.

3) The size of the information/knowledge:

    a) Small sized information/knowledge is considered the key - value type of information/knowledge. Medium sized information is any type of information with size below 1 MB. Any type of information/knowledge exceeding 1 MB is considerate as large sized information/knowledge.

4) The frequency of which the information/knowledge changes over time:

    a) Slow changing information is any type of information that changes at intervals longer than 1 hour. Medium changing information is any type of information changing at intervals bigger than 30 seconds and smaller than 1 hour. Fast changing information is any type of information changing at intervals less than 30 seconds.

5) The maximum delay between the data collection and information consumption by the designated recipients, i.e. is the time critical or not.

6) The accuracy of the information/knowledge in order to be effectively used in the decision making process.

7) The scope of the information/knowledge that define the context where information are valid.

Based on the above criteria, a general information is defined as any type of information that is small or medium sized, slow or medium changing and not time critical. Examples of such types of information are: capabilities description, policies, network profiles, some monitoring data, etc. For general type of information a general information exchange system shall be in place. For special type of information like large sized information or fast changing information or time critical information other specialized systems for information exchange shall be in place.

In the following, a distributed system for general type information exchange is described. The information exchange system is a required component for the GANA.

The key requirements and goals that shall be considered when designing such a system are:

- Scalability: The system should operate efficiently in large networks in terms of network nodes, supported services, autonomic entities within a node, number of interconnection links, transported data or control traffic, number of events, storage resources, etc. In addition, the system should address the needs of different administrative domains interconnected together (i.e. *Interaction between Information/Knowledge Sharing Systems belonging to different administrative domains*). It could be claimed that the system shall be fully distributed, for example, by relying on distributed hash tables (DHT).

- Reliability: The system should be robust against possible failures, e.g. in the network, at the node level, at software, etc. The system shall replicate stored data in order to overcome multiple concurrent failures. In addition, any functionality assigned to a (primary) node providing services to the network nodes should also be replicated to back-up (secondary) node.

- Bootstrapping and Topology Formulation Capabilities: The system shall be able to initiate itself without any pre-configured data. System functionality may be further extended or the system performance may be improved when basic administration parameters are set. In addition, the system should be able to remain operational in dynamic environments, e.g. after link or node failures, node mobility, etc.

- Efficient Data Handling: The system shall be able to manage and update any information stored within it. Data replication shall be easily achieved for any data types stored. Finally, data should have a predefined scope in order to assess the validity of data in different network sections.

- Expressive Queries: The system should support complex (i.e. multi-attribute) queries. Partial queries should be supported (queries that contain only a subset of the attributes originally advertised by resources, considering the other attributes as wildcards). Each query might also have a scope (global, local, n-hops away, etc.).

- Efficient Notification: The system should provide a powerful subscribe mechanism for receiving information of interest. The subscription should support multi-attribute filtering capabilities.

- Standardised Interfaces: The system should provide well-designed interfaces to allow efficient data input and retrieval by/from the autonomic nodes. The control messages of the system should not significantly contribute to network congestion. In addition, It shall provide well-designed communication protocols and primitives to be used by servers that are part of the overlay network, to build and maintain the overlay.

- Security: The system should support capabilities for avoiding or minimising data poisoning. Authentication and authorisation functionality may be needed in various scenarios.

- Extensibility: The system should be extendable to allow the inclusion of future functionality.

- Minimised Operational Overhead: Since the goal of GANA is to reduce the complexity in the network, Information Exchange System shall be optimised for simplicity of installation and maintenance.

AFI proposes the working title "System for Information Exchange in Autonomic Networks" (SIXAN) for this system. Such a SIXAN system can be realized by a system of Information Sharing Repositories that can be considered as forming a distributed system. Ideally such a distributed system can be realized as a dedicated Overlay Network for Information eXchange (ONIX). Any network system (even a router) designed to share information as if it were an Information Sharing system supporting ONIX protocols can join the overlay network and use the protocols of the overlay network to share information and to answer information queries in collaboration with members of the overlay network (ONIX). Therefore, an ONIX system is a distributed system of Information Servers, that supports publish/subscribe, query and find type of services for Information/Knowledge such as Capabilities of network elements, Profiles, Goals and Policies of the autonomic network, pointers to resources and Data, and other types of Information/Knowledge.

NOTE: ONIX" is not the same as "onix" system in [i.50].

ONIX has no decision-making logic (in contrast). An information query only need to be sent to any of the servers (even if the info is actually stored on another server) and the servers will communicate with each other via an internal protocol such as DHT/Chord to find the server having the information, and respond to the query. As a system that is more powerful than a traditional databases, the servers can be XML-based storages.

Services provided

The following services shall be provided:

- Storing and Retrieving Information:
  - Push & Pull models supported.
  - Different classes of information.
  - Add, remove or replace operation supported.
  - Information is described using a mark-up language (e.g. XML).
- Query for Information: by supporting a query language capable of expressing complex queries (partial queries, scoped queries, etc.) (e.g. XPath).
- Disseminating the information: Upon request, periodically or event triggered:
  - Normal Subscription.
  - On-behalf Subscription.
  - Publish & Disseminate.

- Security: Authentication, Authorization, Trust, Confidentiality, Integrity, Non-repudiation, Privacy, Tracking of activities taken and originators of each input to the system for accountability and auditing.

- Reliability, fault-tolerance and accessibility.

9.13.4.1 The Subscription Mechanism

Any authorised entity can query the SIXAN system for data or can subscribe to retrieve information of interest. A subscription is defined by the requestor, the information requested and the trigger to gather and send the information. For defining what information to receive, a mark-up language, as mentioned above, is used. Regarding the triggers for SIXAN to send the information, we distinguish between two types of subscriptions:

1) Timed triggered subscriptions. A network element can subscribe to receive the information at a specified date and time (which could be immediately) or periodically at certain intervals. In the last case it can choose to receive the information once, for n times, or as long it is subscribed.

2) Event triggered subscriptions. A network element can subscribe to receive the information whenever a certain event occurs. Such an event could be, for example, the fact that a node with a specific set of capabilities joins the network, the fact that some resource's description is updated or a parameter's value exceeds a predefined threshold. Also for this type of subscription the network element can choose to receive the information once, n times, or whenever data gets submitted/updated as long as the network element is subscribed.

9.13.4.2 Bootstrapping

SIXAN/ONIX will also be used to store bootstrapping details for the autonomic network. Bootstrapping is a self-sustaining process that progresses without (or with limited) external help. In an autonomic networking environment, bootstrapping requires the provision of initial configuration information to newly joined nodes or newly created networks. This is designed to allow the successful initialization of the network mechanisms without any pre-configured data. Network-wide parameter estimation, data forwarding mechanisms, routing protocols, and security policies are some of the processes that need to be activated by the autonomic nodes without human support or pre-configured software. Decentralized bootstrapping, i.e. process instantiation without the need for special-purpose nodes, is also crucial in autonomic networks since it facilitates the design of functionalities and mechanisms in ad-hoc and resource constrained environments.

Bootstrapping may be realized in multiple phases. Initially, nodes in proximity (or neighbouring nodes) may establish point-to-point communication channels with each other. In a later stage, neighbouring nodes collaborate in order to establish end-to-end communication paths, e.g. by enabling proactive/reactive routing functionality. Service provisioning and fulfilment of any network-wide objectives are realized in later stages, e.g. by provisioning distributed repositories for data, information and knowledge sharing.

9.13.4.3 Inter-Domain Information/Knowledge Sharing through ONIX-to-ONIX Interface

The ONIX system, as a distributed system may be scoped, owned and policied (i.e. governed) by a particular administrative domain. This implies that different ONIX's may exist and so an **ONIX-to-ONIX interface for interdomain Information/Knowledge sharing** may be required.

NOTE: In the next release of the Specification, a detailed inter-domain ONIX-to-ONIX interface will be described.

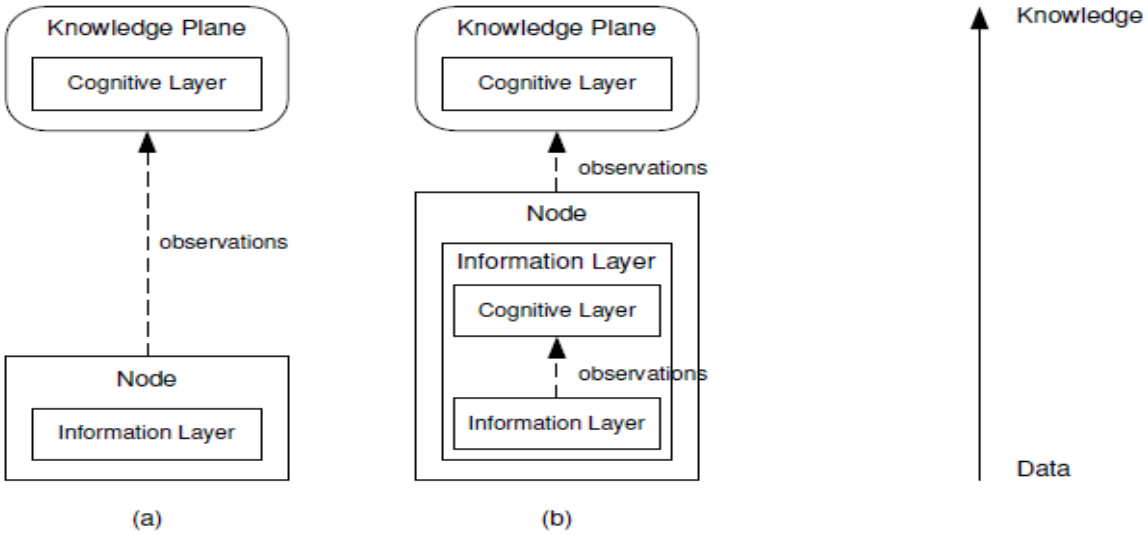
9.13.5 Knowledge Plane as part of the Decision Plane of the GANA Model

9.13.5.1 Knowledge Plane definitions

Citing "Clark, Partridge, Ramming & Wroclawski, from whom the original idea of the Knowledge Plane came from" [i.17]: The Knowledge Plane, is a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network. It is a distributed and decentralized construct within the Internet to gather, aggregate and act upon information about network behaviour and operation (citing "Stephen Quirolgico et al") [i.36].

The Knowledge Plane shall also be able to operate on Incomplete Knowledge or even be prepared to handle the situation of untrusted information sources contributing untrusted information to the knowledge derivation process used by the Knowledge Plane [i.17].

The following texts and figures are extracts from [i.36] on the subject of "Deriving Knowledge for the Knowledge Plane". The successive figures develop, in a step-by-step fashion, the core concepts and processes required in deriving knowledge required by the GANA Knowledge Plane.



NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure 33: General architectures (a) base architecture and (b) architecture with recursively-defined information layer

Citing [i.36]: The **Information layer** represents the set of entities of a node that provides configuration information and produces observations for use by the knowledge plane. In the "base architecture" (see on the figure), the information layer defines the set of sensors (i.e. anything that is a source of information), configuration variables and local information models for a participating node. The **cognitive layer** embodies the knowledge plane's reasoning mechanisms and cognitive services, as targeted to specific knowledge-based applications (e.g. K-apps). What is referred to as "observations" in the figure are part of what are called "Views" on interfaces of DEs and Managed Entities (MEs) in the GANA Model.

[i.36] says, a participating node may comprise both an **information layer** and a **cognitive layer** to support local reasoning. Information from a node may be processed by the node's own cognitive layer, which in turn propagates this knowledge (as observations) to the knowledge plane. This recursive architecture might facilitate scalability by reducing the volume of data and knowledge to transmit and store, and also by reducing associated processing requirements. This principle is a fundamental principle actually embraced in defining the basic 4-Levels of abstractions of the GANA Model i.e. some control-loops, cognition, and intelligence go into the node/device i.e. as provided for by the Function-Level and Node-Level (NODE-MAIN-DE), assuming that we discourage Protocol-Level control-loops and cognition for reasons explained by 4D [i.25], CONMan [i.8] and other initiatives.

The "base architecture" may be easily extended to realize recursively-defined information and cognitive layers [i.36]. This aspect is reflected by the Protocol-Level, Function-Level and Node-Level abstractions in GANA.

As described earlier, a cognition layer maps to a Cognition Level in a particular GANA Level i.e. there is some cognition-level associated with a GANA level e.g. Function-Level (realized by the cognitive capabilities of the specific DEs at this level). Also noted in autonomics is that autonomicity is defined by the presence of a control-loop (not presence of cognition!) and the control loop's autonomic manager component e.g. a DE in GANA, may not need to be cognitive, though it shall implement some control-loop.

From the definition of the Knowledge Plane provided in [i.17] and [i.36] considered as a construct above the nodes/devices and even above the traditional management plane, we can conclude the following: the Knowledge Plane in GANA is realized by the inter-working of the building blocks (functional blocks) described in the next clause ("The Knowledge Plane according to GANA").

9.13.5.2 The Knowledge Plane according to GANA

As depicted on Figure 28, it is realized by the inter-working of the following building blocks:

1) GANA Network-Level-DEs, their cognitive engines, modules or services.

2) Information/Knowledge acquisition and sharing mechanisms (which include publish/subscribe, query/search and find mechanisms that shall be supported by information/knowledge storage Repositories). This includes services provided by a distributed Information eXchange System required in an autonomic network, such as the ONIX system of peer-to-peer based overlay of Repositories that support publish/subscribe, query and find type of services for information/knowledge such as Capabilities of network elements, Profiles, Goals and Policies of the autonomic network, pointers to resources and Data, and other types of information/knowledge.

3) Translation Services that "map" Ontologies or Information Models used to describe node-local information communicated to the Knowledge Plane, to Knowledge Representations e.g. Ontologies that satisfy the requirements of the cognitive services of the Network-Level-DEs. The Translation Services, knowing the source and target representations, translate node-local information received to the appropriate representation for use by a specific cognitive process or the so-called "K-application(s)" i.e. the domains of the Knowledge Plane. Network-Level-DEs: Net-Level-QoS-Management-DE; Net-Level-Security-Management-DE; Net-Level-Routing-Management-DE; Net-Level-Fault-Management-DE; Net-Level-DataPlaneAndForwarding-Management-DE; etc, form the "domains" of the Knowledge Plane, executing "K-application(s)" specific to a DE.

4) A Model-Based Translation Service (MBTS) that translates commands (/responses) from (/to) the Network-Level-DEs into (/from) a target command syntax and semantic formulation acceptable to the type of a target node/device.

In the subsequent sections that look at techniques for Knowledge synthesis, representation and presentation to the Knowledge Plane, there are aspects that could also be implemented as part of the MBTS that are mentioned in those sections.

NOTE: The MBTS is further described in the present document, in Figure 64.

9.13.5.3 GANA Decision Plane

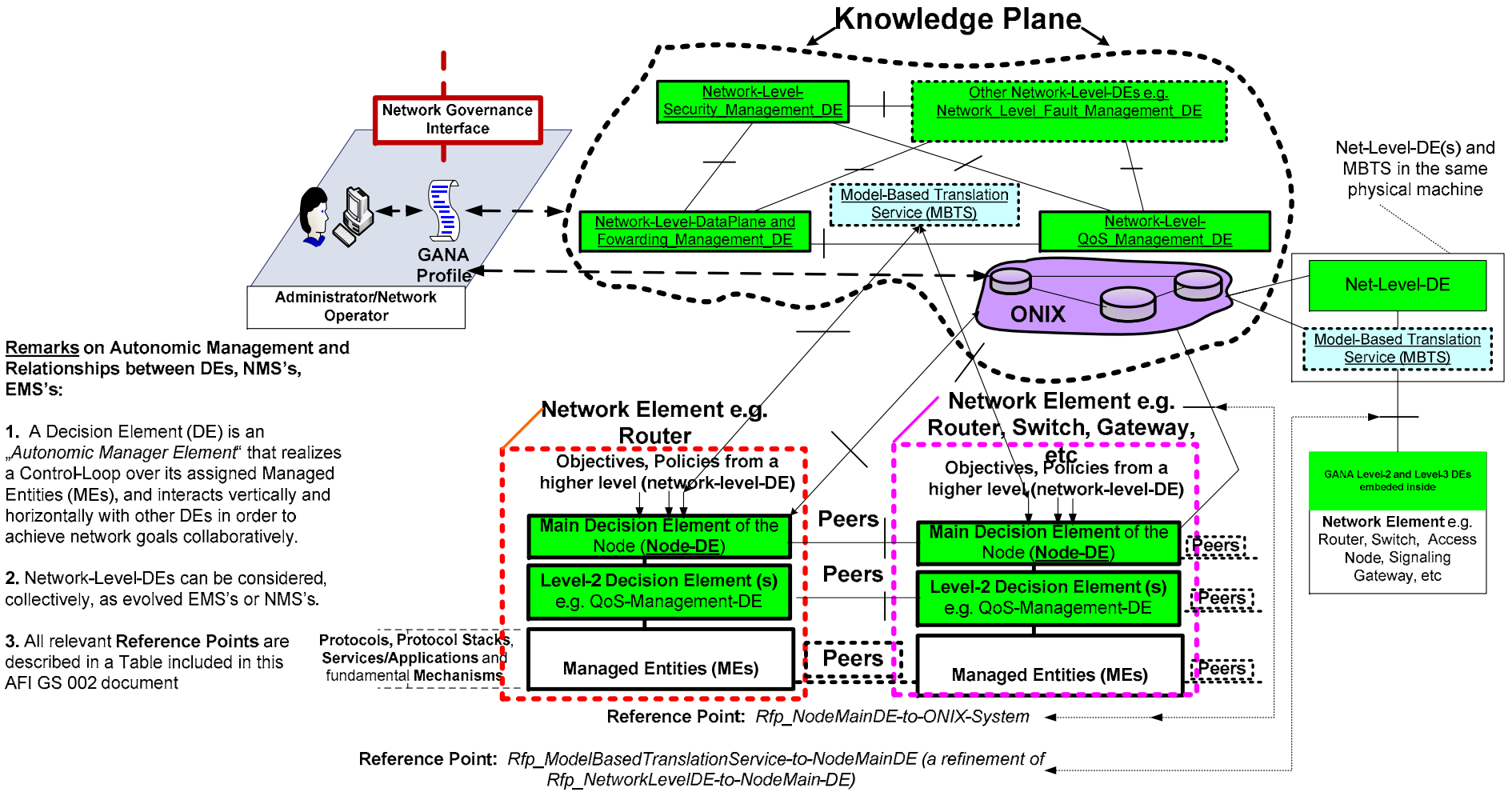
From this understanding, it implies that the GANA Decision Plane (which has a vertical and horizontal view of DE relations) does not wholly belong to the Knowledge Plane, i.e. only the Network-Level part of the GANA Decision Plane is part of the Knowledge Plane.

The properties of the Knowledge Plane, as described by Clark, Partridge, Ramming & Wroclawski [i.17], can be used in designing the Network-Level-DEs, together with properties defined and experimented with in the design of Decision Elements (DEs) in the 4D initiative [i.25], which are Network-Level-DEs in GANA terms. Other properties and concepts from some initiatives may inspire the design of DEs in general.

The GANA Decision Plane encapsulates today's Management Plane and replaces it in the long term, and adds the Horizontal view of the Decision Plane to allow distributed DE-to-DE interactions for network-intrinsic management (for those aspects requiring network-intrinsic management).

Cognition in the Control Plane can be realized through the cognitive properties of appropriate Function-Level-DEs (Level-2 in GANA) and/or protocol-level DEs (i.e. protocols with intrinsic control-loops), and the DE-to-DE interactions for network-intrinsic (i.e. in-network management) management (for those aspects requiring network-intrinsic management).

Figure 34 presents the Functional Blocks that constitute the Knowledge Plane. It also reflects on the Governance Interface, through which the Human Operator interacts with the Autonomic/Cognitive Network. The subject of Network Governance is also covered in the present document The figure also illustrates the Autonomic Functions (DEs) that may be introduced into network elements (leaving out autonomic control-loops that may be introduced in some individual protocols (i.e. protocol-intrinsic control-loops). In evolving today's architectures to "Autonomicity-Enabled Architectures", the Level-2 DEs and associated control-loops, up to the Network-Level DEs, should be the focus, than introducing control-loops within protocols). When instantiating the Knowledge Plane for Ad-hoc/Mesh Networks, possibly the Knowledge Plane may be considered as diffused within the network elements themselves than being a separate/standalone entity as required of Mobile and Wired Networks. It may also be possible to have an isolated/standalone Knowledge Plane for Ad-hoc/Mesh networks (AFI WI#3 addresses this).



Remarks on Autonomic Management and Relationships between DEs, NMS's, EMS's:

1. A Decision Element (DE) is an „*Autonomic Manager Element*" that realizes a Control-Loop over its assigned Managed Entities (MEs), and interacts vertically and horizontally with other DEs in order to achieve network goals collaboratively.

2. Network-Level-DEs can be considered, collectively, as evolved EMS's or NMS's.

3. All relevant **Reference Points** are described in a Table included in this AFI GS 002 document

Figure 34: Functional Blocks of the Knowledge Plane, and some of the related Reference Points

9.13.6 Possible Approaches to Deriving Knowledge for the Knowledge Plane

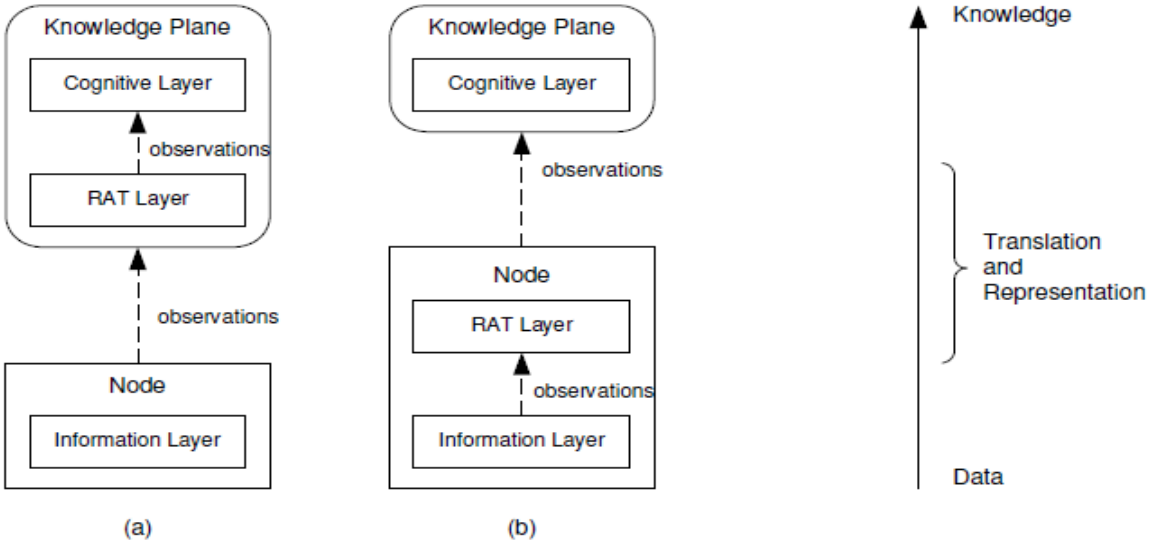
In [i.36], different approaches to deriving Knowledge for the Knowledge Plane are discussed. As already mentioned, the various aspects presented in this clause with respect to Knowledge synthesis, representation and presentation, e.g. the RAT component described later, can be considered in the design of the MBTS (Model Bases Translation Service). However, having said that, a different approaches to realizing the Knowledge synthesis, representation and presentation to the Knowledge Plane's DEs and MBTS, can also be pursued.

NOTE 1: The MBTS is further described in the present document, in Figure 64. Figure 35 presents the different possibilities that can be possible with respect to knowledge synthesis and usage.

As discussed in [i.36], a means is required to transform information e.g. node-local information into a form suitable for use by the target cognitive layer. Transformation might be required because the cognitive layer (1) works at a higher level of abstraction, (2) considers aggregations over sensor observations, or (3) requires a particular form of knowledge representation to facilitate its reasoning mechanism. This is referred to as the Registration, Acquisition and Translation (RAT) layer in [i.36]. The RAT layer may exist within the knowledge plane or within each of the knowledge plane's participating nodes. For more details of the RAT, refer to [i.36].

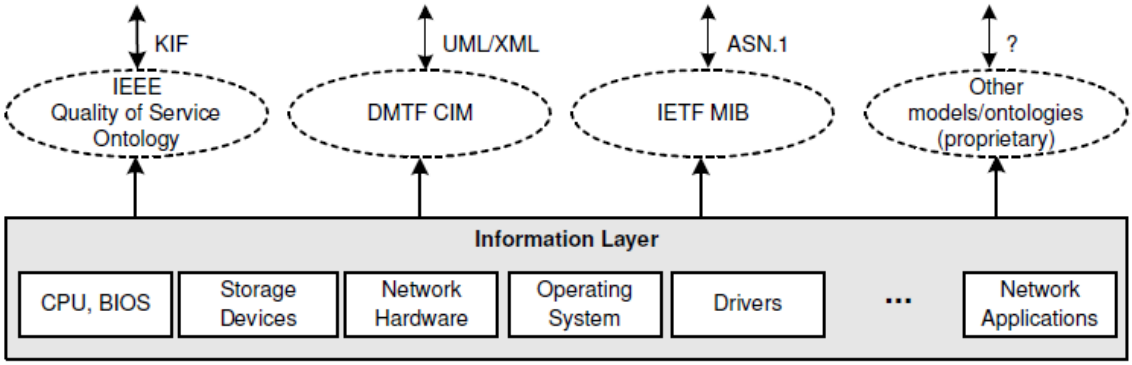
NOTE 2: In the GANA Reference Model presented in the present document, the RAT can be implemented as part of the MBTS (Model Bases Translation Service).

NOTE 3: The MBTS is further described in the present document, in Figure 64. Figure 36 presents the Information layer and RAT layer is presented in a figure in the Appendix. More details about all these figures can be found in [i.36].



NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure 35: Base architecture with RAT layer (a) as part of knowledge plane (b) as part of participating node

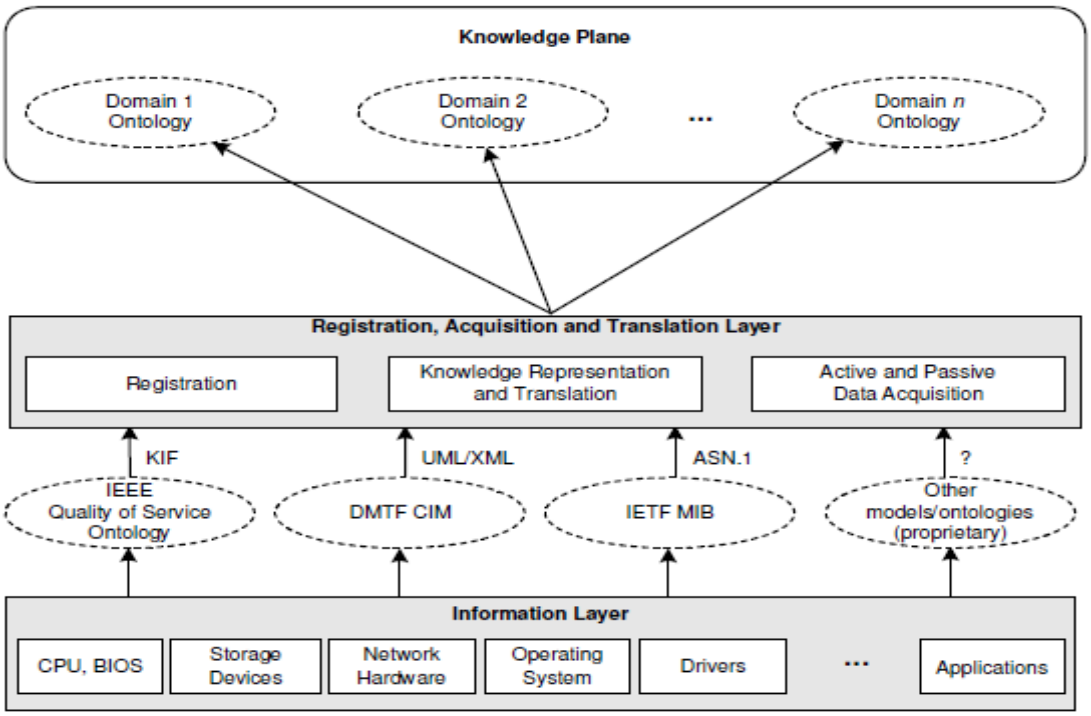


NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure 36: Information layer

9.13.6.1 Knowledge Representation and Translation Component

[i.36] says, depending on the requirements of the knowledge plane, there are at least two approaches to defining such knowledge: the *domain ontologies approach* and the *composite ontology approach*. More details on the approaches can be found in [i.36]. The figures extracted from [i.36], belong to some of the figures that could be considered in the design of Functional Blocks of the Reference Model, like the MBTS. Figure 37 presents *Derivation of domain ontologies*. Other figures that can also be considered in Knowledge representation and translation are included in the Appendix part of the present document. They include: a figure presenting the RAT layer (Figure D.1); and Figure D.2, which presents *Translations of domain ontologies*; Figure D.3, which presents *Multiple translators for heterogeneous nodes*; Figure D.4, which presents *Use of composite ontology for deriving domain ontologies*; Figure D.5, which presents *Composite ontology translation*. For more details about these figures refer to [i.36].



NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure 37: Derivation of domain ontologies

9.13.7 Other perspectives on Knowledge Building and use by Knowledge Plane

Figure 38 describes a set of steps in building the network knowledge. These steps encompass: collecting data from pieces of equipment and links of the networks during the functioning; selecting, filtering and pre-processing these data in order to get a formatted set of data that can be used to infer patterns and extracting the network knowledge (latter actions that can be done by a knowledge plane, by using data mining and knowledge extraction techniques and algorithms). This knowledge then can be used to close the loop for management and control actions.

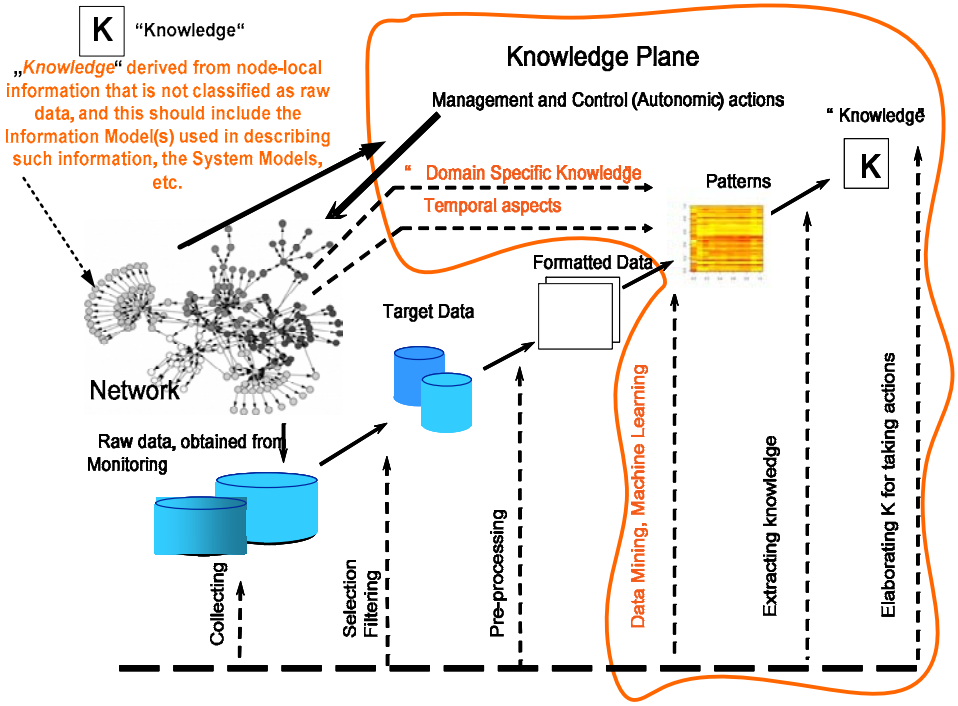


Figure 38: Other perspectives on Knowledge building

9.14 Possible approach to designing the internal modules of a Decision-Element (DE)

The internal modules that can be considered in designing a Decision Element (DE) are depicted on Figure 39. Apart from considering the picture presented in the figure on "*Model of a Cognitive DE*", the approach that can be taken to designing a DE can be based on an approach defined in FOCALE [i.38], which has been elaborated here to indicate where GANA concepts, relations and constrains would have impact and should be taken into consideration. More details can be found in [i.38].

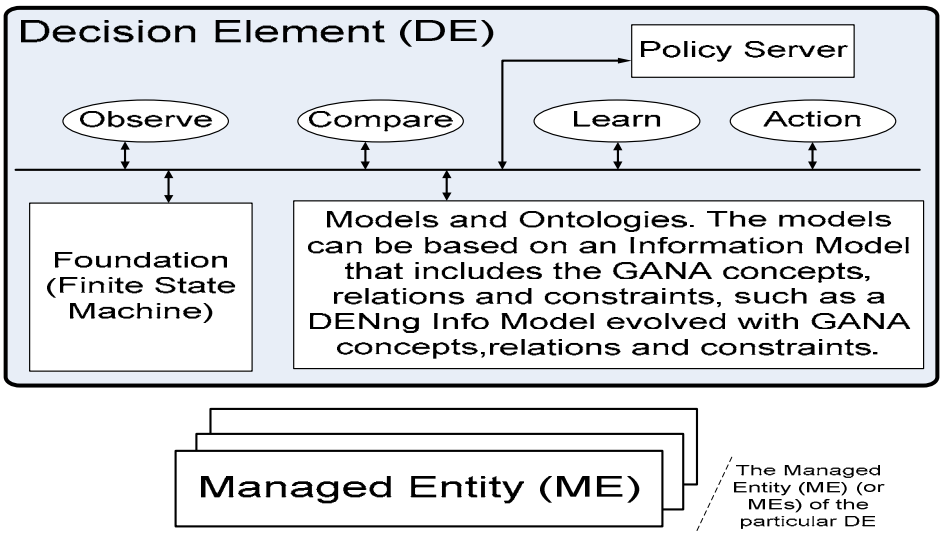


Figure 39: A possible approach to designing the internal modules of a Decision Element (DE)

9.15 Virtualization in GANA

Virtualization of resources (from networking to processing to storage, etc) will be a key characteristic of future networks. Examples are:

1) Virtual Private Networks, interconnecting several network nodes, network end-points and network devices by means of links characterized by some negotiated QoS parameters.

2) Distributed execution environments consisting of virtual resources provided by some clouds.

3) Distributed applications organized according to peer-to-peer architectures and overlay networks. In general, virtualization relies on creating, allocating and managing virtual resources, i.e. independent portions of the capabilities provided by physical resources.

As a consequence, autonomic features shall be able to support the supervision/management of resource virtualization and the creation, maintenance, and orchestration of aggregations of virtual resources (e.g. all the resources allocated to a single user/distributed application).

Communication (e.g. links, Virtual Private Networks, etc.) resources and IT-based resources (e.g. processing, storage, specialized functions, such as identity management, contents delivery, etc.) have different peculiarities, and, therefore, different, but related solutions could be envisaged.

9.15.1 Autonomic capabilities in IT-based resources virtualization

In general virtual resources are provided by "physical" resources, according to a virtualization model. The virtualization model enables to create, starting from a shared ensemble of physical resources, a uniform substrate of virtual resources, dynamically allocable to applications. The model is generalized as it can be applied a different types of resources, such as computing, storage, "things" (e.g. sensors, actuators, machines, smart devices), service components, etc. A virtualization model includes:

- An abstract view of the features/functions provided by the resources, in order to:

  1) simply the access to the capabilities; and

  2) hide the heterogeneity on interfaces/protocols provided by physical resources to control/access their features.

- A mechanism to group the capabilities of the resource (e.g. in terms of processing, storage, bandwidth) in isolated partitions; the partitions are to be used as allocation units of virtual resources; configuration mechanisms allow to associated, in a flexible way, the capabilities of the physical resources (e.g. storage dimension, execution cycles, queue lengths, optional functions) to the partition to be allocated, and to configure the QoS parameters (e.g. to fit the negotiated SLA); each partition is an isolated context both to protect the physical resource and the other partitions to an incorrect/malicious behaviour of the application and create an environment for the enforcement of negotiated performance, by avoiding (functional/non-functional) interferences among the partitions.

- A formalism for the requests and negotiation of virtual resources, e.g. in terms of dimension/quantity of resources to be allocated and description of non-functional features; an associated protocol is adopted to request/negotiate virtual resources.

Through a virtualization model, a physical resource can be "virtualized", so as to create and manage virtual resources to be assigned to applications/users.

A "virtual-machine" like model, similar to the one proposed by [i.9], can be adopted for implementing a virtualization model for IT-based resources virtualizing. The following functions are identified:

- the capabilities offered by the physical resource;

- the virtualisation mechanisms for partitioning the resource capabilities in "isolated" slices, and to provide interfaces to access them (examples of slices are the Virtual Machines);

- the hypervisor for monitoring the state of the resource capabilities and the slices; it implements supervision functions to detect critical situations, and trigger a decision process to select and activate corrective actions; hypervisor shall be able to enforce the policies to fulfil negotiated SLAs, to detect and recover failures, to optimize performances and the use of resources;

- virtual resource allocator in charge to allocate, de-allocate and, possibly, migrate virtual resources; it has to interwork with hypervisor to get information on available resources, with the virtualization mechanisms to request the creation of slices, their configuration (in terms of allocated resources, SLA parameters, etc.), and their termination;

- a set of slices, each of which represents the partition of virtual resources allocated to an application; each slice is an "isolated" context that perform application activities.

The capabilities that can be virtualized through virtualisation mechanisms and partitioned in isolated slices depend on the adopted virtualization model and on the type of the virtualized resource. For instance, in a resource providing computing features (e.g. a server) the virtualization mechanism should consider ports, network bandwidth, CPU cycles, RAM, Disk partitions; a resource which provide some application function (e.g. accounting, or identity management), the virtualization mechanism should consider the creation of independent service instances, and their configuration according to service parameters.

9.15.1.1 Autonomicity and Virtualization within GANA

These functions can be enriched with autonomic features [i.32]. In the proposed GANA Reference Model the autonomic features shall be introduced in the following way:

- Resource virtualization: the Node-Main-DE of a node/device shall be responsible of collecting and aggregating Capabilities of the hosted resources (Managed Entities), which shall include features supported, "cost" (see note 1) of providing a requested service, capabilities related to "partitions" that a Managed Entity (ME) as a resource exhibits (examples include the ones listed above, such as storage, support for creation of multiple instances, etc.).

NOTE 1: When referring to "cost" associated with a service, a partition or slice, the "cost" does not mean monetary metric, but may mean "weight" metric.

- Hypervisor: the Node-Main-DE, acting also as a Hypervisor provides the following autonomic capabilities:

  1) self-CHOP features to supervise the behaviour of the slices;

  2) self-adaptation of policies to guarantee the enforcement of the negotiated parameters associated to the slices;

3) algorithms to monitor/forecast the load, to provide useful information to allocation function.

- Allocator: the Node-Main-DE, acting a Allocator, realizes autonomic features in the algorithms for the negotiation, allocation and, possibly, migration of virtual resources; moreover, by means of DE-2-DE interaction it could cooperate in executing decentralized solutions for virtual resource allocation.

- Slices: slice's DE, which could be the Node-Main-DE or a Function-Level-DE e.g. the Function-Level-Routing-Management-DE, provides autonomic self-CHOP (i.e. Self-*) features to monitor/tune the behaviour of a slice, e.g. in order to fulfil negotiated parameters.

The virtualization model introduces dynamic aspects (e.g. in the creation, allocation, de-allocation, etc. of slices) that shall be mirrored in the DE model. It is necessary to enable the dynamic creation and termination of DE instances according to a given DE model: each type of slice (which depends on the adopted virtualization model and on the type of virtualized resource) should have a corresponding DE model, which defines the autonomic capabilities of the slice. DEs associated to slices are dynamically created when the slice is instantiated, and destroyed when the slice is deleted or migrated. The Hypervisor function of the Node-Main-DE can use the programmability-related operations/primitives on the "*management interface*" of a DE Model (e.g. *Start(Time)*; *Pause(Time*); *Resume (Time), Terminate (Time), etc.*) to create and manage at run-time an instance of a particular type of DEs, in charge to supervise a slice of a specific virtualized resource.

NOTE 2: The behaviour of the programmability-related primitives, the inclusion of more primitives that may be required, is a subject for further elaboration. This implies that multiple run-time instances of the same types of a DE may be created in the same physical resource/box.

Another possibility could be that the instance of the Function-Level-DE created by the Node-Main-DE should be the one to further create instances of its associated Managed Entities (MEs) with the possibility of creating multiple instances of the same type of an ME (provided that the ME as a resource, supports the possibility to create multiple instances that do not interfere with each other undesirably). This means that, for example a Function-Level-DE instantiated by the Node-Main-DE, may autonomically start and manage multiple instances of the same type of an ME, apart from starting and managing other types of its associated MEs that may not support multiple instances of their type of an ME. For example, the Function-Level-Routing-Management-DE (i.e. a specialized Function-Level-Control-Plane-Management-DE) that autonomically manages and control Routing Protocols and Mechanisms on a node/device may create multiple instances of an OSPFv3 Routing Process and collectively manage them, together with say a BGP process if the node/device is meant to run the two routing protocols simultaneously.

External supervision systems (e.g. OSS/BSS applications or external autonomic management systems) have to cope with virtualization: they shall extend their information model in order to include the concept of virtualized resource and aggregations of virtualized resources; for instance, these new information data shall be defined in accordance with the slice parameters. This extension does not depend on the adoption of autonomic solutions for supervising with virtualized resources.

Moreover, in order to interact with autonomic features introduced for handling virtualization of physical resources, these external supervision systems shall interact either with the DEs supervising single slices or with the Hypervisor function of the Node-Main-DE. In the second case, Hypervisor's DE should implement functions:

1) to forward to external systems the information collected from DEs supervising slices instantiated on the physical resource it is executing on; and

2) to dispatch to them commends received from the external systems.

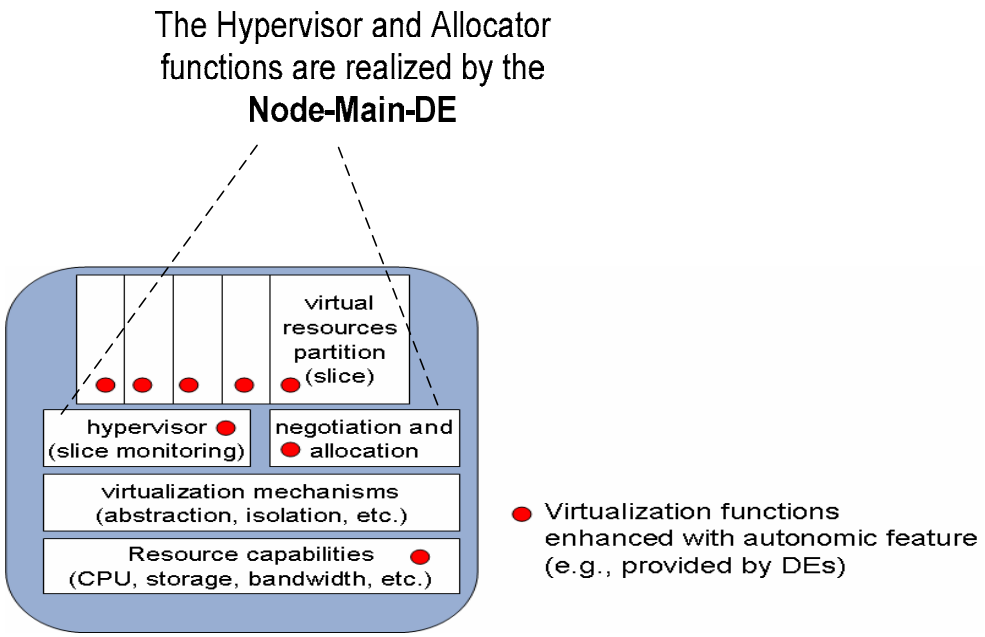


Figure 40: Internal function architecture of IT-based virtualized resources

The autonomic features associated to hypervisor and allocation functions in the Node-Main-DEs shall also cooperate e.g. by exchanging information, and events. Moreover, in the same way, they shall cooperate with the DE controlling the resource as a whole, i.e. a specific Function-Level-DE.

The DEs associated to hypervisors and allocation functions related to the virtualised resources of the same type (e.g. processing, or storage) can interact in order to cooperate in some management activities and to exchange information. These DE-to-DE interactions can be performed though links of an overlay network. They can be used to perform activities to detect/recover failures, optimize resource efficiency and balance load, perform cooperative policies in slice allocation and migration.

9.15.2 Autonomic capabilities in Virtual Networks

Network virtualization is a powerful technique as it provides flexibility, promotes diversity, and promises security and increased manageability: network virtualization is allowing multiple heterogeneous network architectures to cohabit on a shared physical network substrate. Figure 41 gives an illustration of network virtualization. The solid lines symbolize the network connections between nodes and the dashed lines symbolize the mapping between layers. The concept of network virtualization is not new - refer to [i.16] and [i.63], where details on how the subject of network virtualization is being addressed today.

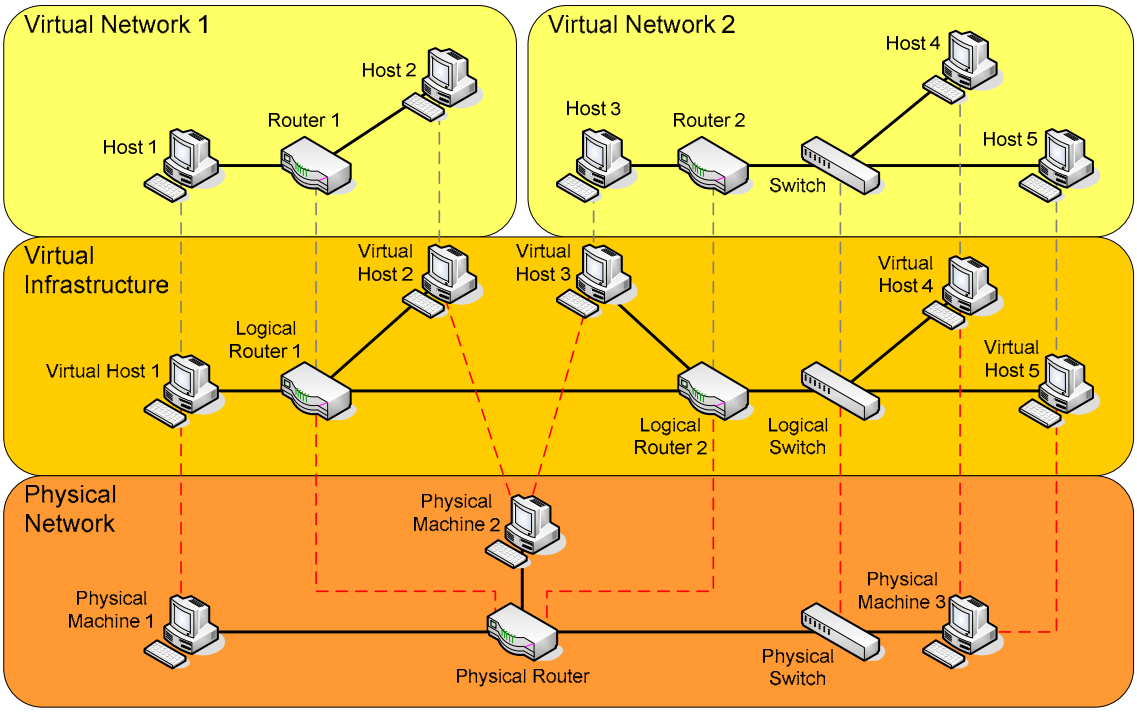
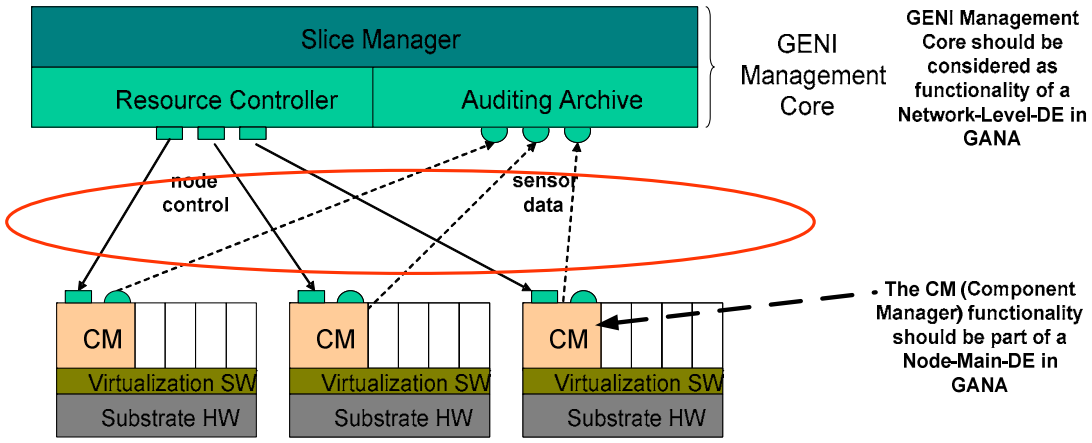


Figure 41: A generic example of Network Virtualization

Introducing autonomics in Virtual Network resources will allow exploiting self-* behaviour in the network. Network architecture will be able to adapt (and even move) its resource capacity in response to the application's needs and requirements. For example, it will be possible to relocate a virtual autonomic router (or control plane) to new physical hosts and node in case of increased workloads.

9.15.3 Mapping GENI into the GANA Model

To illustrate how network virtualization concepts can be incorporated into the GANA Model, the approach proposed by the GENI project [i.46] is considered. There are a number of concepts (it not all) in GENI that can be incorporated into the GANA Reference Model. Here is illustrated how GENI can be fused with GANA so as to reason on design principles using a single unified model.



NOTE: This figure was extracted from the [i.46]: Copyright of the original figure belongs to the GENI [i.46].

Figure 42: Mapping GENI onto the GANA Model

9.15.4 GENI virtualization approach when applied in GANA: Autonomic Virtual Router case

A classical router architecture is composed of two planes; the upper control plane where various routing protocols maintain the routing information base (RIB) and the lower forwarding plane containing the forwarding information base (FIB). The RIB injects the most suitable next-hop router for every available destination into the FIB. The actual processing of packets to be forwarded by the router is performed in the forwarding plane by the means of appropriate classification using the FIB, queuing and scheduling techniques.

The Control-Plane, considered a sub-plane of the **GANA Dissemination Plane**, is considered by the networking community as an extension of the Management Plane. The Control Plane itself needs to be autonomically managed and controlled by specific DEs. At the FUNCTION-LEVEL in GANA, there ought to be a "Control-Plane-Management-DE" that autonomically manages control plane protocols and mechanisms such as routing protocols. Due to the need for "further specializations" of the Control-Plane, there is a special type of such a DE, called the Function-Level Routing-Management-DE that needs to work together with a counterpart on the Network-Level (Network-Level-Routing-Management-DE). This means other types of specialized control-plane management related DEs need to be introduced in a similar fashion for autonomic management and control of other types of control protocols and mechanisms such as GMPLS, etc.

Regarding the Data Plane, at the FUNCTION-LEVEL in GANA (i.e. Level-2), there ought to be a Data-Plane-and-Forwarding-Management-DE that autonomically manages Data Plane protocols and mechanisms. In GANA, the Data Plane consists of protocols and mechanisms that handle individual packets (extending up to the traditional layer 4 protocols such as TCP and UDP) based on the state that is output by the Decision Plane (i.e. the Data Plane Management-DE in particular). This state includes the forwarding tables, packet filters, link-scheduling weights, and queue management parameters, as well as tunnels and network address translation mappings (*Definition adopted but with modification from the 4D architecture [*i.25*]*). Example elements of the Data Plane i.e. protocols or mechanisms belonging to this plane are: IP Forwarding, Layer 2.5-Fowarding, Layer 2-Fowarding, Layer 3-Switching, Layer 2.5 switching e.g. MPLS, Layer 2-Switching, Physical Layer technologies. For more information on the types of Managed Entities (MEs) assigned to be autonomically managed and controlled by specific DEs, refer to the DE-ME Mappings table provided in the present document (see clause 9.11.5).

The creation of a complete virtual router requires both a virtualized control plane and a virtualized forwarding plane. Using virtualization techniques leads gaining the ability to isolate different users' systems from one another whilst enabling each user to have a significant amount of flexibility in what they run and how they configure it, to the extent where different layer 3 implementations can run concurrently, assuming the same layer 2.

Existing virtualization techniques, such as XEN, work well as a solution for implementing virtual control planes. XEN, for example, provides us with the ability to restrict the proportion of the CPU or memory a virtual system is allocated, along with restricting access to the memory space of the other virtual systems. A complete router platform, such as XORP, can easily be run in the virtual system with only a small amount of additional work required to provide the interconnect between the RIB and the FIB (refer to the XORP project for more details).

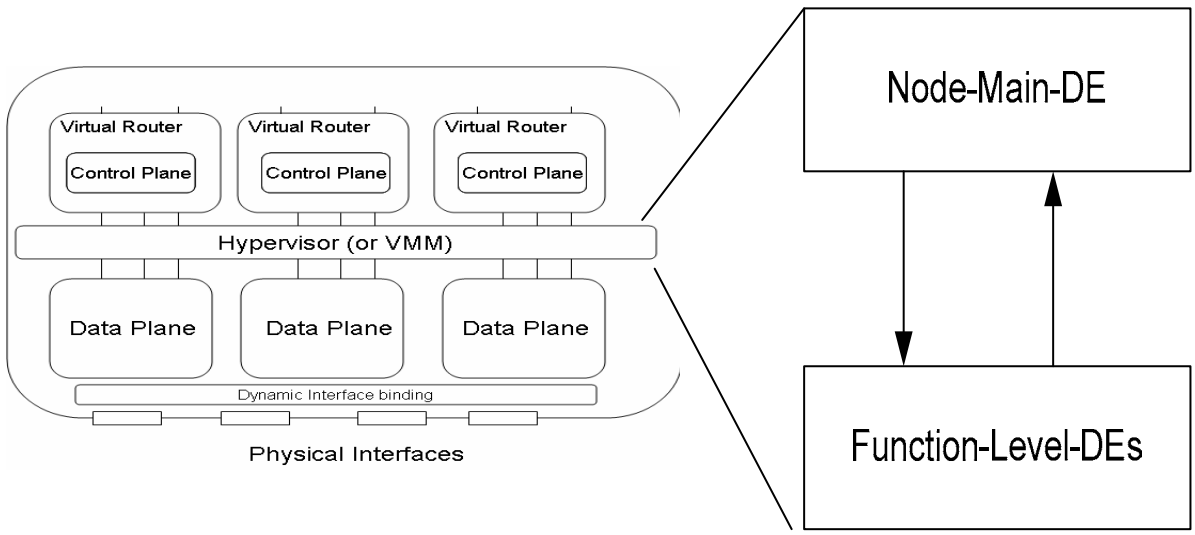


Figure 43: Example of Virtualised Router

NOTE: Though here are given some descriptions of the roles Decision Elements play with respect to virtualization, the complete picture on a ***hierarchy of control loops and the associated Decision Elements*** that drive the control loops still needs to be elaborated. Also, extending the virtualization concept to a ***Data Center providing IaaS (Infrastructure as a Service)***, is another perspective that needs to be elaborated in the subsequent releases of the present document.

It should be noted, using entire system virtualization and undertaking forwarding in the virtual system results in poor performance (e.g. forwarding plane in Linux kernel). So whilst a virtual system is the ideal place for a router's control plane it is not an ideal location for a forwarding plane.

There is a need to investigate a different scheme for virtualizing the forwarding plane of networking systems/nodes. One important requirement is the re-programmability. For instance, in order to support a number of virtual routers on the same physical hardware it should be possible to run multiple virtual forwarding paths in parallel. With this enabling capability, it might be possible developing self-organizing mechanisms for forwarding paths (e.g. for adapting to the incoming traffic or a new forwarding path experiments without the undesirable property of affecting the operation of the existing network). In this direction, OpenFlow is an interesting approach.

9.15.4.1 OpenFlow and GANA

OpenFlow claims to add a feature to switches, routers, access points, allowing these datapath devices to be controlled through an external, standardized API. (e.g. add/edit/remove flow table entries). Specifically, a datapath of an OpenFlow Switch consists of a Flow Table, and an action associated with each flow entry. In other words, in OpenFlow the network datapath is controlled by one or more remote controllers that can run on a PC. These remote controllers and their role are in essence the *Network-Level-DataPlane-And-Forwarding-Management-DE* in GANA.

An OpenFlow Switch consists of at least three parts:

1) A Flow Table, with an action associated with each flow entry, to tell the switch how to process the flow.

2) A Secure Channel that connects the switch to a remote control process (called the controller, i.e. *Network-Level-DataPlane-And-Forwarding-Management-DE* in GANA), allowing commands and packets to be sent between a controller and the switch using.

3) The OpenFlow Protocol, which provides an open and standard way for a controller (i.e. the *Network-Level-DataPlane-And-Forwarding-Management-DE in GANA)* to communicate with a switch.

OpenFlow supports network virtualization: the same hardware forwarding plane can be shared among multiple logical networks, each with distinct forwarding logic. Specifically slicing can be performed basically with two methods: using VLANs and with the FlowVisor.

In general the introduction of GANA DE autonomic features can be associated to Hypervisor and Control Plane:

- Hypervisor function of the Node-Main-DE should provide the similar autonomic capabilities described above for the IT-resources.

- Control Plane's DEs, e.g. the Routing-Management-DE: should provide autonomic capabilities to optimise connection management for the data plane (where the actual forwarding logic stands) for all interfaces (both packet-switched and non-packet-switched).

DEs associated to hypervisor function in the Node-Main-DE and to the Control Plane (e.g. Routing-Management-DE) should also cooperate e.g. by exchanging information, and events. Moreover, in the same way, they shall cooperate with the DE controlling the resource as a whole e.g. with the DataPlane-And-Forwarding-Management-DE.

Analogous considerations discussed for the virtualization of IT-based resources apply also to the virtualization of network resources, for instance, those concerning the interaction with external supervision systems, or the dynamic creation of instances for DEs supervising virtualized network capabilities, such as virtual private networks or virtual links.

9.15.5 Formalism for virtual resource request and negotiation

Virtual resource request and negotiation require understanding how to represent information (or knowledge) related to network and resources. It could be both numeric and semantic. Regarding the latter, a language might be necessary to describe both network resources and the related services in an abstract way. This means creating an ontology of resources and related data. A possible approach might be adopting a reference language developed by W3C. The idea of using RDF to describe network concepts has been already considered: for instance Network Description Language (NDL) [i.41] provides ontology for computer networks that can be easily extended, even if this approach does not fully cope with handling network resource and services requests (from Service/Applications Layers). Another interesting proposal is the Network Resource Description Language (NRDL). Through NRDL each communication can be enriched with detailed information about QoS and network requirements (i.e. bandwidth, jitter, delay, etc.).

9.16 Stability in Autonomic Networks : Stability in GANA

In general, towards realizing networks' autonomic behaviours, the presence of control-loops in the system is essential. Inputs to a control loop consist of various status signals, information and views continuously exposed from the system, component(s) or resource(s) being controlled (e.g. protocols, nodes, functionalities, etc.), along with (usually policy-driven) management rules that orchestrate the behaviour of the system or component(s). Outputs are commands to the system or component(s) to adjust its operation, along with status to other autonomic systems. Practically, control loops consist of iterative and (most of the time) distributed algorithms, that enable various node's self-* behaviours and hence, guide them to act in line with their own optimization goals or towards achieving global optimal network objectives. Henceforth, future autonomics envisions the aggregation of node-scoped control loops, i.e. within a node, in terms of interacting intra/inter-node control loops or triggered/managed low level control loops by higher level control loops within the node or the network. Intuitively, the above view leads to a hierarchal control loops paradigm that enables the efficient design of autonomic nodes and architectures [i.15].

As the designers of a network's architecture increase its autonomic attributes, in terms of introducing various self-* functionalities (i.e. control loops) at node or network level, the inherent issue of stability becomes more and more complex and crucial. In general, the stability of dynamic systems has been extensively studied since the beginning of *Control Theory*, but when viewing it through a dynamic networking environment, additional drawbacks and challenges emerge [i.18].

In most cases, stability is defined as a property of a (dynamic) system or element through which it is reassured (it reassures) that its output will ultimately attain a steady state (i.e. the state when the recently observed behaviour of the system will continue into the future). When considering an autonomic networking environment, where multiple self-* functionalities shall operate, interact and collaborate, in terms of node or network wide control loops, its stability implies reaching an equilibrium point over a finite time frame. In other words, the system/network should be stable in the sense that parameter values change, however remaining bounded in a small neighbourhood of a final value. This is especially important since self-* algorithms mainly run completely automatic and without the possibility of manual intervention.

Despite the vital role of stability in future autonomic networking, there is a lack of a concrete framework and corresponding theoretic and designing tools for addressing and treating autonomic networks' stability. Recent attempts to address the latter are either closely coupled to the studied networking environment [i.29] or the autonomic functionality that is engineered [i.12]. Towards enlightening the evolutionary path of future autonomics via highlighting the significance and role of stability in such dynamic networking paradigms, the present document makes the subsequent contributions:

- Identifies, categorizes and discusses the key factors that affect autonomic networks' stability attributes from both theoretic and designing point of view.
- For each one, corresponding theoretic tools and concrete methodologies for studying and treating them are provided.
- Proposes a concrete framework for addressing both off-line (network designing phase) and on-line (network runtime phase) aspects of stability.

9.16.1 Stability Issues in Autonomic Networking

There are two key complementary notions of stability when designing autonomic network architectures namely, autonomic node's stability and autonomic network's stability. Node stability concerns the interactions of control loops (self-* functionalities) that exist within the same node, either in the same or different layers of its protocols' stack or concerning various networking functionalities. Network stability copes with the interactions among control loops located in different components of the network; therefore, a two steps approach is required, i.e.:

- Initially, the proficient collaboration of control loops towards reaching a stable outcome shall be reassured without considering the drawbacks imposed by the networking environment (i.e. assuming a deterministic environment).
- Then, the same study should be repeated when the proposed autonomic approach (e.g. mechanism, architecture, etc.) functions/operates over the actual network. At that point, additional challenges emerge that may affect or prevent the efficient collaboration of control loops, like the presence of churn, the loss of communication between nodes (and their corresponding control loops) or even, the existence of miss-behavioural or malicious autonomic nodes.

In either of the previous cases, a methodological approach is required when addressing and studying stability in an autonomic networking paradigm, placing special emphasis on the communication and collaboration among the introduced control loops, regarding their managing, managed, peering and sibling relationships. Specifically, the following key issues shall be considered:

Issue I. Self-* functionalities (i.e. control loops) interactions and their convergence at a stable outcome (i.e. equilibrium point). Thus, it is vital to reassure that only important changes cause the triggering of actions or the recalculation of parameters (in order to reduce the causes of instability). Towards this direction, we introduce and exploit well established guidelines that stem from concepts in the fields of Game Theory and Optimization Theory towards:

- Establishing well-defined valid operating regions of particular control loops.
- Decoupling control systems by ensuring that they control different independent outputs based on independent inputs and if this is not possible, then tuning them so that they impose control at very different timescales. Such an approach will allow us to decouple systems that would otherwise be strongly coupled.

Issue II. Conflicts resolution, not only between self-* functionalities and behaviours (i.e. control loops) belonging to different network components but also among control loops within the same node. Conflicts may occur when more than one control loops manage or affect the same functionalities or resources, especially when they are placed in a hierarchical manner.

Issue III. Time scaling issues among collaborating control loops. In this case, the main difficulty is that timing varies significantly when considering various self-* functionalities which are interacting, while residing on different layers of a node's/network's protocols stack. Thus, it shall be reassured that the following two dynamic and interacting sets of "events", that is {signalling, monitoring information and decision making} and {changing environment and triggering events}, are changing on a similar timescale.

In the following we analyze each of the above critical issues and we provide concrete methodologies and key theoretic tools for addressing and treating them.

9.16.2 Designing for Stability

This clause places emphasis on revealing vital aspects related to stability (i.e. control loops stability) that need to be addressed while still being at the design time of an autonomic network and its corresponding autonomic elements, such as the GANA DEs. Hence, we propose solutions and corresponding theoretic tools to address the latter, in contrast to aspects of stability that shall be handled at run-time (i.e. as the autonomic system evolves) which are highlighted in clause 9.16.2.1.

9.16.2.1 Stable Autonomic Behaviours Design through Game Theory - From Theory to Theory

Despite the variety of alternative autonomic architectures that may emerge when obtaining the solution of the corresponding initial system's optimization problem, one common designing attribute characterizes them, the decentralized nature of the consequent autonomic algorithms/mechanisms. This not only necessitates the collaboration of various network components to achieve different layering objectives, but also implies the distribution of the decision making procedures of the network among its components, instead of traditional centralized approaches, which eventually increase the role of individual network components.

Aiming at composing efficient distributed and iterative autonomic algorithms, appropriate designing theoretic tools need to be adopted, which are promoting network's autonomic nature. To that end, Game Theory and Network Utility Maximization (NUM) theory via its decomposition methods [i.24] (i.e. a theoretical tool for designing optimal distributed algorithms over layered architectures) consist of some of the most widely adopted mathematical frameworks applied in autonomic networking. On one hand, these analytical tools can be used to derive distributed algorithms and determine their efficient collaboration (i.e. autonomic node's DEs (i.e. control loops) signalling), providing theoretically founded answers to the question: *who does what in the autonomic network and how to connect them.* On the other hand, they inherently provide methodologies not only for investigating and reassuring that the corresponding derived distributed algorithms are reaching a stable outcome, in terms of equilibrium points, but also for enabling the following desirable designing attributes:

a) limited amount of overhead among nodes (and control lops);

b) fully asynchronous updates; and

c) robustness to arbitrary signalling information [i.33].

In general, the following thread of analysis holds. An autonomic network's/functionality's stability is assured via (a) DEs cooperation stability, which can be further reassured via (b) their corresponding control loops collaboration stability and thus, via (c) proving the stability of the distributed iterative algorithms that steer autonomic node's control loops, obtained via the solution of the corresponding optimization problems.

In the following clause, we place emphasis and analytically present a concrete methodology for studying autonomic mechanisms' stability via game theory.

9.16.2.1.1 How to Treat Stability via Analytical Methods? - A Game Theoretic Approach

Game theory is currently widely explored in autonomic networking due to the following two main inherent attributes:

i) Autonomic nodes' selfish non-cooperative nature can be properly defined and formulated as a non-cooperative game, where nodes act as individual players aiming at maximizing their own interests. Then, game theory provides the foundations and mathematical tools for studying and solving such problems.

ii) The distributed nature of the produced algorithms, which seek non-cooperative game's solution and steer corresponding autonomic nodes DEs (i.e. are their control loops), facilitate the efficient design of a network's autonomic mechanisms.

A non-cooperative game consists of three basic components:

iii) a set of players S, consisting of N autonomic nodes;

iv) a set of actions A_i, i.e. the feasible strategy space of i autonomic node; and

v) a set of preferences, which can be expressed via appropriately defined utilities $U_i(a)$ of each autonomic node, where $a_i \in A_i$.

Thus, the corresponding non-cooperative game is formulated as follows:

$$\max_{a_i \in A_i} U_i(a_i)$$

$$s.t.\ \ a_i \in A_i$$

Furthermore, regardless of the designed problem's formulation, a non-cooperative game's investigation requires the consideration and analysis of the following key steps:

a) Game's steady state via the existence of equilibriums (e.g. ***Existence of Nash equilibrium***).

b) Equilibrium's properties (e.g. ***Nash Equilibrium's properties***).

c) Optimality of equilibrium (e.g. ***Pareto optimality***).

d) Convergence of equilibrium via studying the convergence of users' corresponding (best) response towards selfishly maximizing their utilities, thus reaching game's equilibrium. (e.g. ***Convergence of Nash equilibrium***).

9.16.2.1.2 How to address stability via Game Theory?

Steps A and D can provide the answer to the previous question. Specifically, upon setting a non-cooperative game, with respect to the required behaviour of the autonomic nodes in the network, and upon deriving distributed algorithms that steer nodes to have the expected behaviour (i.e. autonomic node's control loops), the ability of the corresponding autonomic network to reach a stable outcome (i.e. an equilibrium point) can be analytically proved:

1) By showing the existence (or even the uniqueness) of such a stable point (**step A**).

2) Via proving that the derived distributed (and often iterative) algorithms (i.e. control loops) will always reach such a point **(step D**).

Intuitively, that latter suggests that autonomic nodes' interactions, via their corresponding DEs (control loops), will always reach a stable outcome; thus, leading to autonomic network stability, under the assumption that DEs communication synchronization is always achieved. Game theoretic tools that allow treating issues A and D are super-modular games, utility functions' properties (e.g. quasi-concavity), potential games, coalition games etc. Moving one step forwards, the stochastic nature of the networking environment should be considered (i.e. dropping the assumption of reassured DEs communication and synchronization). This makes the analysis of the corresponding games much more difficult. To that end, stochastic games formulations can be applied. A deeper analysis on game theory in autonomic networks is out of the present document's scope; interested readers are referred to [i.21] and [i.37].

9.16.2.1.3 Addressing Stability in an Architectural Level - From Theory to Practice

Apart from treating stability via analytical theoretic means, in the following we identify a plethora of vital autonomic network architectures' designing aspects that play a crucial role in determining their stability attributes. Throughout our analysis, the benefits of adopting GANA [i.15] for devising stable autonomic networks are also revealed, since GANA inherently adopts (i.e. inherent architecture's attributes) the key prerequisites illustrated below.

9.16.2.1.4 Hierarchy of Control-Loops (DEs)

An important feature of an autonomic architecture is to maintain a hierarchy of control-loops (as defined in GANA). The benefits of introducing hierarchy to manage complex autonomic systems are extensively justified in [i.18]. From an autonomic network's stability point of view, the introduction of hierarchy allows the horizontal (among network nodes) and/or vertical (different functional levels) partition of the decision making process. Thus, the failure of a single entity or DE will not result in the total failure of the network under control. DEs with independent goals and policies will continue to operate and manage their corresponding protocols, allowing the failed DE to restart or "correct" itself in the mean time. This feature of GANA allows the network (or at least part of functionalities) to be stable in spite of the failure of a single DE or few DEs.

Moving one step forward, some of the major drawbacks of traditional hierarchical systems are large convergence time, sensitivity to failures, large computational power requirement and communication overhead. GANA on the other hand does not follow the traditional hierarchical systems architecture; i.e. it's distributed hierarchical system architecture. Thus, it allows communication between sibling and peer DEs, providing a distributed solution to control its MEs.

To that end, GANA incorporates the following key design principles:

1) Lower level DEs expose "views" up the Decision Plane, allowing the upper (slower) control loops to control the lower level (faster) control-loops driven by lower level DEs).

2) Changes computed in the upper DEs implementing slower (i.e. within larger time frames) Control-Loops are propagated down the DE hierarchy to the Functions-Level DE(s) implementing the faster control-loops that then arbitrate and enforce the changes to the lowest level Managed Entities (protocols and mechanisms).

This operation reduces the drawbacks of large convergence and sensitivity times, traditionally found in conventional hierarchical systems. Finally, the nature of the distribution of the tasks to DEs inside the node and Network-Level DEs further ensures that the communication and computation power requirements are kept to a bare minimum inside the node.

9.16.2.1.5 Concept of "Ownership"

The "Concept of Ownership" is another feature of the intrinsic stability attributes that shall be considered for autonomic network architecture (as is also defined in GANA). This concept requires that every ME parameter is managed by a single DE, i.e. no two DEs (i.e. control loops) can control the same ME (i.e. functionality, parameter, resource, etc) at any given point of time in the network. This is important from system's stability point of view since it relieves the burden of "*conflicts resolution*". Specifically, if an ME (i.e. ME parameter) is controlled by two or more DEs at the same time, contrasting, conflicting and at times repetitive policies, objectives and reconfiguration requests, etc., originating from different DEs lead to an unstable ME and thus, to an unstable autonomic network. Through the "Concept of Ownership", GANA ensures that this instability is avoided.

9.16.2.1.6 Separation of "Operating Regions"

Another prerequisite of an autonomic architecture is the efficient separation of the "operating regions" for the control-loops as advocated in [i.34]. This can be achieved by decoupling control systems and ensuring that they control different independent outputs based on independent inputs, as defined in GANA. If it is impossible to decouple certain outputs and inputs from affecting each other, it is important to reassure that they impose control at different timescales on their MEs.

9.16.2.1.7 Model-based Techniques

Model-driven approaches for design DEs and their inter-relationships should also be exploited to efficiently address aspects related to stability of control-loops. Specifically modelling and validation of DEs autonomic behaviours using Formal Description Techniques (FDTs), such as the well-known and successful ITU-T SDL language, can be explored to address certain aspects of stability. Such an approach would enable the design, model-checking and verification of DEs, as well as the validation, simulation and some partial code-generation of autonomic behaviours of DEs. Finally, the aspects (i.e. methodology) [i.35] that need to be taken into account when following such an alternative are:

1) A Meta-Model that enforces constraints in the design models for DEs and their interactions with assigned MEs and with other DEs, that should be embedded in the Modelling Tools e.g. a "model editor", is required.

2) A Model-Walker that can be designed for walking over a design model in order to detect conflicting control-loops and overlaps in the so-called "valid operating regions" of control-loops specific to DEs.

3) Simulations for detecting behaviour conflicts between interacting control-loops designed to operate within a node/device and in the network.

9.16.3 Addressing Stability at Runtime

After applying the methodologies and guidelines presented in the previous clauses, an autonomic network is intrinsically equipped with interacting control loops having some degree of self-stabilization. However, it is possible that due to the stochastic nature of the networking environment, situations may occur, which have not explicitly been considered during the design time of the Decision Elements. We denote such situations as *"emergent situations/behaviours"*. Intuitively, "emergent behaviours" can be considered as an indirect interference between a set of DEs, whereby each DE is optimizing its own goal based on its embedded logic, but the overall set of executed actions is leading the system to an unstable state of oscillations and continuous responses of diverse control loops. In order to avoid such emergent behaviours, the concept of Action Synchronization Functions (ASFs) has been proposed in [i.39] (which is also part of a GANA's Decision Elements). In simple words, [i.39] introduces ASFs in the GANA hierarchical structure, in order to allow DEs on a lower level to resolve potential conflicts via requesting DEs belonging to a higher level (in the DE's hierarchy) for taking over their synchronization and coordination. Specifically, after a number of DEs have referred to a higher level DE in order to be informed whether particular tentative action(s) are allowed or not, the higher level DE selects the optimal subset of tentative actions reported by the corresponding lower level DEs. The tentative actions are gathered over a pre-defined time window. Consequently, the higher level DE responds back to the requesting lower level DEs on whether they are allowed to proceed with executing tentative actions or not.

There are various aspects of stability which are addressed by the architectural role of the ASF namely:

1) Acting as an arbiter for conflict resolution among DEs with potentially interfering actions;

2) Exploiting the GANA hierarchical structure in order to realize the notion of hierarchical optimal control - autonomic entities on a higher level have access to more global information; and

3) Enabling the gathering of actions for synchronization over a pre-defined time period has a smoothing effect on the rate at which control loops operate, thereby avoiding sudden oscillations and chaotic situations in the autonomic network.

In [i.39], the issue of conflict resolution is tackled via deriving a binary integer program, which aims at the optimization of a set of key performance indicators (KPI) by selecting an optimal subset from the actions waiting to be synchronized.

9.16.3.1 Autonomic-aware Metrics to Infer and Self-assess Stability

Since an autonomic network is expected to be self-adaptive to changes and challenges in its environment during its operation, it shall be able to self-assess and infer stability related problems experienced at various levels of the Decision Plane Hierarchy. The assessment of stability at different levels where a DE implements an autonomic functionality (e.g. autonomic routing, autonomic QoS-Management, autonomic Mobility-Management, etc.), requires that every DE shall implement some monitoring functionalities towards self-awareness (i.e. implement *Counters* for storing statistics e.g. i) the number of times a DE enforced a change the behaviour of its assigned Managed Entities (MEs) in reaction to specific events over a period of time, or ii) the number of times a DE received information indicative of instability problem from its MEs). What is also required, are *Timing Variables* (e.g. timers) for storing time durations that measure some DE activities. Diverse types of Timing Variables are required for each type of input that flows into a DE-according to the "valid operating region" of its associated control-loop that was captured and defined at design time.

Thus, every DE shall expose the "views" captured by the *Counters* and the *Timing Variables* to its upper DE, which then assesses the degree of stability of the autonomic functionality realized by the particular lower level DE and decide to enable an appropriate response strategy towards reassuring stability. The upper DE may further aggregate some statistics and "views" and expose then further up the Decision Plane (possibly up to the level of network-level DEs) where more sophisticated decisions can be taken based on the wider knowledge about the network that is gathered by network-level DEs. Finally, an autonomic behaviour triggered by a DE reacting to a change may inductively span a number of DEs, and their associated control-loops, which are enabling (via collaborations and interactions) *ultimate goal* (i.e. an autonomic behaviour). Such an *ultimate goal* could be the (re)-setting of a parameter value on a single ME or multiple parameters on an ME(s) managed by the first triggering DE in the DE interaction chain, or could be the setting of multiple parameters on ME(s) managed by one or more other DEs involved in the DE interaction chain. A DE e.g. a Node-DE, belonging to a higher level than the DEs involved (which could be Function-Level DEs), provided it knows the *causality graph for actions/policies employed by different DEs* to achieve the *ultimate goal*, could then use information stored in *Timer* variables stored by lower-level DEs in the interaction chain, in order to infer stabilization time and challenges, via using this knowledge to improve cognitive response strategies over time.

In conclusion, the issue of autonomic networks stability. Despite the existence of recent elegant analytic results in specific networking paradigms, the issue of stability in autonomic networking still lacks of concreteness, generality and mainly extensive validation; hardening the wide applicability of autonomic solutions in realistic environments. Following a pragmatic and concrete thread of analysis we identify, categorize and discuss all the key aspect that affect or characterize autonomic architecture stability, from theoretic analysis and network design point of view, to practical implementations and runtime solutions. Moving one step further, we propose concrete methodologies and highlight corresponding theoretic tool for addressing and studying stability problems in autonomic networks.

10 Network Governance

10.1 GANA Network Governance Interface

This clause addresses, Service Management View, the Use of Policies, Policy Frameworks, and Profiles in GANA.

In a production-level autonomic network, an implicit or explicit agreement between a client and a service provider specifies service level objectives, both as expressions of client requirements and as provider's assurances. These objectives are expressed in a high-level, service-, or application-specific manner rather than requiring low-level, resource specific performance. This clause presents a framework that addresses the gap between high-level specification of client performance objectives and existing resource management infrastructures. Thus providing operators with means for decision oriented operational tasks based on the use of policies rather than low level command execution.

The self-* capabilities of Autonomic Networks require shifting from traditional command and control management paradigms towards new governance models. Nevertheless, although Policy Based management (PBM) seems to fulfill some of the governance requirements, current architectures need to be enhanced to include the key concept of knowledge. This requires using richer semantic models to allow the infrastructure taking advantage of inference and reasoning capabilities. Therefore a new enhanced semantic and knowledge based model should complement current information models such as SID and CIM, rather than inventing a new complete information model.

NOTE: The concepts related to policy-based management presented in this clause address the requirements specified in GS AFI 001 [i.51], such as those specified in the clause "Requirement framework for a Policy- based management" of GS AFI 001 [i.51].

This approach for Network governance gives the operator a mechanism for controlling the network. This is done from a high level business point of view. Thus, the operator is not required to have deep technical knowledge of the network to effectively operate it.

The network governance interface is meant to provide a mechanism for the operator to adjust the features of the demanded service/infrastructure in a high level language. In order to achieve this objective the framework should provide a business language that will help the operator to express what it is needed from the network. Such business language may be modelled by the use of ontology to add semantics and enable machine reasoning on the goals. These indications are afterwards translated to a set of policies that will clearly define the valid operating region for the autonomic functionality.

This business level language will be oriented to the definition of the desired network behaviour for each service. Since it is not possible to satisfy all the requirements that arise from the definition of all the services in the catalogue, some additional business rules for conflict resolution have to be defined, defining higher priorities for specific behaviour depending on users, services, etc.

Services converge at a network level, and at this point the GANA framework provides the GANA Network Profile as the tool to distribute the information on how the network should behave. Such a hook role for the network profile is depicted in Figure 44, where it is clearly seen that the network profiles are used to join service and network levels. Moreover, the concept of GANA Network Profile includes how a Profile encapsulates Policies, Configuration-Data (Config-data) and Objectives, while manifesting the notion of Policy Continuum and Profile Continuum as described in this clause.

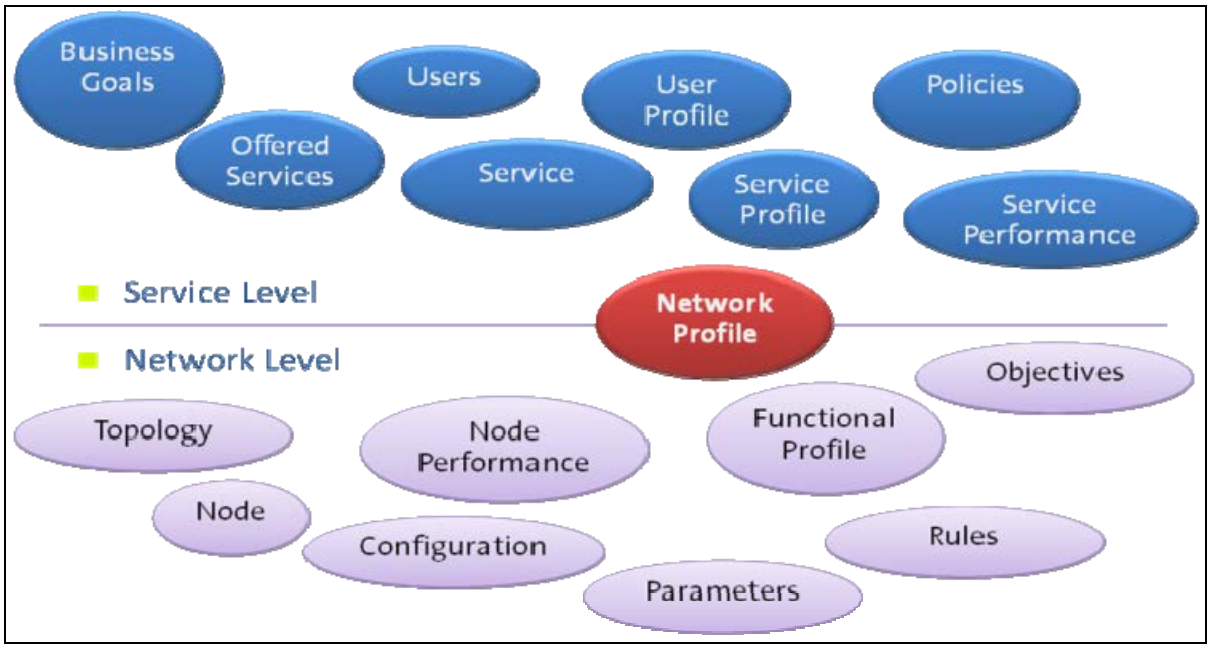


Figure 44: Network Profiles as a Hook between Service and Network Levels of Management

The mapping from Business level language and specific policies is supported by the policy semantic description. This model will describe the business level layer based on services and quality goals. Together with this service level description made by operators, vendors will provide a semantic description of the more technical layer based on resources. All the actors share the same ontology schema that contains the needed hooks for the reasoning process to translate the business needs to a specific policy language that will be later enforced in policy enforcement points.

There is no restriction on the policy language to be used. On one hand the semantic model can map business needs into actions specified in different policy languages to cover different network and vendor technologies. However we can state that policy languages compatible with the IETF architecture and CIM or SID information models are recommended. Since CIM policy language is quite complex, CIM SPL as proposed by DTMF may be a good choice. On the other hand, policy encapsulation in network profiles as defined by GANA is also independent of the policy language used. Thus any policy statement is distributed to the whole network using the same distribution mechanisms independently of the policy language used.

Summarizing, the approach presented here establishes a business layer with interfaces for network governance based on proper knowledge handling and network profile distribution mechanisms.

The GANA Network Profile as specified in clause 10.1.1 provides a mechanism for distribution of the technical representation of the network business level policies. This technical representation still consists of policies, objectives and configuration data, which however are technical and only implicitly contain the notion of services. Thus an entity is needed to convert the high level service specific policies in the (i.e. the Service Profile) defined by the operator into the GANA Network Profile. This is done by the Policy Server.

In Figure 45 a semantic model describing services, goals, users, etc is depicted as the background knowledge, then after some reasoning mechanisms over this knowledge the Policy Server will end up in the GANA Network Profile generation.

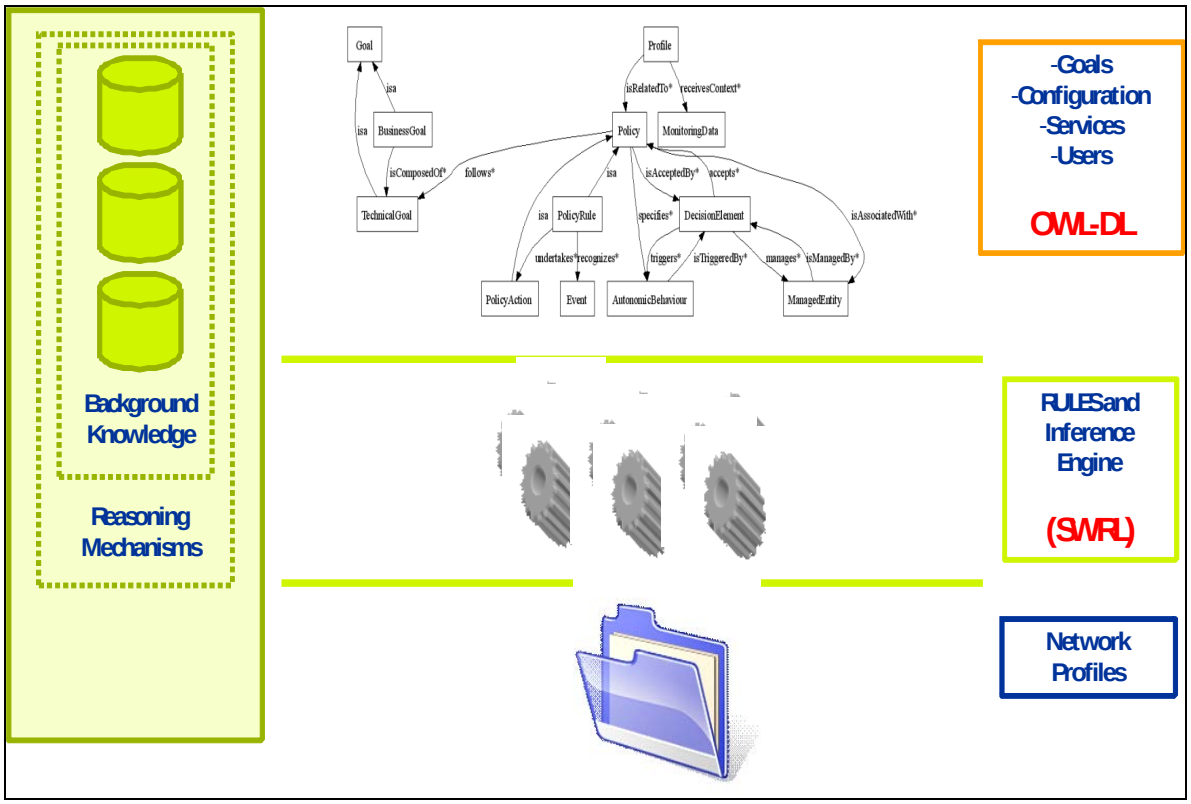


Figure 45: Knowledge Handling

Once the GANA Network Profile has been created it has to be efficiently distributed to the network using the mechanisms that GANA offers for such duty (i.e. ONIX, an Overlay Network for Information eXchange). The components of the policy framework architecture are depicted in Figure 46. The main components are:

- **GUI:** Graphical User Interface that will provide a friendly way of creating and editing policies using a high level business language. These policies compose the Service Profile.
- **Policy server:** This component is in charge of translating business language in the Service Profile to the GANA Network Profile, which contains more specific policies, objectives and configuration data. Additionally the policy server checks whether the different indications given by the operator have conflicts. If so, they will be resolved according to the rules that were also imposed by the operator. It is empowered by semantic technologies in order to enable reasoning capabilities to map the service requirements to the network level without direct intervention of the operator.
- **Policy semantic model:** This component contains the knowledge that is needed to make the translation from business and services to GANA Network Profiles.
- **The policy publisher** is the GANA mechanism to efficiently distribute network profiles to the nodes in the infrastructure (in the context of GANA Network Profiles a Network-Level-DE using ONIX performs the role of the policy publisher).
- **Network nodes:** From the policy framework point of view, it is the point where the policies are executed. They download the policies that they need in order to apply them.

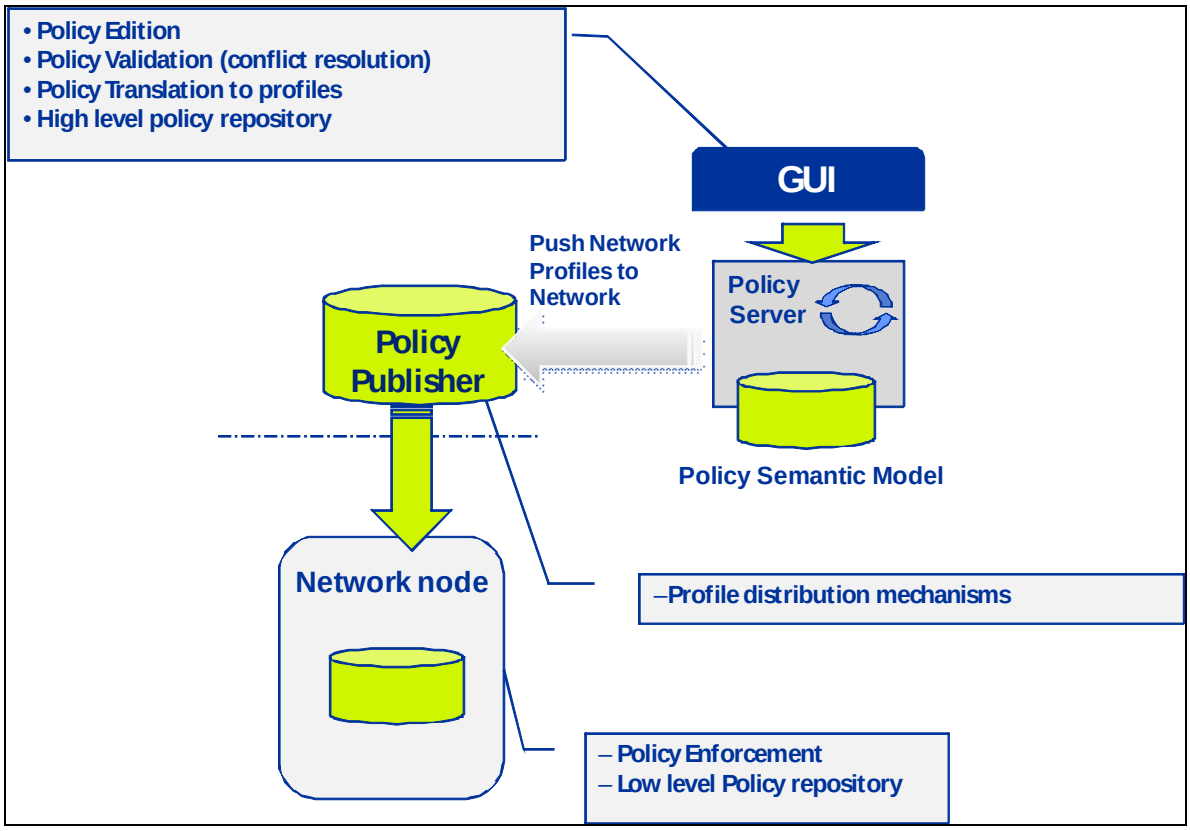


Figure 46: Policy Framework Architecture

The following use case describes how the whole process works:

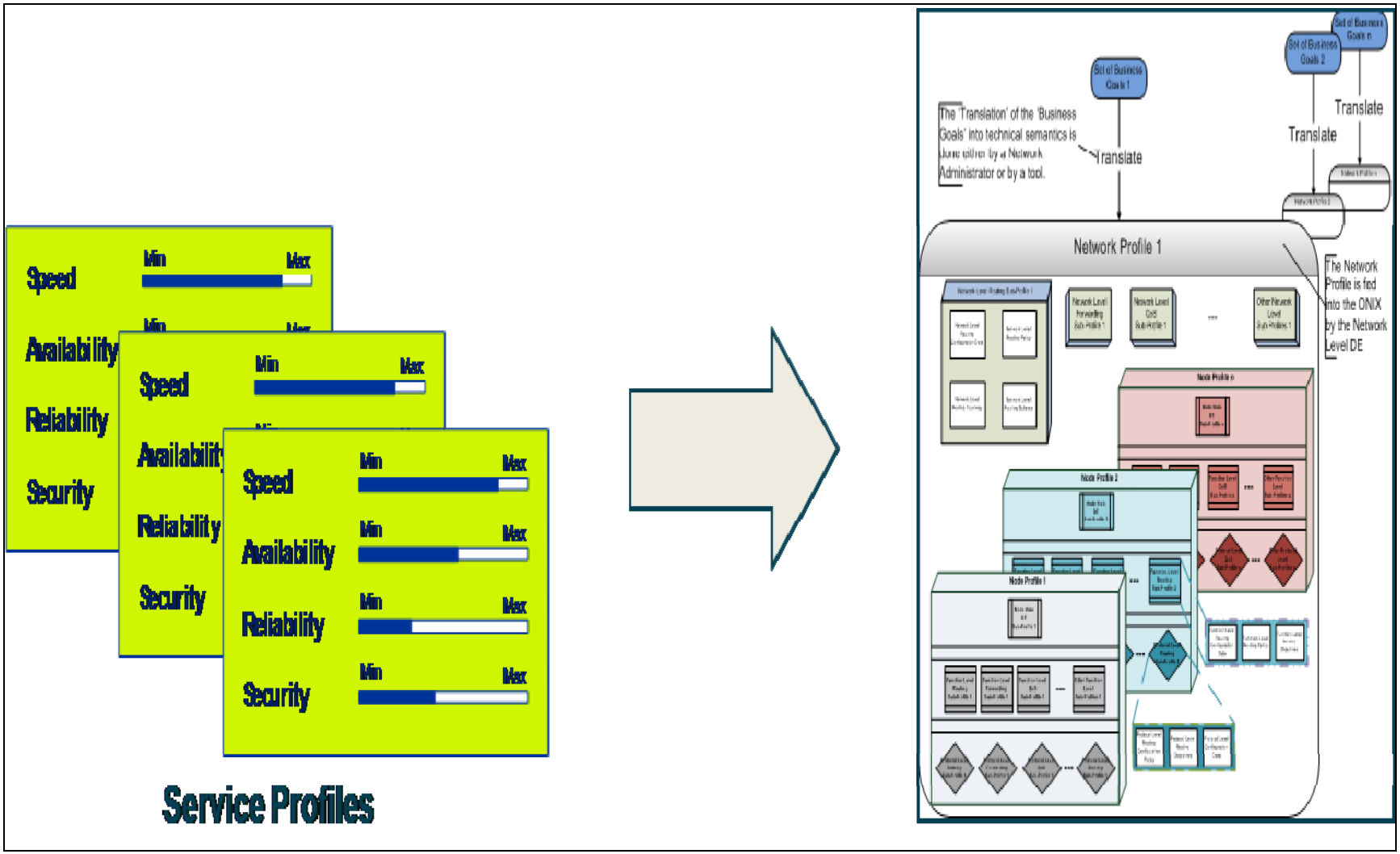
1) The operator selects the desired network behaviour for each service. Different rules can be enforced for conflict resolution based on the relevance of the service, relevance of the user, timely planned behaviour for the network, etc.

2) The operator's tool generates a GANA Network Profile. Policies are encapsulated in a Network Profile. In the process of combining service requirements at the network level, the previous rules are used for conflict resolution; e.g. increasing security may decrease speed.

3) The network profile is pushed in the ONIX and are distributed to the nodes by a Network-Level-DE (this process is described in detail in clause 10.1.2). The Policy Publisher is a role that can be fulfilled by an ONIX Information Server (since Policies are encapsulated in a Network Profile that is stored into the ONIX, and a Network Profile contains Node-Profiles that are distributed to individual nodes by ONIX). Node-Profiles encapsulate Policies, Configuration Data and Objectives as well.

Figure 47 illustrates the process described above. The operator defines a service profile using a high-level business language through the user interface. The policy framework translates it into the **network profile** to be pushed into the ONIX.

NOTE: On the right hand side of the figure, the Network Profile Structure is not **readable** but it is the same structure that is presented in Figure 50 and is **more readable**.



NOTE: The same Network Profile and is more readable in Figure 50.

Figure 47: Derivation of network profiles from high-level business goals

The approach described so far gives the operator a mechanism to govern the network behaviour. The human operator does not need to be a specialized technician, since the input to the policy framework is a set of parameters defined using a high-level business language.

Feedback Loop mechanism

The low-level derived policies, disseminated through the ONIX, establish the initial behaviour of the network elements. These network nodes, which have been provided with autonomic capabilities, have the possibility to modify their policies in response to changes in their environment. Therefore, a mechanism is needed for the operator to check if there has been any deviation with respect to the policies originally commanded. To that extent, a feedback loop should be established, able to access the current working policies of the network nodes and compares them with those defined by the operator.

Two main communication channels appear as necessary between the network governance layer and the network nodes:

a) The first one is used by the network elements to notify of modifications of their local working policies. This will keep informed the governance layer about the current runtime situation. The network governance will use these notifications to assure that the network behaviour effectively corresponds to the one previously defined by the operator.

b) The second communication channel is used by the network governance layer in order to query a given network node about its current working policies. Therefore, the possibility of inspecting the behaviour of a given node or set of nodes can be offered to the human operator through the graphical user interface.

Figure 48 illustrates this process.

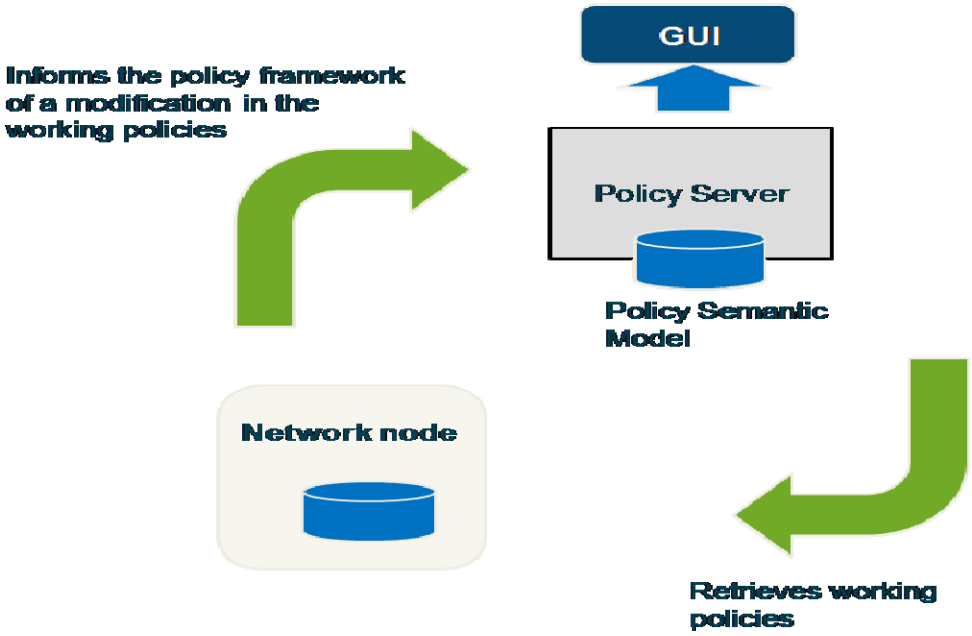


Figure 48: Graphical representation of the process involved in the feedback loop mechanism

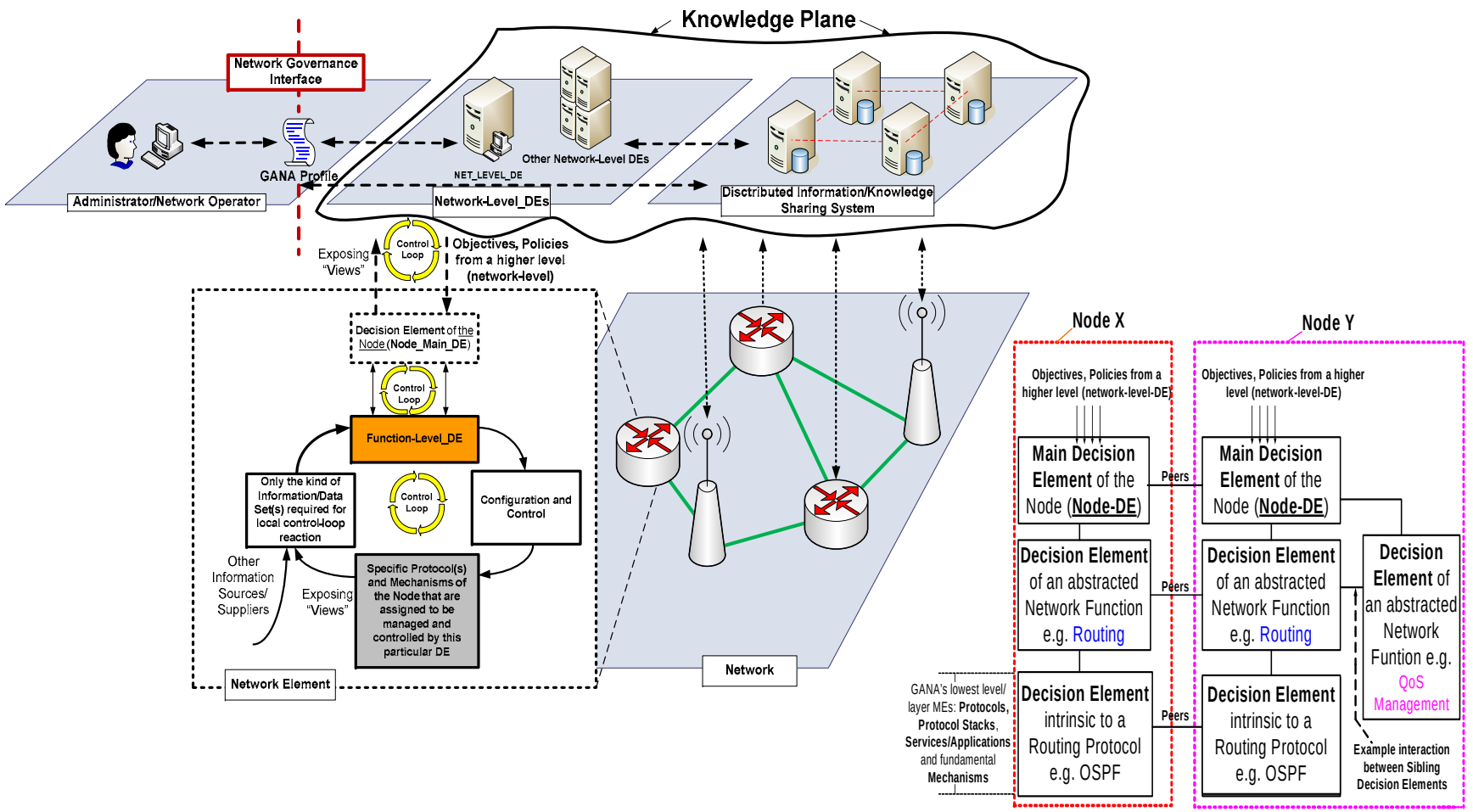
Once the policy framework receives the low-level policies (asynchronously or as a result of a query), it should compared this information with the policies that have been originally enforced. This comparison necessarily implies that both types of policies should be expressed in the same language. Therefore, the network policies should be translated to business terms,

The fact that a semantic model has been used in the business-to-network derivation procedures explained above is advantageous for the network-to-business translation needed now. To that extent, the same architectural elements can be used:

- **Policy semantic model:** This component contains the knowledge that is needed to make the translation from network profiles to business goals.

- **Policy server:** This semantic-enriched component is in charge of translating low-level language in the Network Profile to the Service Profile, which contains high-level business goals. Additionally the policy server checks whether the different indications given by the operator have conflicts. If so, they will be resolved according to the rules that were also imposed by the operator.

- **GUI:** Graphical User Interface that will provide a friendly way for the operator to inspect the current behaviour of the network.



NOTE: All the ***Types of DE Interfaces*** depicted illustrate the need for ***"node/device-intrinstic management"*** and ***"network-intrinsic management or in-network management"*** in Self-Managing Future Networks.

Figure 49: The Network Governance Interface and relation with the Knowledge Plane in GANA

10.1.1 GANA Network Profiles

AFI defines a Profile as the minimum composition of Policies, Functional Objectives and Configuration Data required as the starting blocks for achieving specified network goals and implementing network functionalities such as routing, forwarding, security, etc. A Goal, in the context of GANA is defined as the overall target of the network. Goals in turn describe the Policies, Objectives and the corresponding Configurations required for a network and its devices. It can be classified into two types - a Business Goal, and a Technical Goal. Business Goals are the business use cases for the network, and is written by the network operator/provider through the network governance interface as described in the previous clause. On the other hand, Technical Goals are a translation of the Business Goals in to technical semantics of the network. In context of GANA the GANA Network Profile is a formalised approach to specify and structure the Technical Goals of the Network. Thus every set of business goals are translated to a GANA Network Profile.

As only one set of business goals can be catered by a network at any given time, only one type of GANA Network Profiles can be applied to a network. It is to be further investigated to see whether a network can be provisioned to cater the requirements of two independent sets of business requirements; i.e. whether a given network can simultaneously apply and use two different and independent sets of GANA Network Profiles. This is a topic for future study (ffs).

In line with the definitions presented above, the GANA NETPROF, as shown in Figure 50, consists of a Network Profile (NETPROF), Vendor Specific Node Configurations Options and a Configuration Options Map. The NETPROF is a composition of Policies, Objectives and Configuration Data that is structured along the hierarchy of the GANA Reference Model. Thus a NETPROF is composed of the following:

1) Network-Level DE Sub-Profiles.
2) Node Profiles (for various node roles):
    a) Node-Main DE Sub-Profile:
        i. Node-Level Sub-DE Sub-Profiles.
    b) Function-Level DE Sub-Profiles.
    c) Hooks for Protocol and Vendor Specific Node Configurations Options.

The decomposition of the NETPROF and its relationships with the Vendor Specific Node Configurations Options and a Configuration Options Map is explained below. The relationships between the two decomposition and the reasons for this type of decomposition are explained later in clause 10.1.1.3.

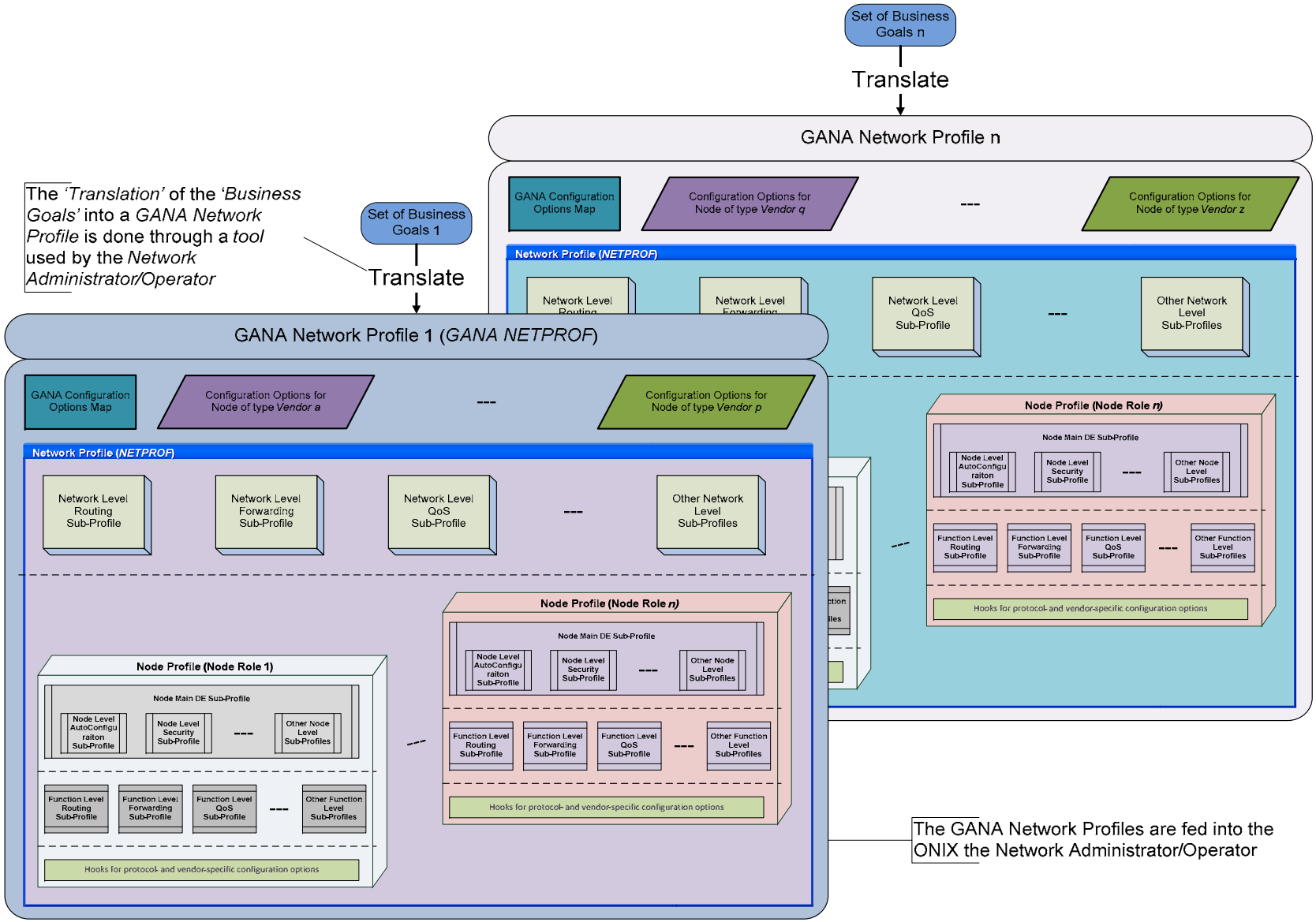


Figure 50: GANA Network Profile (for readability of some parts see Figure 51)

10.1.1.1 Vertical Decomposition

The vertical decomposition of the NETPROF is shown in Figure 51. Here the *NETPROF* is decomposed along the GANA hierarchy. Thus the *NETPROF* is composed of many *Network-Level DE Sub-Profiles* and *Node Profiles*. Each *Node Profile* is described for a certain role a node would play in the network. It shall be noted here that the *Node Profile* is for a specific role (dynamic) that a node would play, not for an individual device/node. Thus, it provides *Policies*, *Objectives* and *Configuration Data* for the various DEs and protocols/MEs of the Node, enabling it to play its prescribed role in the network.

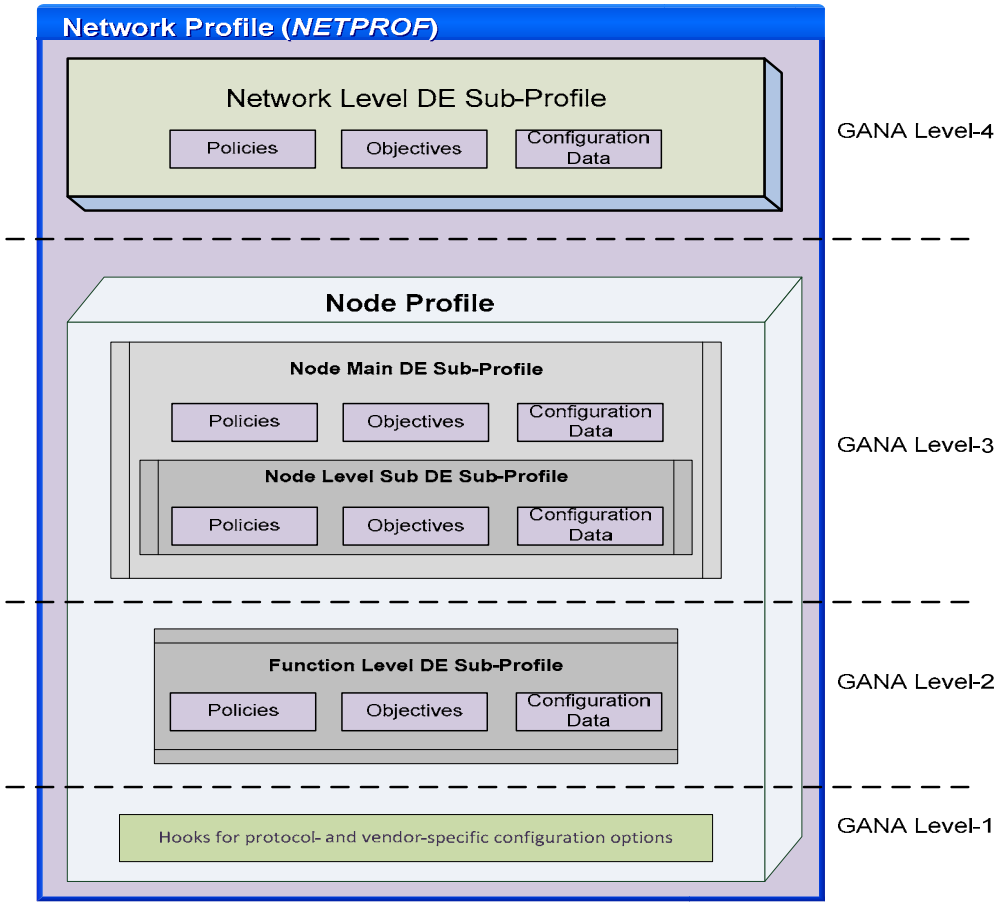


Figure 51: Vertical decomposition of the Network Profile

A *Node Profile* is composed of a *Node-Main-DE Sub-Profile* and several *Node-Level Sub-DE Sub-Profiles*, *Function-Level Sub-Profiles* and *hooks for protocol and vendor specific node configurations options*. A *Sub-Profile* at any level is composed of the *Policies*, *Objectives* and *Configurations* that are applicable for that level. For instance, a *Function-Level Sub-Profile* such as *Function-Level-Routing-Sub-Profile* is composed of *Routing Policies* (e.g. policies for setting route and protocol attributes), *Routing objectives* (e.g. objective to achieve a desired routing topology) and *Routing Configurations Data* that need to be applied by the DE controlling the routing functionality at GANA's function level.

Figure 52 illustrates this decomposition for the case of Network level Routing Sub-Profile.

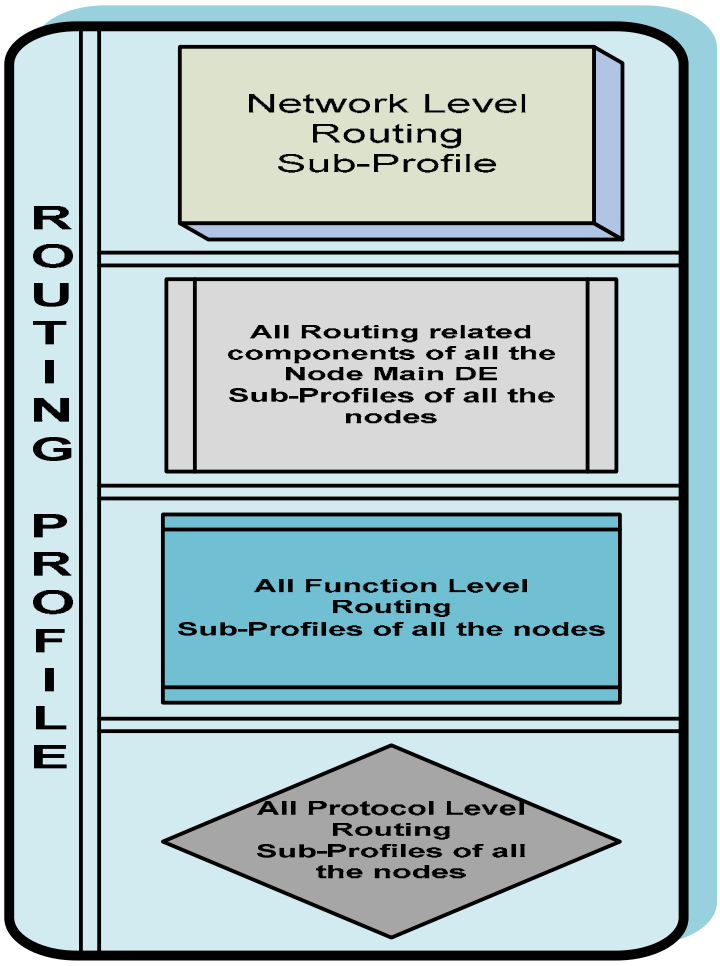


Figure 52: Vertical Decomposition of the Routing Profile

10.1.1.2 Horizontal Decomposition

The horizontal decomposition of the *NETPROF* is shown in Figure 53. Here the *NETPROF* is decomposed along the abstracted functionalities of the GANA architecture, i.e. abstracted functionalities such as Routing, Forwarding, QoS, Security, etc. Thus each of these functionalities can be considered to be contributing a profile of their own. Thus an aggregation of these abstracted functionalities' profiles composes the *NETPROF* horizontally. This is illustrated in the figure as *Routing Profile*, *Forwarding Profile*, *QoS Profile* and *Security Profile*, etc. On the other hand, each abstracted functionality profile can be seen to be horizontally composed of *Routing Policies*, *Routing Objectives* and *Routing Configurations Data*. This is illustrated in Figure 54.

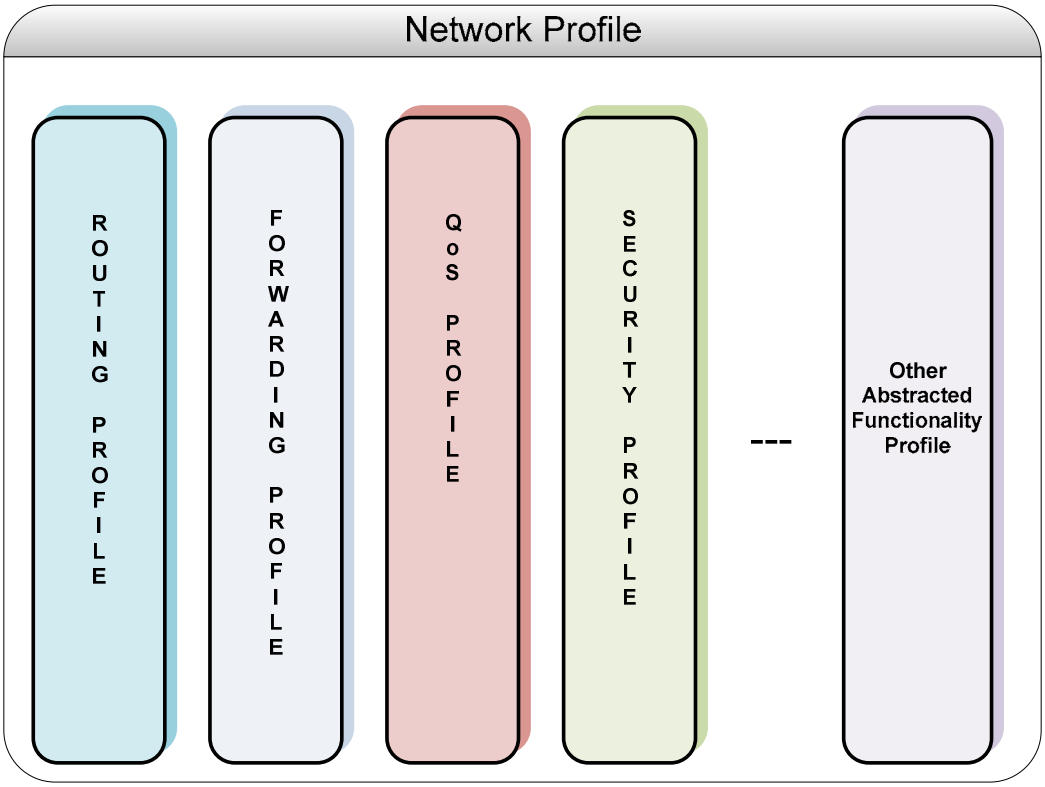


Figure 53: Horizontal Decomposition of the Network Profile

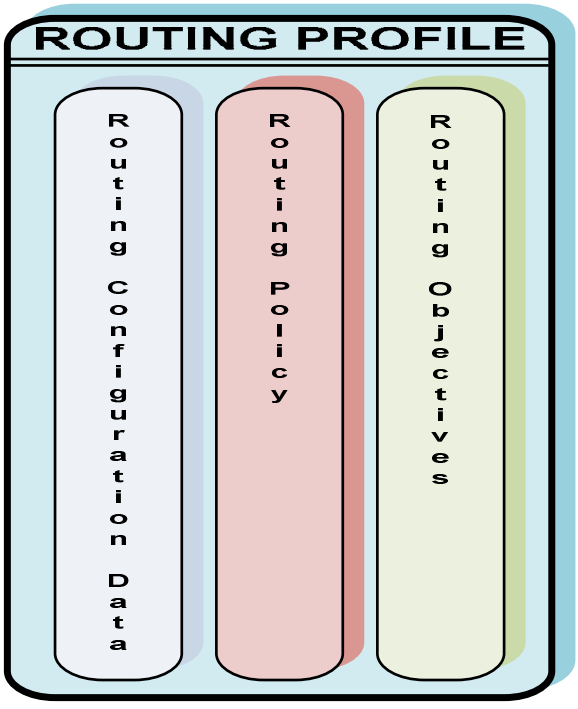


Figure 54: Horizontal Decomposition of the Routing Profile

10.1.1.3 Relation between Vertical and Horizontal Decomposition

The Vertical Decomposition allows the operator to design policies, objectives and provide configurations as per the roles the various nodes are expected to play in the network. The Horizontal Decomposition allows the operator to view the same NETPROF along the various abstracted networking functionalities. Thus the relation between the Vertical and Horizontal Decomposition is pretty straight forward. This is expressed in Figure 52 and Figure 54.

10.1.2 GANA Network Profile Files and their Relationships

The *GANA NETPROF* was designed for the following purposes:

1) To provide a detailed, structured and monolithic framework for specifying the policies, objectives and configuration data for a network and its nodes.

2) To provide a flexible framework for (re)configuring nodes, based on the dynamic roles they are computed to play in the network.

3) To standardize a common data structure for the policies, objectives semantics and configuration data for the network, and finally.

4) To separate between a node's role and functionality from its vendor specific configurations data, i.e. to keep the roles agnostic to the vendor specific configuration demands. In other words, to provide a framework that allows a vendor-specific-free implementation of the GANA's DEs.

The following requirements are achieved by providing a framework wherein the vendor specific details are separated from the policies, objectives and DE specific configurations. Thus the vendor specific details are separated from the *NETPROF*. The *NETPROF* is designed to accept all types of vendor specific configurations for a device/node. This is achieved by providing hooks for protocols and vendor specific node configurations, as illustrated in Figure 51 and Figure 52. This approach would provide the following key advantages to the operators and vendors:

1) The network operator can use existing configuration files without any further changes, thus preventing configuration errors caused by the transition to a GANA conformant network.

2) The network operators do not have the additional task of manually copying every configuration parameter currently configured through some script into the *NETPROF*. The network operator continues to maintain his/her configuration files that are vendor specific for the nodes/devices used in their networks.

3) Finally, by having these so called "hooks", a node's role is not confined to any specific vendor at design time, allowing dynamic role switching and re-configurations of the nodes.

While some configuration parameters are static; whose values are not manipulated by the DEs, the configuration values for the many parameters need to be adjusted at runtime in a dynamic manner to reflect the goals and objectives of the network. For instance, the value for a parameter such as ***Area ID*** is dynamic, as the network gets partitioned and repartitioned due to failures, occurrence of new nodes and changing network conditions. The problem arises, when the configuration parameters such as ***Area ID*** are expressed in different vendor specific semantics for each device. For instance, in Quagga, ***Area ID*** is expressed as simply ***Area.*** In order to avoid any vendor specific implementations of the DE, a simple solution that provides a mapping between parameter semantics used in DEs and their corresponding vendor specific semantics is needed. For this purpose, the *GANA Configurations Options Map* (MAP) is provided.

The MAP is a simple tabular structure (table 3) that maps configuration parameters used in GANA in its standard form (taken from RFCs) to vendor specific formats. The MAP is not just restricted to the semantics of the parameter, but rather expands to the expression of the values of the configuration parameters as well. An example of such a map is given in table 3.

Table 3: GANA Configurations options MAP

GANA Standard	Vendor 1 Specific		Vendor 2 Specific		Vendor *n* Specific	
Parameter Name as per the RFC	Parameter Name	Value Format	Parameter Name	Value Format	Parameter Name	Value Format
Router ID	*Router_ID*	<IPv4 address format>	*RouterID*	<Hash code format>	*Router-ID*	<IPv4 address format>
Area ID	*Area*	<IPv4 address format>	*Area_ID*	<IPv4 address format>	*AreaID*	<IPv4 address format>
…	…	…	…	…	…	…
Parameter N	*P_N*	<Vendor 1 specific format>	*Param_N*	<Vendor 2 specific format>	*Parameter N*	<Vendor n specific format>

The MAP can be expanded with new columns that represent new vendor and their vendor specific parameter naming and value semantics. It should be noted here that while some of the vendor specific parameter values can be directly computed from the standard value, the values of others may not be a straightforward conversion/translation. For this purpose, GANA allows the use of Converters (e.g. a Web-Service description) which can be used to convert the standard value into the vendor-specific value and vice-versa.

When, a node boots up in the network, it aggregates its Capabilities, and self-advertises itself. A Network-Level DE (e.g. Network-Level-Routing-Management-DE) computes the type of role a node/device will play in the network based on the Capabilities of the node/device. Based on this, an appropriate "Node Role", i.e. a Node Profile is chosen from the NETPROF. However, the vendor specific block still remains empty. Based on the Capabilities, a corresponding vendor specific configuration options (CONFIG) is chosen. This CONFIG is then appended to the profile that was chosen for this node. This results in a Node Configuration Options (NODECONF) file. This NODECONF is then uploaded to ONIX which disseminates it to the appropriate node as depicted on Figure 55.

During the reconfiguration of a node, the current NODECONF of the node, located in the ONIX is used by the Network-Level DE as depicted on Figure 56. A filter is applied to this NODECONF, and an object that encapsulates the vendor specific configuration is retrieved (step 1). Using the MAP, the vendor specific configuration options object is converted to a GANA Standard Configuration Options object (step 2). The Network-Level DE can now modify the values of the configuration options in this standard form (step 3). The resulting configuration is now converted back to vendor specific format using the same MAP (step 4). Finally, in step 5, the object is marshalled and replaces the old vendor specific configurations part of the NODECONF. ONIX pushes the updated NODECONF to the corresponding node, and the reconfiguration process starts.

The GANA NETPROF is formalized through the well known industry de facto XML standard. This provides a formal and standardized approach to designing and engineering GANA NETPROFs. To summarize, GANA NETPROF thus consists of following types of files expressed in an XML format:

1) NETPROF - The Network Profile consisting of the Policies, Objectives and Configurations Data required for the DEs of the Nodes.

2) CONFIG - The Vendor Specific Configuration files holding all the configurations parameters and their default values for one type of vendor specific node.

3) MAP - The GANA Configurations Options Map.

4) NODECONF - The Node Configuration file providing node-specific Policies, Objectives and vendor specific Configurations Data to the Node.

The vendor specific configurations are saved in separate xml based files - "GANA_Configuration_Options_X_Y.CONFIG", where X represents the vendor and Y the version. The structure that all CONFIG files shall conform to is given by aXML schema.

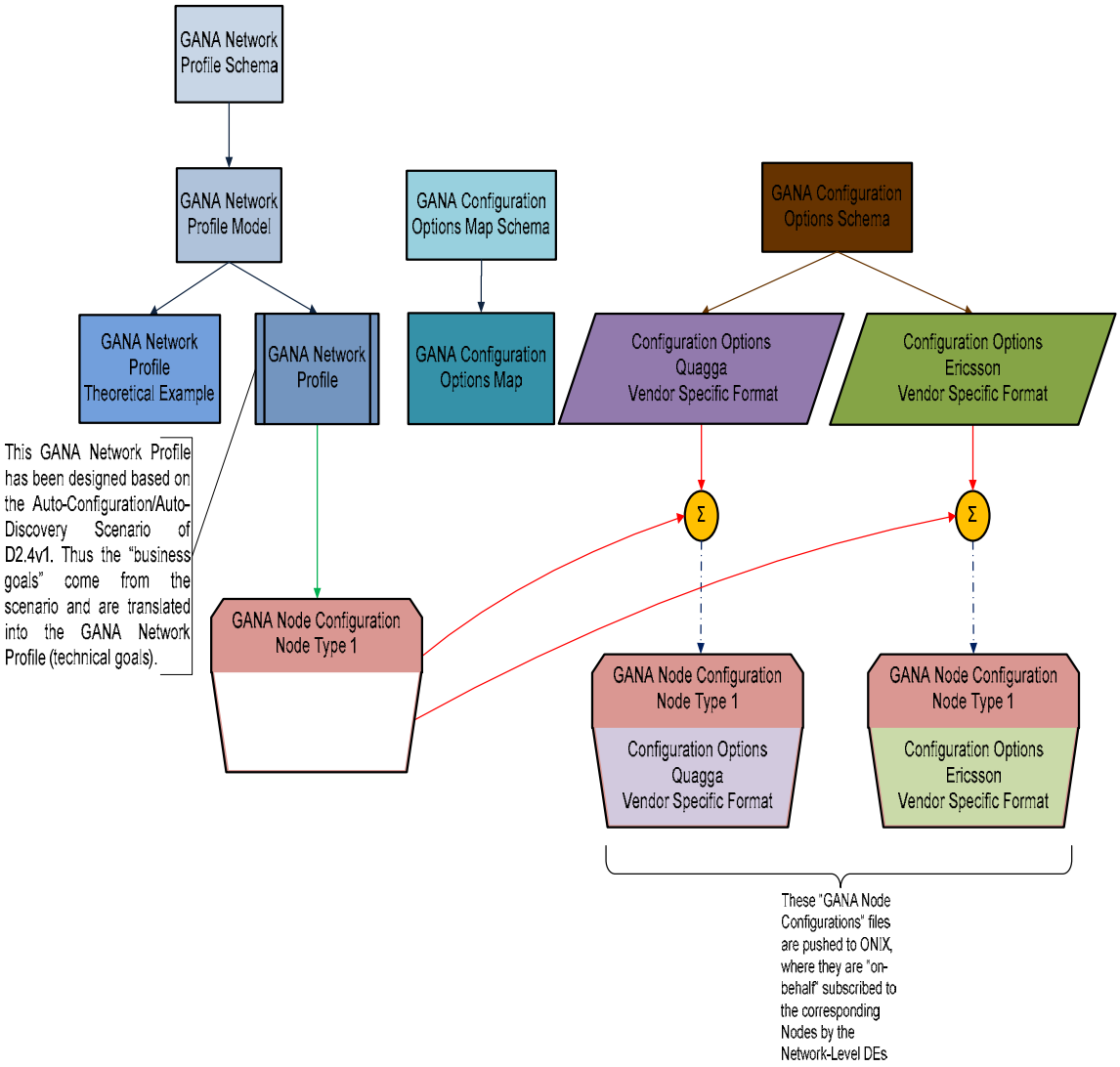


Figure 55: GANA Network Profile Files and their Relationships

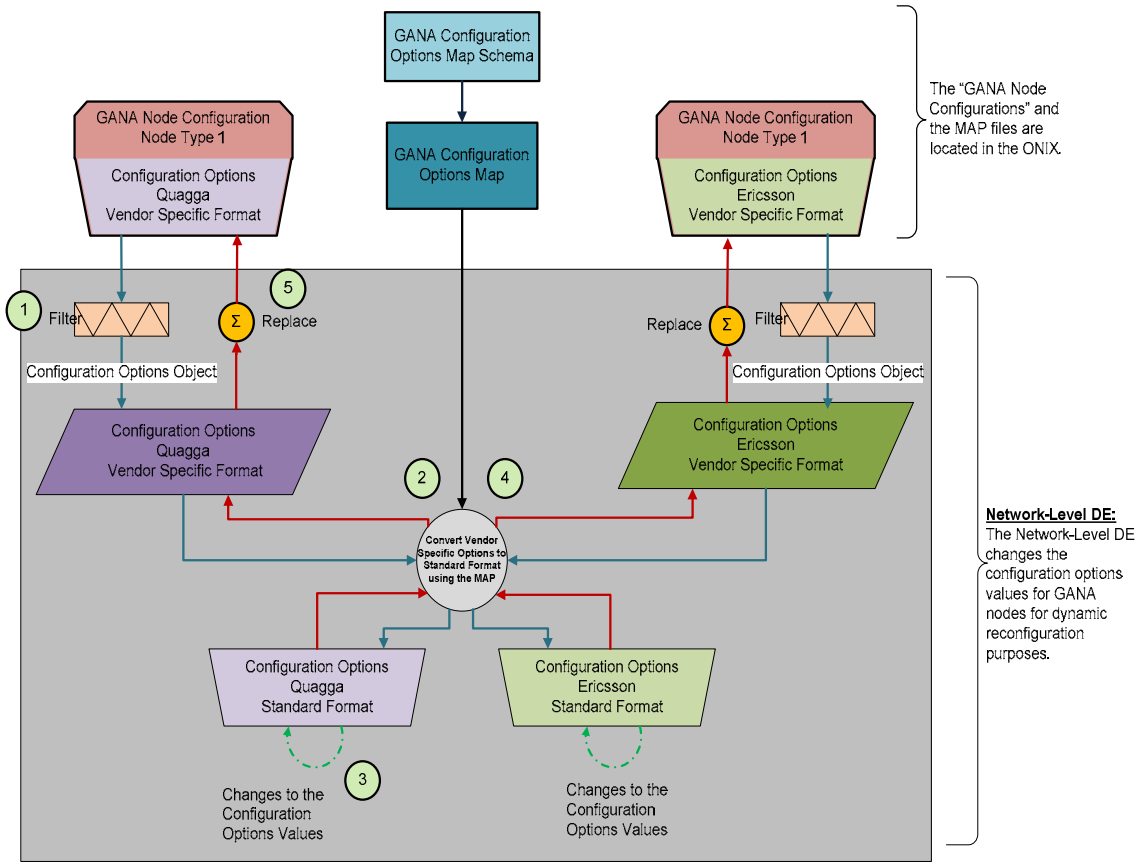


Figure 56: Dynamic reconfiguration of a node by the network-level DEs using vendor specific configuration options and "GANA Configuration options Map"

10.1.3 GANA Network Profiles and Policy creation, distribution and modification

Given the above described components, this clause describes how the creation, distribution and modification of policies is performed through GANA Network Profiles. In the context of GANA the entities, which handle and apply policies are the Decision Elements (DEs), thus as seen in the previous clause the Network Profile contains policies for all DEs on the different levels of the GANA hierarchy. The process of creation, distribution and modification of policies for Network-Level-DEs and of policies for DEs inside of nodes differs and are now described separately.

10.1.3.1 Network-Level Policies

Network Level policies, which determine the behaviour of the overall network, are addressed to the Network-Level-DEs. Because Network-Level-DEs are not configured through GANA Node Configuration Files, the distribution of policies to Network-Level-DE can only be done by setting the policies in the *NETPROF*. The policies for the Network-Level-DE are created (or generated) by the network operator/administrator according to his overall network goals. Each Network-Level-DE receives a Network Profile (horizontally decomposed according to its functionality) and can apply the policies on the network level. If a modification of policies for Network-Level-DEs is needed, which can be done by the operator or by other Network-Level-DEs, the *NETPROF* is adapted to the changes. Equally to the policy creation process, after the modification of the policy the Network-Level-DEs will receive the new version of the Network Profile that contains the modified policies and will apply these. Since the policies in the Network Profile represent the goals of the network, they should only change, to influence the behaviour of the overall Network or of a group of nodes performing the same role. Thus policies set inside the Network Profile are considered as ***long-term policies***.

10.1.3.2 Node- and Function-Level Policies

There are two ways to set and modify policies for DEs on the Node- or Function-Level. First the process of the distribution of ***long-term policies*** as described for Network-Level-DEs can be also applied to DEs on the other levels of the GANA hierarchy. The policies for these DEs can also be set or modified inside of the Network Profile and are then distributed to node inside of the GANA Node Configuration Files (as described in clause 10.1.2). The Node-Main-DE distributes the policies from the GANA Node Configuration File to the underlying DEs. Still it has to be mentioned that, due to the fact that the Network Profile contains Node-Profiles for node roles and not for individual nodes, policies that are set for a Decision Element inside of a certain Node-Profile, will affect the corresponding DEs an all nodes performing this role. Thus long-term policies are expressing the behaviour of DEs for device roles and not individual devices.

In the context of GANA Network Profiles the configuration of individual nodes is performed not directly through a Network Profile, but though GANA Node Configuration Files (*NODECONF*), which contain a part on the Network Profile needed by the node and vendor specific configuration options. Setting policies directly in the GANA Node Configuration Files is the second possibility of policy creation/modification. Because the creation of the Node Configuration Files is performed by Network-Level-DE, these can influence the behaviour of the underlying DE in the nodes by setting or modifying policies inside of the GANA Node Configuration Files. Obviously these policies will affect only the behaviour of the DEs inside the individual node, which is getting configured through the configuration file. Contrary to policies which are set inside the Network Profile, policies set in the GANA Node Configuration Files affect only a certain node. Thus they can be modified often and can be seen as ***dynamic policies***.

10.1.3.3 Routing Policy Example

To exemplify the processes described in the previous clause the distribution and modification of a routing policy to control the routing behaviour of the network is described in this clause.

The GANA Network Profile does not enforce the usage of a certain policy language in the Network Profile, because this is specific to DE implementation issues; thus any policy specification language can be used. For these examples the standardized **Routing Policy Specification Language next generation (RPSLng) [**i.11**]** is used to express policies for the Function-Level-Routing-Management-DE. Policies defined used the RPSLng can express the handling of inter-domain routes by an Autonomous System (AS) (e.g. set/filter imported routes, set/filter exported routes, define route aggregation, set route/protocol attributes…).

The RPSLng policy shown in Figure 57 implies the following behaviour of an AS:

1) Import routes from trusted AS from BGP into OSPF.

2) Do not export routes to malicious prefixes/AS.

3) Do not export IPv6 multicast.

```
as-set: AS-TRUSTED
members: AS2, AS4

mp-filter: MALICIOUS
members: AS3,{fd00:1234::/64}

as-num: AS1
mp-import: BGP4 into OSPF
        afi ipv6
        from AS-TRUSTED accept ANY

mp-export: to AS-TRUSTED annpunce NO MALICIOUS
        except { afi ipv6.multicast}
```

Figure 57: RPSLng Policy Example

The sets "AS-TRUSTED" and "MALICIOUS" inside of the policy can then be used to dynamically adapt the handling of routes to the current knowledge state.

In the context of the GANA Network Profiles the Network-Level-Routing-Management-DE can set this policy in the Routing-Profile inside of the Node-Profile for the role of Autonomous-System-Border-Router (ASBR) in the Sub-Profile for the Function-Level-Routing-Management-DE. Each ASBR will receive this policy inside of the GANA Node Configuration File created by the Network-Level-Routing-Management-DE. Figure 58 shows how this policy can be embedded inside the GANA Node Configuration File (NODECONF).

```
<Network_Profile>
  <Gana_Version>0.1</Gana_Version>
  <Description/>
  <Net_Level_Sub_Profiles>... </Net_Level_Sub_Profiles>
    <Node role-id="3">
      <Role_Description>Autonomous Systen Border Router(ASBR)</Role_Description>
      <Node_Main_DE_Profile>… </Node_Main_DE_Profile>
      <Func_Level_Sub_Profiles>
        <Standard_Functions>
          <Function type="Routing">
            <Standard_DE_Profiles>
              <DE type="RM_DE">
                <Version value="1">
                  <Description/>
                  <Policies>
                     <Policy id="1">
                       <as-set><name>AS-TRUSTED</name>
                         <members> AS2, AS4</members>
                       </as-set>
                       <mp-filter>
                         <name>MALICIOUS</name>
                         <members> AS3,{fd00:1234::/64} </members>
                       </mp-filter>
                       <as-num>
                         <name>AS1</name>
                         <mp-import> BGP4 into OSPF afi ipv6 from AS   TRUSTED accept ANY</mp-import>
                         <mp-export>to AS-TRUSTED annpunce NO   MALICIOUS except { afi ipv6.multicast}</mp-export>
                       </as-num>
                     </Policy>
                  <Configuration/>
                  <Objectives/>
                </Version>
              </DE>
            </Standard_DE_Profiles>
          </Function>
        </Standard_Functions>
      </Func_Level_Sub_Profiles>
      <Protocol_Level_Sub_Profiles> … </Protocol_Level_Sub_Profiles>
    </Node>
  </Node_Profiles>
</Network_Profile>
```

Figure 58: Policy embedded in Routing-Profile

The Function-Level-Routing-Management-DE can then enforce this policy through adaption of the configuration of the routing protocols.

The Network-Level-Routing-Management-DE can modify this policy according to its knowledge about the sets "AS-TRUSTED" and "MALICIOUS". For example if it has established a *Trust Relationship* with a Routing-Management-DE in an peering AS it can add the number this AS to the set "AS-TRUSTED"; or if the Network-Level-Security-DE discovers that a routing through a certain AS does not function properly (e.g. AS routing black hole) it can notify the Network-Level-Routing-Management-DE, which on its part can add this AS into the "MALICIOUS"-set in the policy to pervert adverting routes through it. After the policy is updated, the Function-Level-Routing-Management-DE will receive the update inside of a updated *NODECONF* and reconfigure the protocols according to the new values in both sets.

10.2 Capabilities Self-Description and Self-Advertisement

10.2.1 General Considerations

Each DE or ME of a node/device should be able to self-describe its capabilities. As a node/device boots up in the network, its NODE_MAIN_DE initializes itself and performs auto-configuration of its secure Neighbour Discovery (ND) related parameters and secure self-advertisement of its Capabilities to on-link Neighbours. It then tries to discover the ONIX system located in the network. Once the ONIX is discovered by the node i.e. by the NODE_MAIN_DE, it tries to get authenticated by the ONIX, and advertises its point of attachment to the network, i.e. the address information concerning its interfaces, and Capabilities to the ONIX. The NODE_MAIN_DE creates a Capability Description of the node/device by triggering the process that is described in Figure 59 for collecting the capabilities of individual DEs and MEs of the node/device and presenting the overall capabilities of the node/device to the network.

NOTE: The behavioural model can be applied to any type of a network (not just an IPv6 network used here only for illustration purposes). Even in an IPv6 network different protocols/mechanisms than ND or the DHCPv6++ can also be used to exchange information. For more information on DHCPv6++ (some proposed extensions to the protocol, not yet standardized though, maybe in the future), refer to EFIPSANS deliverables D2.3, D2.4 and D2.6 [i.5].

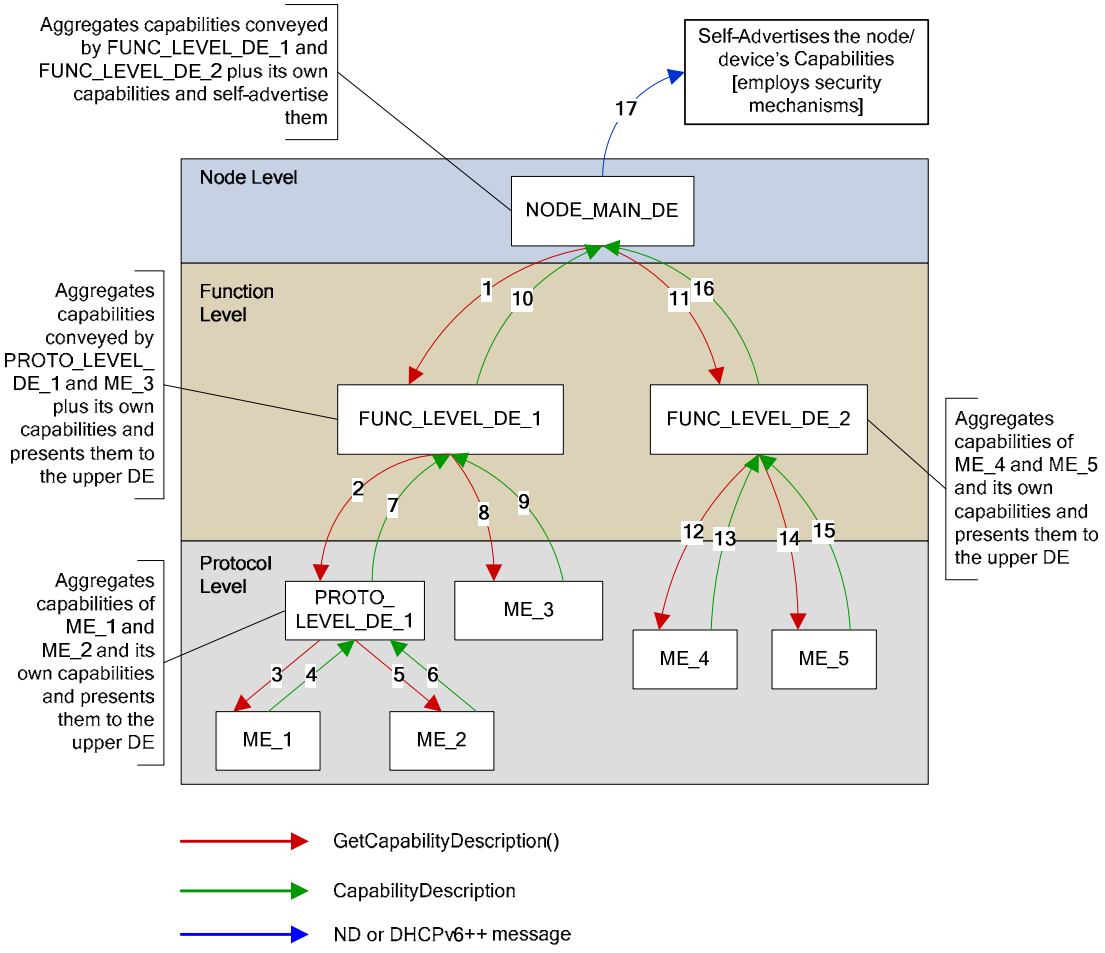


Figure 59: Illustration of Self-Description and Self-Advertisement of Capabilities of a Node/Device in an IPv6 Network

The capabilities composition process is an iterative process. It starts with the *NODE_MAIN_DE* asking for the capabilities description of the lower level DEs. These DEs will also ask their Managed Entities (MEs) to provide the capabilities description and the process continues down to the MEs at GANA's lowest layer. For asking the capabilities description of the Managed Entities (MEs) the DEs use the **GetCapabilityDescription()** primitive provided by the **General_NonSensoryInformationRetrival_Interface** of the Management Interface of an ME (refer to GANA models of an ME presented earlier for more information).

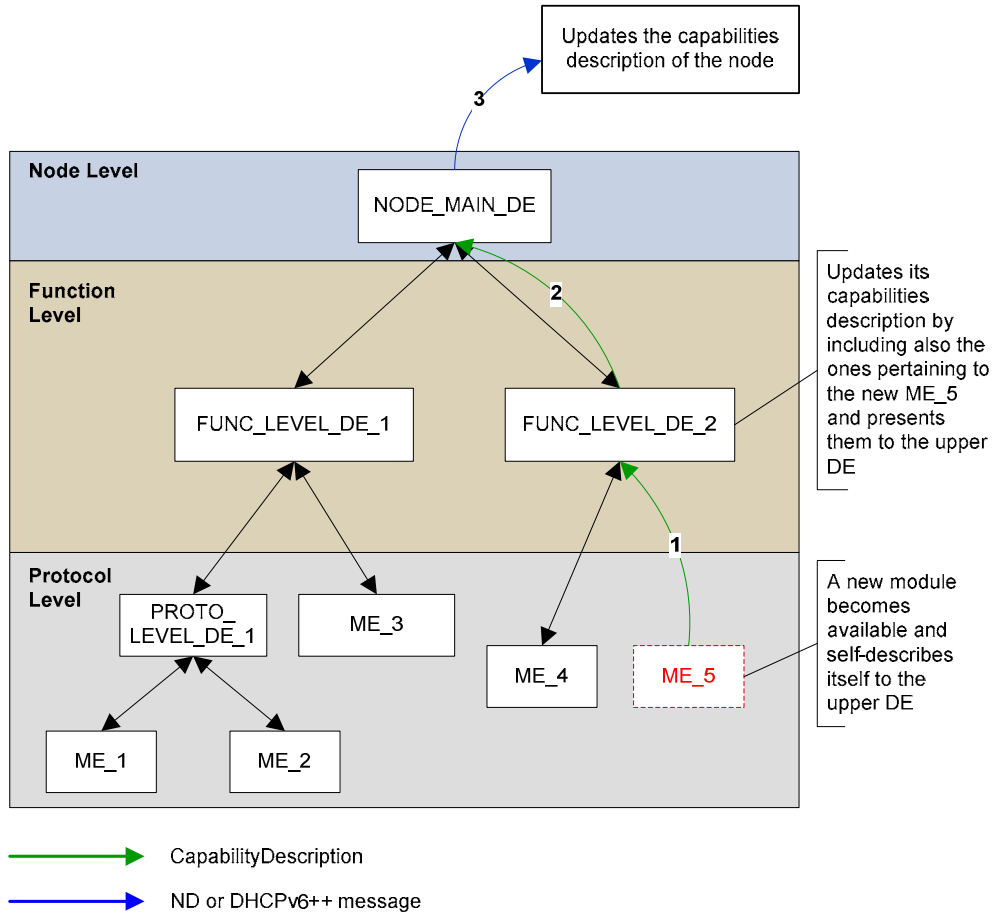


Figure 60: Updating the Capabilities Description when a new module becomes available

As highlighted in Figure 59, the *NODE_MAIN_DE* asks for the capabilities of the *FUNC_LEVEL_DE_1* (message 1). The *FUNC_LEVEL_DE_1* in turn, asks for the capabilities of the *PROTO_LEVEL_DE_1* (message 2) and *ME_3* (message 8). The *PROTO_LEVEL_DE_1* will ask for the capabilities of its Managed Entities, *ME_1* and *ME_2*. After receiving the capabilities description of the Managed Entities *ME_1* and *ME_2* (messages 4 and 6), the *PROTO_LEVEL_DE_1* aggregates them and responds to the *FUNC_LEVEL_DE_1* request for capabilities description by offering the aggregate composite view of its own capabilities and the capabilities of *ME_1* and *ME_2* (message 7).

In a similar way, the *FUNC_LEVEL_DE_1* aggregates the capabilities received from *PROTO_LEVEL_DE_1* and from *ME_3* and then present the aggregated composite capabilities to the *NODE_MAIN_DE*. In this manner, each DE has an overall view of the capabilities of the MEs under its control (direct or indirect). After receiving the capabilities description from the *FUNC_LEVEL_DE_1* and *FUNC_LEVEL_DE_2* (message 10 and 16), the *NODE_MAIN_DE* aggregates the Capability descriptions and then self-advertises (publishes) the overall capabilities of the node to the ONIX framework or/and will self-advertise its capabilities to its on-link neighbours or to different network level DEs.

The Capabilities of a node can change during its operation and in Figure 60 is presented the process that takes place when a new module becomes available, either by virtue of being instantiated into the node/device or being orchestrated/launched by the DE responsible of the ME (the new module).

Each module should be able to self-describe its capabilities. The ME_5 pushes its capabilities description to the upper level DE that manages it (message 1). *FUNC_LEVEL_DE_2* then updates its capabilities description by including also the capabilities provided by *ME_5* to the aggregated capabilities description and then pushes the composite capabilities description to the *NODE_MAIN_DE* (message 2). The *NODE_MAIN_DE* then updates the capabilities description of the overall node using for example DHCPv6++ messages (if the capabilities are to be stored in ONIX) or it will disseminate the updates to the on-link neighbours using the EFIPSANS proposed Extended Neighbour Discovery messages i.e. ND++ (message 3). However, the ONIX itself does not rely on IPv6. The IPv6 case is only provided for illustration purposes.

10.2.2 Capability Description Files and their Relationships

The Capability Description Model is formalized, through a definition of a extendable XML schema. The capabilities in this model are categorized in accordance to the description given in the previous clause. Thus there are capabilities belonging to the Node-Main DE and Function-Level DE. Each Function-Level DE holds the capabilities of the protocols they manage.

In addition to the above classification/grouping of capabilities, a new classification called "Node Attributes" is also included. This is for representing those Capabilities that are common for the entire node and cannot be associated with any single DE or ME. These include node attributes such as CPU properties, memory properties, number of physical interfaces, etc. The node attributes has to be computed and populated by the NODE_MAIN_DE.

Further the NODE_MAIN_DE has to set the roles the node is currently playing in the network and a list of possible role, which the node is able to perform. The role information is important for Network-Level-DEs, because they are deciding, which roles a node has to perform.

Due to the heterogeneous device capabilities, the model contains only generic capabilities common for GANA conformant networks this model needs to be expanded and extended as and when new capabilities requirements arise. The schema ensures that the extended model is backward compatible with previous DE versions that interpret the model to understand the capabilities captured in it.

10.3 Decision Notification in GANA for the "Human in the Loop" towards building Trust and Confidence

1) At the early stages of the operator building trust and confidence in autonomic networks, certain types of decisions should be communicated to the administrator by the autonomic network as "tentative decisions" that are not yet executed and can be executed if the operator confirms. Potentially, a decision made at the "node-level" in GANA or at the "network-level" by Network-Level-DEs are candidate for "decision notification" to the human so that the human can close the loop by confirming to the DEs that they can go ahead and execute the decision. There are some types of decisions, though, that should be handled by the autonomic entities (DEs) without the need to notify the human of the decision being made for execution.

2) **(a)** When the administrator informs the network that for certain types of Decisions (the human would specify them using some means e.g. a Rule or Policy Specification Language) he/she wants a Decision Notification before the DE executes the decision, the DEs should notify the administrator so that the human closes the loop by providing a response to the Decision Notification. This may happen during the early days/weeks/months of operating the autonomic network till the time when the administrator has build trust and confidence. **(b)** When the administrator deactivates Decision Notification then the DEs shall proceed with executing Decisions without issuing Decision Notification to the Human. **(c)** When Decision Notification is deactivated as in **(b),** the DEs shall however inform the human to facilitate for any offline analysis (possibly by simply logging the Decision(s) that were taken in response to "a triggering event", and the logging should include both the "Decision(s)" and associated "Event(s)").

10.4 Autonomic Services Management in a GANA compliant network

10.4.1 Service provision scheme and its evolution

The continuous increase of the number of user equipments, in combination with the evolution of the traditional client/server model for service provision, towards more distributed application structures, have increased the complexity for service management and paved the way for more advanced services. Especially, by taking into account that even simple users through their own devices (e.g. smart phones) can concurrently have the role of service consumer and service provider.

Service instantiates a specific functionality offered to a user by the execution of one or more applications. An application is considered the "software components providing services to users by utilising service capability features" [i.6], whilst a service [i.7] is the "user experience provided by one or more applications or an aggregation of a number of service capability features". There are various types of services that could be considered in a future Internet environment: conversational services (e.g. voice, multimedia telephony), fixed or mobile converged services, group communication services, integrated services (e.g. a mixture of telephony and messaging services), media streaming applications, non-real time, interactive applications (e.g. Web browsing, chat), real-time, interactive applications (e.g. real-time gaming applications), ubiquitous services (e.g. associations with huge number of sensors, RF tags, etc.).

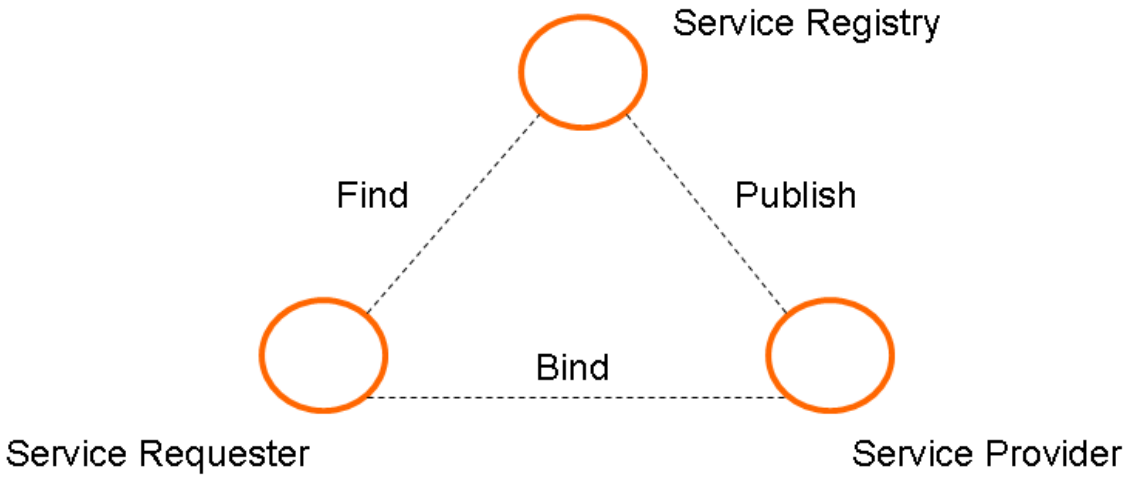


Figure 61: Traditional Service Provision Scheme

The role of the Service-Management-DE in the context of the GANA architecture is twofold. The Service-Management-DE undertakes to control those tasks that are responsible for the service provision and specifically:

a) service discovery;

b) service delivery;

c) service adaptation; and

d) service publish.

Furthermore, the service management DE has the capability to cooperate with other network management DEs of the GANA architecture (e.g. NODE-MAIN-DE and the rest of the other Function-Level-DEs other than itself) for network management purposes. The Service-Management-DE acting on behalf of the services/applications, should provide to the NODE-MAIN-DE and the rest of the other Function-Level-DEs other than itself, a description of "survivability requirements" of the services/applications it orchestrated/provisioned in a GANA node. A "survivability requirement" is the delta-time within which an adverse event such as a failure should be detected by some entity and notified to an event-notification consumer who then employs some fault-tolerant mechanisms upon the reception of the event-notification. Some of the fault-tolerant mechanisms may be implemented by the services/applications and some adaptive mechanisms may be employed by the Service-Management-DE. Since the Service-Management-DE realizes a Control-Loop over the Services/Applications Layer within the node, the DE should listen for those events from the applications/services that require the DE to be the one to invoke self-adaptation mechanisms for the health of the running service(s)/application(s). Views from the other Function-Level-DEs should be shared by all the DEs on this level, including the Service-Management-DE, so that the Service-Management-DE can invoke its own adaptive strategies for the whole Service Layer of the node.

10.4.2 Service Management

The key modules that exist in a complete service provisioning framework are depicted in Figure 61. Initially, a Service Provider, which essentially comprises an actor, internal or external to the actual system, provides services to terminals/user equipments of one or more networks. The provision requires the initial registration of the service. The action of service registration essentially corresponds to the notion of Service Publishing which captures all functionality related to service registration and advertisement. Hence, a service can become accessible and consumable by any Service Requester. The latter, discovers a Service by utilizing functionality identified by the concept of Service Discovery. Finally the service is delivered to the requestor through predefined protocols (Service Delivery) and can be adapted in order to match the requester's requirements (Service Adaptation).

The Service Discovery capability aggregates all functionality related to the discovery of a specific Service or a set of services by a registered client. The implementation of this action is directly influenced by the structure of the communication scheme. Therefore, two cases are distinguished, specifically centralized (traditional client server communication) and ad hoc (peer to peer scheme).

The Service Publish includes all functionality related to the registration and possible advertisement of a service by a service provider. This module implements the following:

- SW Component Profile registration.
- SW Component upload.
- SW Component Binding Rules registration (Application Registration).
- Application Binding Rules registration (Service Registration).
- Service Profile registration.

Services may also be published and discovered through the ONIX system of the autonomic network.

The Service Delivery module aggregates all functionality related to the delivery of a specific service or a set of services to a registered client. The term delivery embraces all procedures involved from the selection of a service (right after the discovery phase) until its consumption by the client. The implementation of this action, as in the case of the Service Discovery is directly influenced by the structure of the communication scheme.

Finally, the service adaptation module includes all those schemes that are used in order to differentiate the result that an end-user experiences. The feature of adaptability is an important characteristic of pervasive services. An important phase of service adaptation procedure is the transition from the current state to the most suitable one of the entity being adapted, considering the specified policies of the involved entities and the contextual environment. Service adaptation could be executed by various actions, such as:

- Change of the infrastructure/environment in which the application is executed.
- Change of the content parameters consumed by the involved applications (server, client).
- Change of the applications interactions that compose a service.
- Change of the SW components that define one or more applications.

10.4.3 Service Management and Network Management Levels Cooperation

Existing network management systems have limited capabilities for their cooperation with service management systems e.g. IMS. Another important aspect that should be taken into consideration is the cooperation between the network management systems (whose role shall be fulfilled by Network-Level-DEs) and service management mechanisms. The last decade there is a lot of literature and research work for the automation of communication systems network management task by reducing the human intervention and handling complex situations. Self-management of future Internet network infrastructures necessitates the introduction of decision making techniques and the capitalization of the existing knowledge and policy frameworks, as it is depicted in Figure 62, where the cognitive cycle is present for the automation of network management tasks.

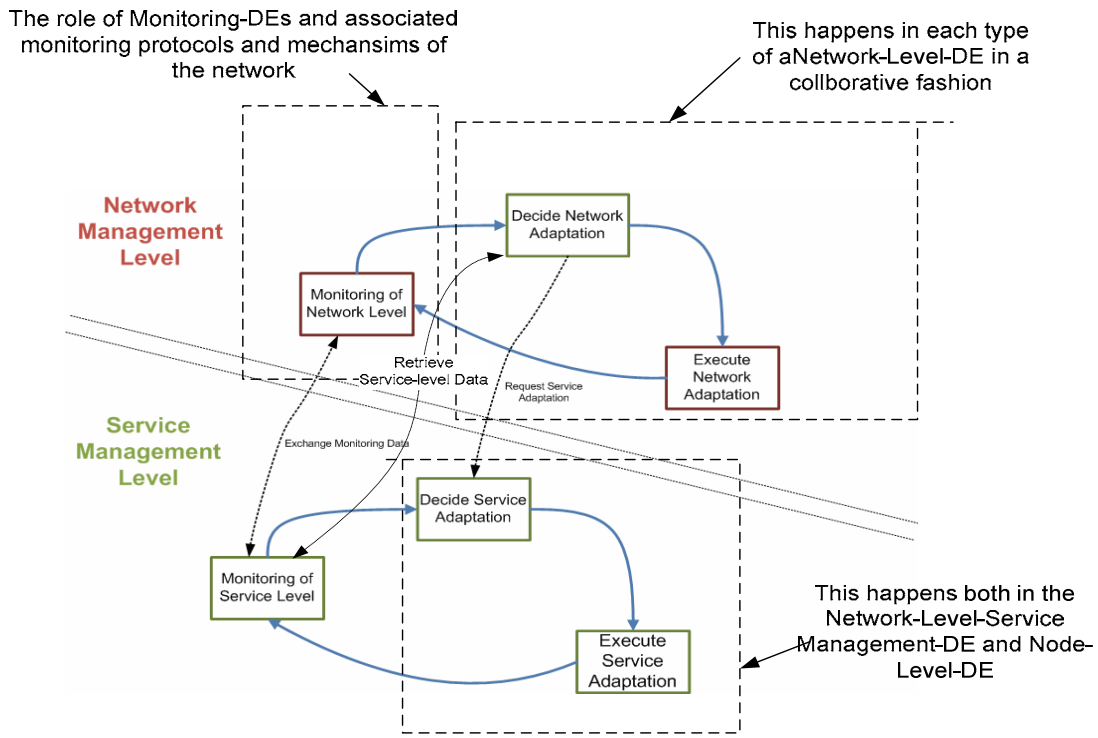


Figure 62: Service and Network Management Systems (i.e. Network-Level-DEs) interaction

For an efficient and scalable network management, where various stakeholders participate (network operators, service providers, end users), a distributed approach is required. Dynamic network (re)-configuration in many cases is based on cooperative decision of various future Internet Elements and distributed network management service components. Hints and requests/recommendations are exchanged among the layers, in order to indicate a new situation or an action for execution. The automated and dynamic incorporation of various layers/levels requirements (e.g. SLAs) into the management aspects provides also novel features to network management capabilities. Moreover, the resolution of conflicting requests is an issue of situation awareness and elements' domain policy prioritization. Three main communication channels () are necessary between the network management and service management systems:

a) The first one is used for the exchange of monitoring data that is sensed locally either at the service or at network management level. The network management system (i.e. Network-Level-DEs) could use service-level data (e.g. service type, codec, data rate etc) as an input for the situation deduction and specifically for the identification of possible faults or optimization opportunities. Furthermore, the service-level parameters could be used by the objective functions that the network management system (i.e. Network-Level-DEs) should solve for the decision taking phase, according to the identified fault. On the other hand, the service management level could exploit network-level monitoring data e.g. for the selection of the most efficient application server taking into account network conditions or for the building of the service path.

b) The second communication channel is used by the network management system (i.e. Network-Level-DEs) in order to trigger service adaptation actions. Each network management system (i.e. a Network-Level-DEs, and in particular the Fault-Management-DE) has a list of configuration actions that could be triggered as a remedy to a detected fault. Hence, one more configuration that could be triggered by the network management system is the service adaptation e.g. service re-composition.

c) The third communication channel is between the Network-level-DEs responsible for network management, which communicates with Network-Level-Service-Management-DEs in order to access service-level data. Network-Level-Service-Management-DEs retrieve data from various Node-Level-Service-Management-DEs.

11 Mapping and Impact of GANA on today's Management Paradigms

This clause provides a mapping of GANA to today's Management Paradigms and Frameworks, as well as GANA mappings and fusion with systems such as Operations Support Systems (OSS's), while painting a picture on what autonomics and self-management would imply in terms of integration and evolutions of Network Management Systems (NMSs), OSS's, etc.

In the process of mapping and superimposing GANA into the system and network architectures as well as the fusion of GANA and today's management paradigms and frameworks, it also useful to show a reflection of the implications or impacts on today's functional planes and layers of the network. For example, there may be an implication that functional planes such as the Control-Plane and Management-Plane become merged or need to be re-factored.

11.1 GANA Decision Plane and today's Management Plane

In GANA, it is envisaged that today's traditional Management Plane becomes part of what is called the GANA Decision Plane. The GANA Decision Plane: makes all decisions driving a nodes' behaviour (including the behaviour of all Managed Entities of the node) and network-wide control, including reachability, load balancing, access control, security, and interface configuration. Replacing today's Management Plane, the Decision Plane operates in real time on a network-wide view of the topology, the traffic, events, context and context changes, network objectives/goals/policies, and the capabilities and resource limitations of the nodes and devices of a network of some scope (Definition adopted but with modification from the 4D architecture [i.25]). All DEs and their Vertical and Horizontal Interfaces (as described in clauses 10.1.1 and 10.1.1.2) with other DEs make up the Decision Plane of the network.

The components, protocols and mechanisms of the traditional Management Plane also need to be autonomically managed. In a GANA Node, the components, associated protocols and mechanisms of the management plane such as Agents (associated with what is called the Manager-Agent Model), as well as and mechanisms to enforce policies on such Managed Entities (MEs) may need to be handled by the Node-Main-DE. On the other hand, over time, in the network evolution path, Management Applications will be invoked solely by Network-Level-DEs as described later in the clause 11.7 on GANA and Network Management Systems (NMS's).

11.2 GANA and today's Control Plane

The Control-Plane, considered a sub-plane of the GANA Dissemination Plane, is considered by the networking community as an extension of the Management Plane. The Control Plane itself needs to be autonomically managed and controlled by specific DEs. At the FUNCTION-LEVEL in GANA, there ought to be a "Control-Plane-Management-DE" that autonomically manages control plane protocols and mechanisms such as routing protocols. Due to the need for "further specializations" of the Control-Plane, there is a special type of such a DE, called the Function-Level Routing-Management-DE that needs to work together with a counterpart on the Network-Level (Network-Level-Routing-Management-DE). This means other types of specialized control-plane managed related DEs need to be introduced in a similar fashion for autonomic management of control other types of control protocols and mechanisms such as GMPLS, etc.

11.3 GANA and today's Data Plane

At the FUNCTION-LEVEL in GANA, there ought to be a Data-Plane-and-Forwarding-Management-DE that autonomically manages Data Plane protocols and mechanisms. In GANA the Data Plane - Consists of protocols and mechanisms that handle individual packets (extending up to the traditional layer 4 protocols such as TCP and UDP) based on the state that is output by the Decision Plane (i.e. the Data Plane Management-DE). This state includes the forwarding tables, packet filters, link-scheduling weights, and queue management parameters, as well as tunnels and network address translation mappings (Definition adopted but with modification from the 4D architecture [i.25]). Example elements of the Data Plane i.e. protocols or mechanisms belonging to this plane are: IP Forwarding, Layer 2.5-Fowarding, Layer 2-Fowarding, Layer 3-Switching, Layer 2.5 switching e.g. MPLS, Layer 2-Switching, and Physical Layer technologies. For more information on the types of Managed Entities (MEs) assigned to be autonomically managed and controlled by specific DEs, refer to the DE-ME Mappings Table provided in the present document (see clause 9.11.5).

11.4 GANA Mappings to the TMN Logical Layered Architecture (LLA)

The **Network Elements (NEs) Layer:** A network element is expected to support a certain type of management functionality. The GANA Decision Plane drills into the network element architecture and introduces Autonomic Manager components (Decision Elements) at the three levels of autonomicity/self-management within a so-called GANA Node, which were described earlier together with the fourth level in GANA (the network-level) (see clauses 9.2 and 10). Taking into account the evolution of networks towards self-managing networks, the most important GANA Levels for Self-Management within a node/device architecture would be the FUCNTION-LEVEL and the NODE-LEVEL since the FUNCTION-LEVEL introduces Decision-Elements (DEs) directly above the existing Protocol Stacks and Mechanisms for autonomic management and control of the diverse Protocols, Protocol Stacks and Mechanisms, each of which is assigned to specific DEs as depicted in the DE-ME Mappings Table provided in the present document (see clauses 9.11.5 and 9.11.6). The Model of a GANA Node is that of a "virtual node". It is a "virtual node structure" and is there to help produce "Specifications" and does not necessarily mean that implementations shall follow the GANA Node structure, since some of the Building Blocks (i.e. DEs) may be merged in an actual implementation, provided that the expected behaviours on standardized interfaces are guaranteed.

The **Element Management Layer (EML):** What this layer involves is the management of the individual devices in the network and keeping them in their proper operational state. Functions to view and change a network element's configuration, to monitor alarms/notifications from the devices and to trigger the devices to run self-tests are also part of this. GANA requires that element management be done both at the element level and at the network-level by the inter-working of the Network-Level-DEs and the Node-Level-DEs down to the protocol-level/mechanism-level DEs of an node/device. This is because, some degree of Self-Management should be performed by the node/element itself, following the rationale behind Hierarchical, Peering and Sibling Relations among DEs-responsible for autonomic management and control of functions, protocols and services. Up the GANA Decision Plane, to the network-level, Network-Level-DEs are still required to perform some degree of element management since certain types of network problems or decisions may be required rather at the network-level, resulting in Network-Level-DEs issuing commands to the element via its Node-Main-DE. So, we see that GANA distributes the functions of this management layer among the DEs in the Vertical view of the Decision Plane, following the benefits of having the levels of Self-Management introduces.

The **Network Management Layer (NML):** The layer makes use of the functionality provided on the Element Management Layer and involves the management of the relationships and dependencies between elements/devices for the goals of maintaining end-to-end connectivity of the network and ensuring that the network is fully and properly operating as a whole. Here again, the GANA DEs with their Hierarchical, Peering and Sibling Relations, across the node/device architectures and upwards to the Network-Level-DEs collaboratively perform Network Management. So, it can be seen that GANA distributes the functions of this layer among the DEs in the Vertical view of the GANA Decision Plane as well as in DEs in the Horizontal view of the GANA Decision Plane (realized by Sibling and Peering DEs across node/devices), following the benefits of having the levels of Self-Management introduced by the GANA Model.

The **Services Management Layer (SML):** This Layer involves the management of the services that the network provides, as well as ensuring that the provided services are running smoothly, with acceptable Quality of Service (QoS) and are functioning as expected. In order to realize the objectives, the functions of this layer rely on the functionality provided by the underlying network management layer. This kind of Service Management needs to be addressed within the Operations Support Systems (OSSs). This subject is covered in more detail in the clause on Network Governance in GANA compliant network (see clause 11). However, from another kind of Service Management, from GANA point of view, within a GANA Node, GANA introduces a view of the notion of a Service-Management-DE that is responsible of orchestrating services, service-discovery, interpretation of service and application requirements at run-time and requesting the network layer to behave in a service/application-aware manner. The Service-Management-DE, as a Decision Element, shall also realize a control-loop over the services/applications as its Managed Entities (MEs), and collaborate with other DEs responsible of autonomic management of the network layer protocols in order to realize and collaborative self-adaptation on both the service-layer and the network-layer). The Service Management DE has been set as a placeholder for Service Orchestration in the GANA Model. The Service Management DE triggers service discovery mechanisms to discover services meant for Application-oriented end-user through the network. This second aspect of service management i.e. autonomic services is addressed in the clause on Autonomic Services Management (see clause 11.4).

The **Business Management Layer (BML):** This layer involves management of the business processes and support functions, including billing, invoicing, helpdesk, etc. This kind of management is addressed within the Business Support Systems (BSSs). It is not clear at this stage, as to what aspects of GANA would play some role, apart from saying the Business Goals are supposed to be translated into so called GANA Profiles, which encapsulate Policies that are then applied at different Levels in GANA, a subject addressed as part of what we call Network Governance (see clause on Network Governance in GANA compliant network (see clause 10).

11.5 GANA-DEs vs TMN Architecture, and impact on EMS and NMS levels

Figure 63 illustrates the possibility of having Hierarchies of Network-Level-DEs.

At the NMS layer: GANA Network-Level-DEs shall be flexible enough to learn in real time when network configurations, software loads and capabilities have taken place and are able to assimilate and cope with these changes with zero or minimal management function down time.

At the EMS layer: The GANA Network-Level-DEs at this level are intelligent and cognisant Network Element Management systems that can for example exchange data with peer Network-Level-DEs (i.e. evolved EMS's) so as to be able to actively manage the border relationships between different network technologies to assure services and maintain QoS.

At the Network Element (Node/Device): For each Managed Entity (ME) or resource, management and decision making capabilities are realized by the Node-MAIN-DE, down to the Function-Level-DEs, down to Protocols-Level-DEs, which enable the reduction of the load on upper management layers and the data sent northbound to upper DEs, since some degree of self-management and control should be realized at GANA Levels of self-management within a node/device architecture.

EMS and NMS interfaces have to be standardised and that the Decision-Making process in both systems should be coordinated. The GANA DE-to-DE interfaces are meant to address this issue if one assumes that an ***evolved EMS is a set of Network-Level-DEs*** that implement diverse autonomic functionalities.

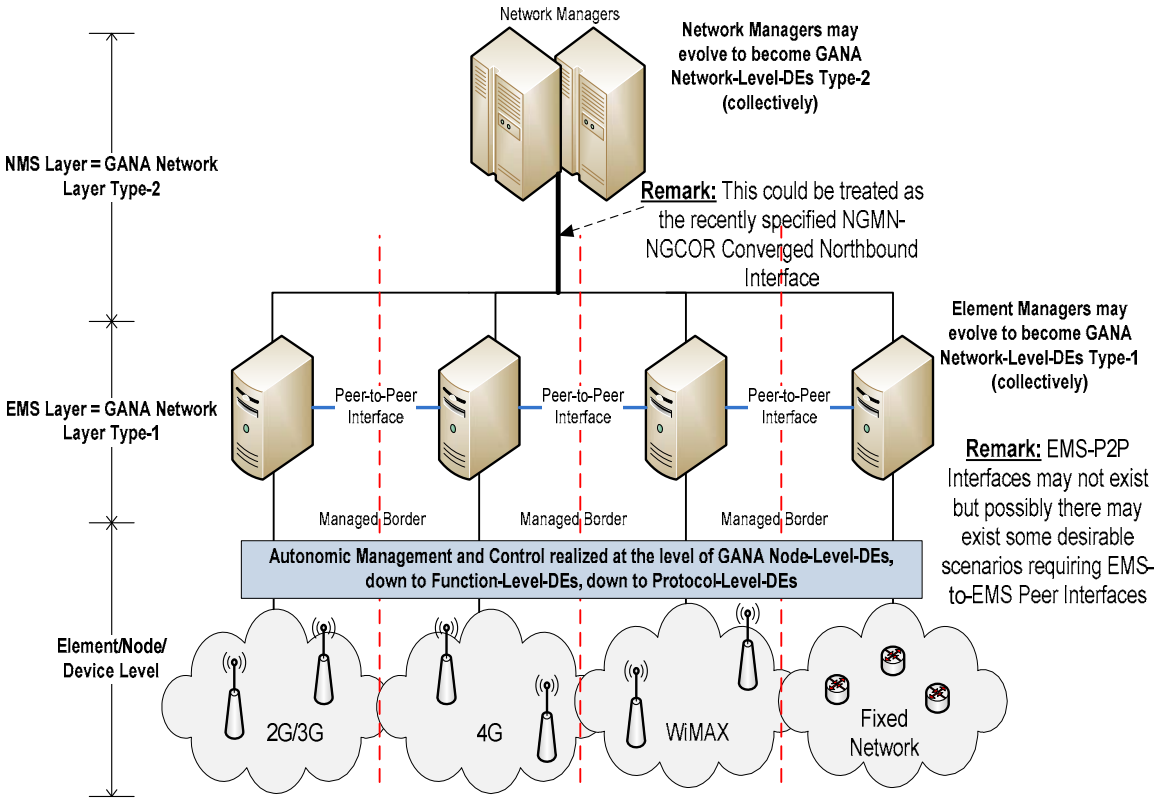


Figure 63: Hierarchies of Network-Level-DEs, and DE-to-DE Interactions Managing Border relationships between different network technologies to assure services and maintain QoS

The elements at the "EMS-layer" i.e. the Network-Level-DEs (Type-1) designed for autonomic management and control of a specific technology domain, may have multiple "Peer-to-Peer Interfaces" with counterparts responsible for the different technology domains, in order to realize a collaborative peer-to-peer distributed decision-making process that does not need to be realized through the logically centralized upper layer (i.e. NMS Layer i.e. Network-Level-DEs (Type-2)).

NOTE: The Network-Level Decision Elements, which realize the domains of the Knowledge Plane, can be designed to run various types of Self-Management Operations for the network and various types of Optimization Algorithms. Various implementation options can be considered for the Knowledge Plane. For example, the validated behaviours and algorithms of Network-Level Decision Elements may be used to evolve traditional EMS/NMS (OSSs) or, the DEs as well as the other elements of the Knowledge Plane may be implemented to run as standalone entities that interwork with traditional NMSs. Such an interworking can easily be implemented (as discussed in next clauses, as well as in Figure 64 and Figure 67). In general, the ways to implement the Knowledge Plane can still be abstracted from an architectural specification level.

11.6 The FCAPS Framework in the GANA architecture

The TMN Reference Model provides for a two-dimensional structured Framework for developing management systems, which includes the Management Layer Model (the TMN LLA) and five (5) Function Areas namely: Fault Management, Configuration Management, Accounting Management, Performance Management and Security Management. The GANA addresses the FCAPS areas by virtue of the DEs and their assignment to specific Managed Entities (MEs), and the supporting components and mechanisms that enable the realization of Self-Management within each of the FCAPS Function Areas. For example, the Fault-Management-DE, Resilience and Survivability DE of the GANA Decision Plane realize autonomic fault-management, resilience and survivability at each level of abstraction within the GANA Model. Other DEs include Auto-Configuration-DE for Auto-Configuration/Self-Configuration of node/devices and the network as a whole, Monitoring-DE for monitoring and accounting at the node/device level and network-level, Security-Management-DE at the node/device level and the network-level for autonomic security management e.g. self-defending/self-protection behaviours. On the other hand, autonomic Performance Management is meant to be handled by every type of a DE, with some DEs such as QoS-Management-DE and Monitoring-DE performing some key role in performance management. For more information on the types of Managed Entities (MEs) assigned to be autonomically managed and controlled by specific DEs, refer to the DE-ME Mappings Table provided in the present document (see clauses 9.11.5 and 9.11.6). In GANA, the FCAPS functions become diffused within node/device architectures, apart from implementing the Management-Plane of the overall network architecture.

11.7 GANA and Network Management Systems (NMS's)

Figure 64 illustrates the integration of today's Network Management Systems (NMS's) and GANA, and how the NMS's are impacted by GANA, and how they will evolve over the time and how their components become integrated as part of the Network-Level-DEs.

NOTE: Here, the focus is simply to illustrate how the "migration" of an NMS or "co-existence" of the traditional NMS and Knowledge Plane could happen. The NMS is not necessarily a classical NMS. In this illustration example, the EMS is missing between NMS and Node and so Reference Point "Refp_5" could be what is the so called "Direct Interface" that gets rid of the EMS. This is not to say that EMS's should be eliminated. Generally speaking, as indicated on Figure 64, Network-Level-DEs (in the Knowledge Plane) evolve EMSs or NMSs or may be implemented as separate run-time entities that then interwork with EMSs or NMSs.

In the possible co-existence of NMS and Network-Level-DEs in the evolution of network management, there are two possible implications of the co-existence and evolution that should be considered on the Reference Points ("Rfp_4" and "Rfp_5") identified on the figure:

1) The Manager in the traditional NMS, or the MBTS, may create a Wrapper packet/message that encapsulates the COMMAND and send the packet/message to the node/device where the Node-Main-DE extracts the COMMAND and relays it to the appropriate FUNCTION-LEVEL-DE responsible for autonomically managing and controlling the ME targeted by the COMMAND. The DE then reasons about whether to apply the COMMAND, and if yes, the DE executes the COMMAND directly on the ME's management-interface OR issues the COMMAND via the "loopback interface" to the local Management Agent/Deamon/Server for execution if the DE manages and controls the ME indirectly through the Agent/Deamon/Server.

2) The Manager in the traditional NMS, or the MBTS, may send the COMMAND directly in its native (technology-specific) form, to the node/device using the means and procedures used today, in which case the Management Agent/Deamon/Server on the target node/device should then be modified to NOT directly execute the COMMAND on the target Managed Entity (ME) but should relay it to the appropriate local DE (inside the node/device) that is responsible for autonomically managing and controlling the ME targeted by the COMMAND (especially if it is a "WRITE/SET" type of a COMMAND). The DE then reasons about whether to apply the COMMAND, and if yes, the DE issues the COMMAND via the "loopback interface" to the Agent/Deamon/Server for execution.

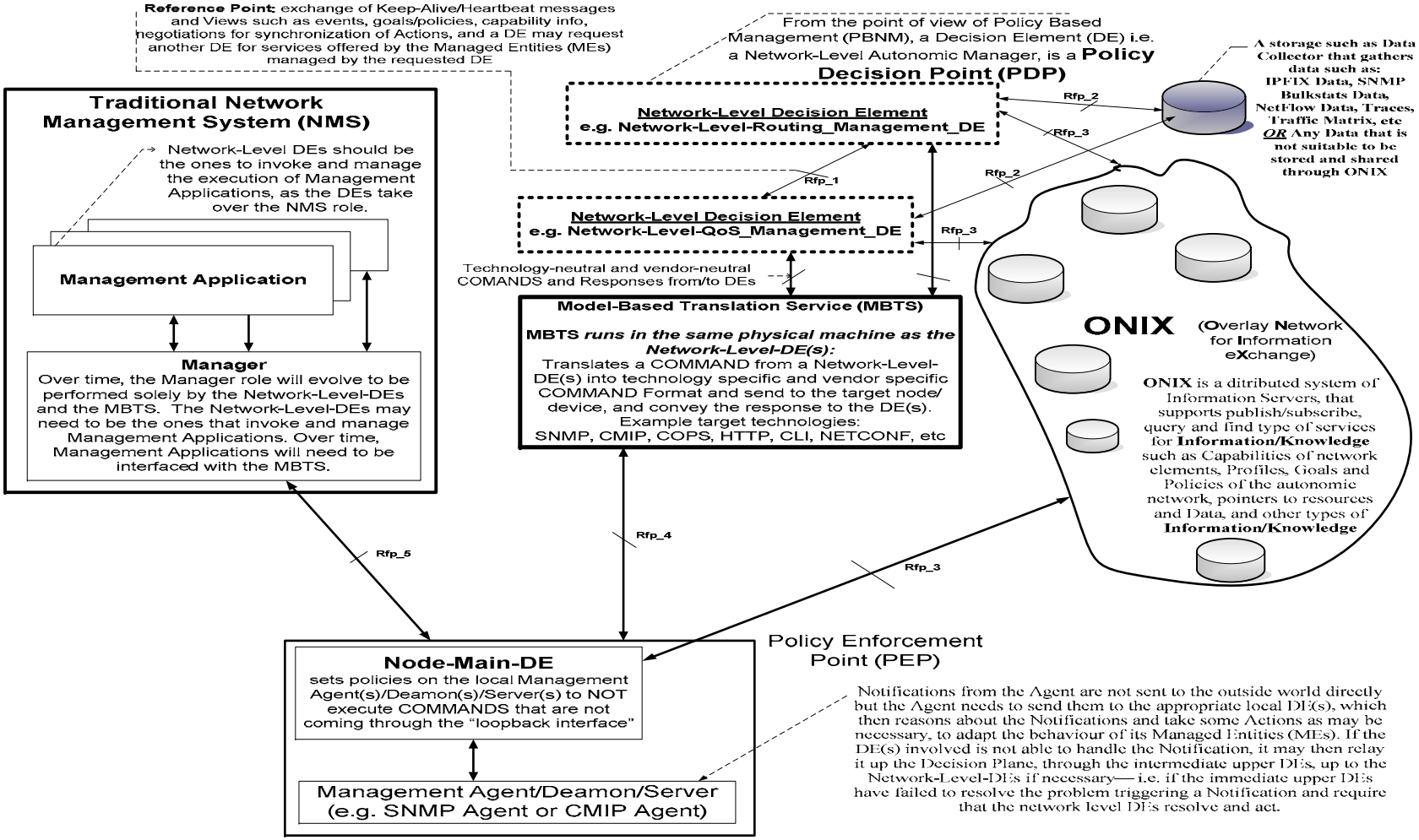
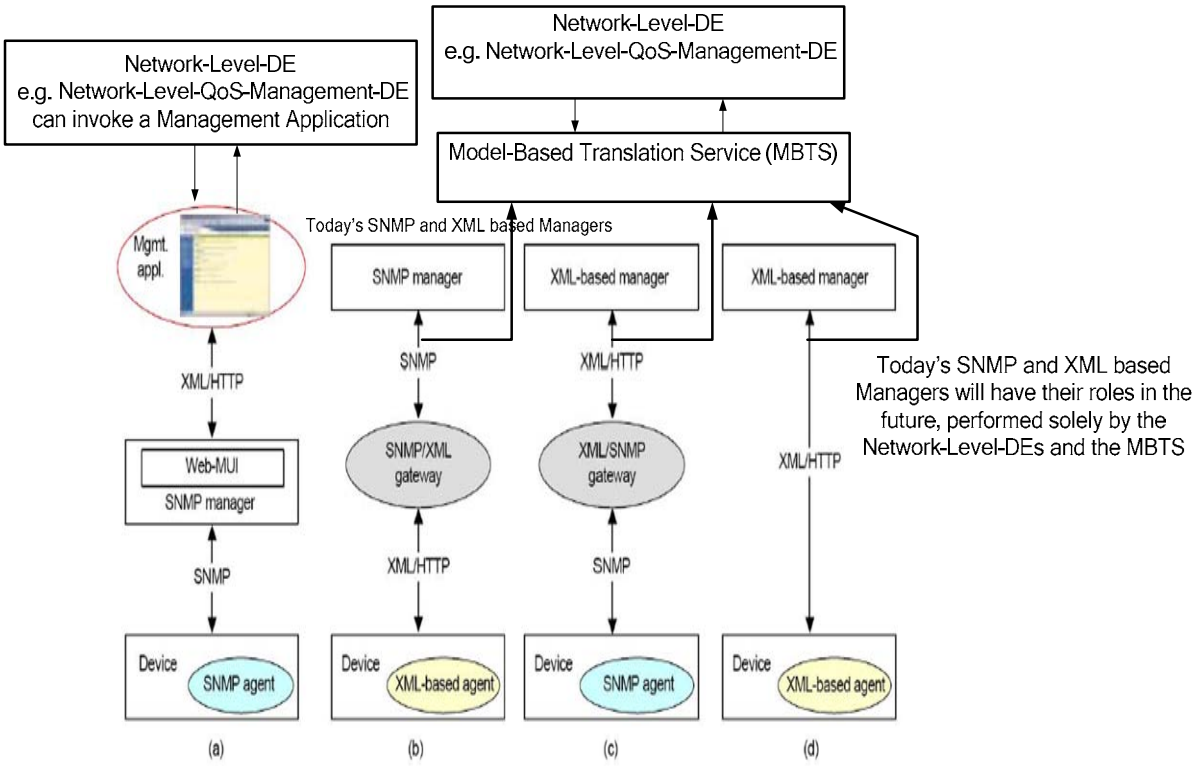


Figure 64: The Evolution of Network Management Paradigms as necessitated by the GANA approach

11.7.1 Migration/Co-Existence scenarios that would still accommodate SNMP/XML/HTTP-based management within GANA Network_Level_DEs-driven management

Figure 65 illustrates the link between a DE and a Management Application that uses HTTP and XML-based management approaches. The Network-Level-DE (s) that could execute management applications is decided by the designer of the DE and/or the application designer. Also illustrated is how DEs and the MBTS would fit into the picture that includes today's SNMP and XML based managers. Today's SNMP and XML based Managers will have their roles, in the future, performed solely by the Network-Level-DEs and the MBTS. However, what is also worth mentioning is that the SNMP messages sent to Agents on target devices shall be subjected to the Policies enforced on the target node/device by the Node-Main-DE of the target node/device, as described earlier. Meaning that the Agent may need to be constrained from applying the COMMANDs (especially the WRITE/SET COMMANDs) directly since it shall be given a go ahead by the node/device's local DEs responsible of autonomically managing the Managed Entities (MEs) being targeted by the COMMAND. For more details on that various types of setups of the indicated management options with the different technologies, we refer the reader to [i.58].

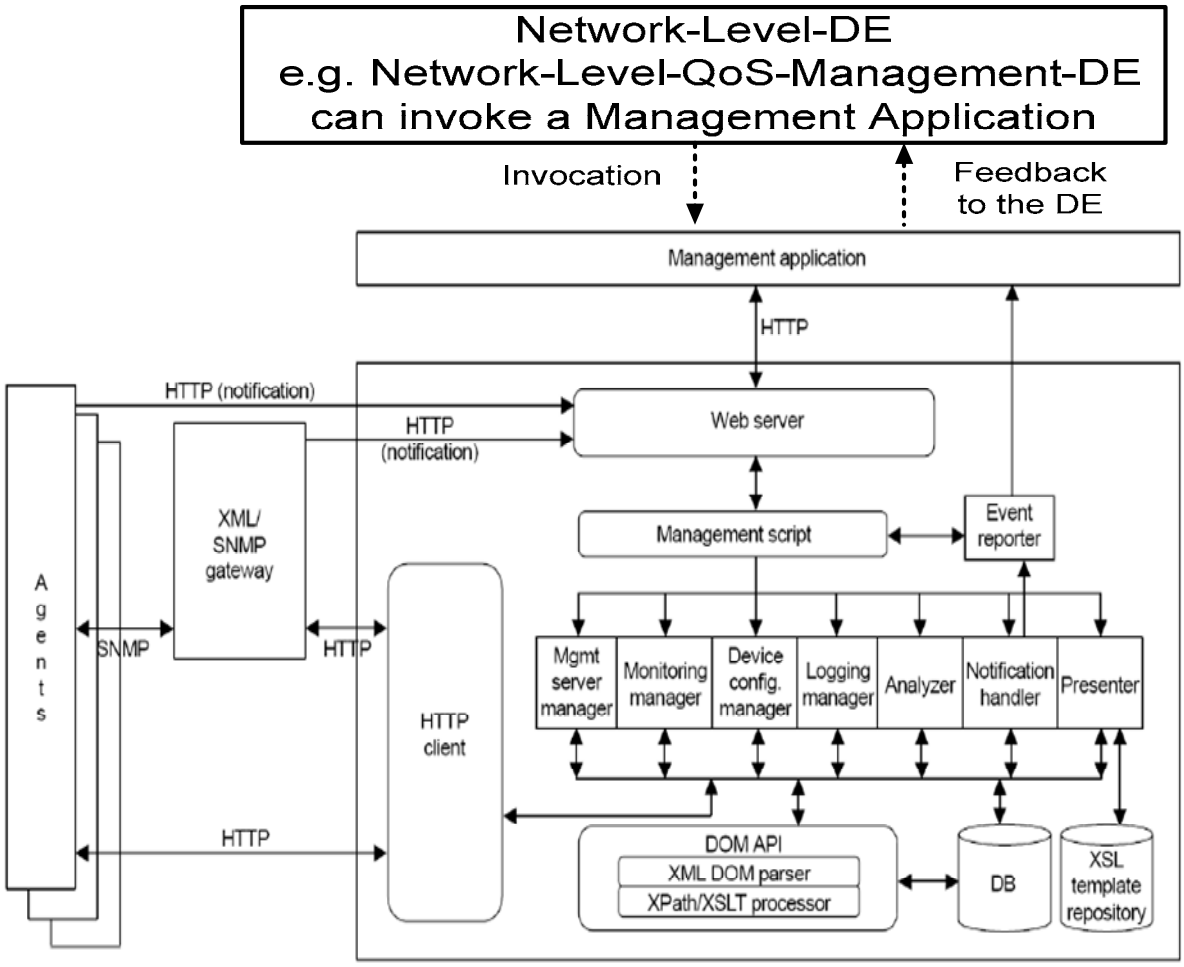


NOTE: Copyright of the original diagram that was modified belongs to Taylor and Fransis Group, LLC, Auerbach Publications: Book "Advances in Network Management by Jianguo Ding: ISBN 978-1-4200-6452-0.

Figure 65: Migration (or Co-Existence) scenarios from (with) current SNMP/XML/HTTP-based management to (with) GANA Network_DE-based management

11.7.2 Link between a DE and Management application that is HTTPand XML based

Figure 66 illustrates the link between a DE and a Management Application that uses HTTP and XML-based management approaches. The figure elaborates more details on HTTP and XML based management. The Network-Level-DE (s) that could execute management applications is decided by the designer of the DE and/or the application designer. Again, what is also worth mentioning is that the SNMP messages sent to Agents on target devices shall be subjected to the Policies enforced on the target node/device by the Node-Main-DE of the target node/device, as described earlier i.e. the Agent may need to be constrained from applying the COMMANDs (especially the WRITE/SET COMMANDs) directly since it shall be given a go ahead by the node/device's local DEs responsible of autonomically managing the Managed Entities (MEs) being targeted by the COMMAND.



NOTE: Copyright of the original diagram that was modified belongs to Taylor and Fransis Group, LLC, Auerbach Publications: Book "Advances in Network Management by Jianguo Ding: ISBN 978-1-4200-6452-0.

Figure 66: The link between a DE and a Management Application that uses HTTP and XML-based management approaches

Web-based Enterprise Management that employ DCOM/ XML, CIM and COM based access to managing various managed Objects, would follow similar integration approaches described above, for integrating Decision Elements as the Autonomic Managers of the GANA Decision Hierarchy.

11.8 GANA and OSS's

Some of the aspects related to OSS's have already been reviewed and discussed in previous clauses in relation to TMN Logical Layered Architecture (LLA) and fusion of NMSs and EMS's with GANA.

Figure 67 illustrates the integration of an OSS with GANA Network-Level-DEs and the ONIX distributed system. It indicates the type of Interface that needs to be shifted towards communication with Network-Level-DEs. It also indicates the possible type of an interface for making use of ONIX Services and accessing Information/Knowledge shared through ONIX.

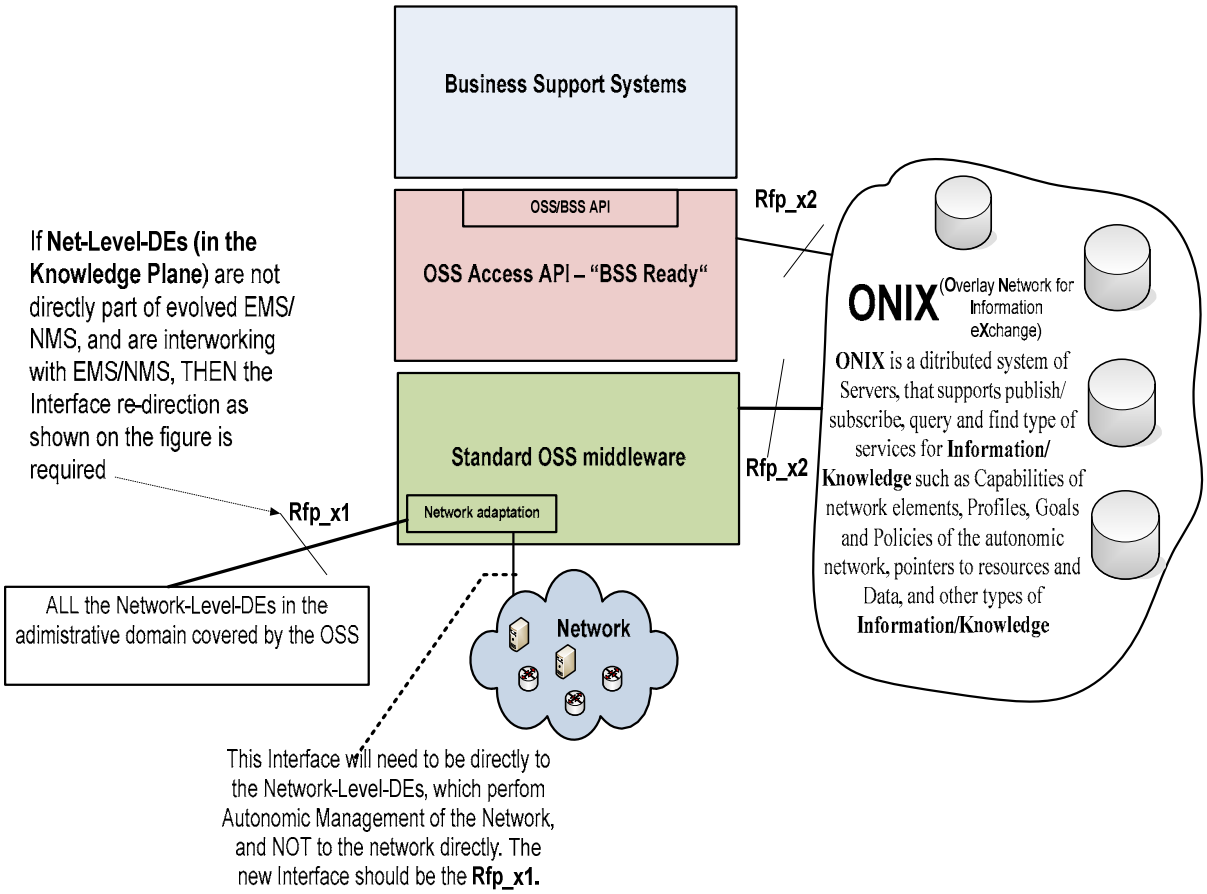


Figure 67: How to integrate an OSS and GANA Network-Level-DEs and the ONIX

Other points noted in relation to previous discussion on this subject: EMS and OSS interfaces have to be standardised and that the Decision-Making process in both systems should be coordinated. The GANA DE-to-DE interfaces are meant to address this issue if one assumes that an ***evolved EMS is a set of Network-Level-DEs*** that implement diverse autonomic functionalities.

11.8.1 Requirement framework for a Policy-Based Management

GS AFI 001 [i.51] specifies in more detail the Requirement framework for a Policy- based management.

NOTE 1: Also, in relation to this subject, in the present document, *Policy-based management* within the *GANA Reference Model* is a subject covered in the clause *"***Network Governance, Service Management View, and the Use of Policies, Policy Frameworks, and Profiles in GANA**".

Here, we reflect on how the Reference Model defines the concepts that are relevant to topic, in order to provide some coherence across the two specifications (GS AFI 001 [i.51] and GS AFI 002 (the present document)).

From management architecture perspective, Figure 68 maps "Operational requirements" specified in the corresponding clause in GS AFI 001 [i.51], which need analyzed when considering the real operator architecture which will handle the implementation. The network environment (Vendors' domain in the figure) will be managed by business objective which will drive the service objective. The service objective will be then translated into policy rules which will be enforced in the operator's network through the operator's Network Management System (NMS). The policy enforcement will be then executed by the different vendor's Element Management System (EMS) and the Network Elements (NE) forming the operator's network. Each layer of the whole architecture will take into account its intrinsic constraints. All these aspects are covered in the clause "**Network Governance, Service Management View, and the Use of Policies, Policy Frameworks, and Profiles in GANA**".

Autononomic & Cognition requirements

This requirement aims at introducing Self-Awareness, Learning, and Reasoning capabilities and mechanisms. It also includes gathering information, transforming it into knowledge & distributing it for various needs. There is also a need to building and ensuring Trust & Confidence as well as Stability in Autonomics loops and in the network. The subject of stability is addressed in the corresponding section in the present document.

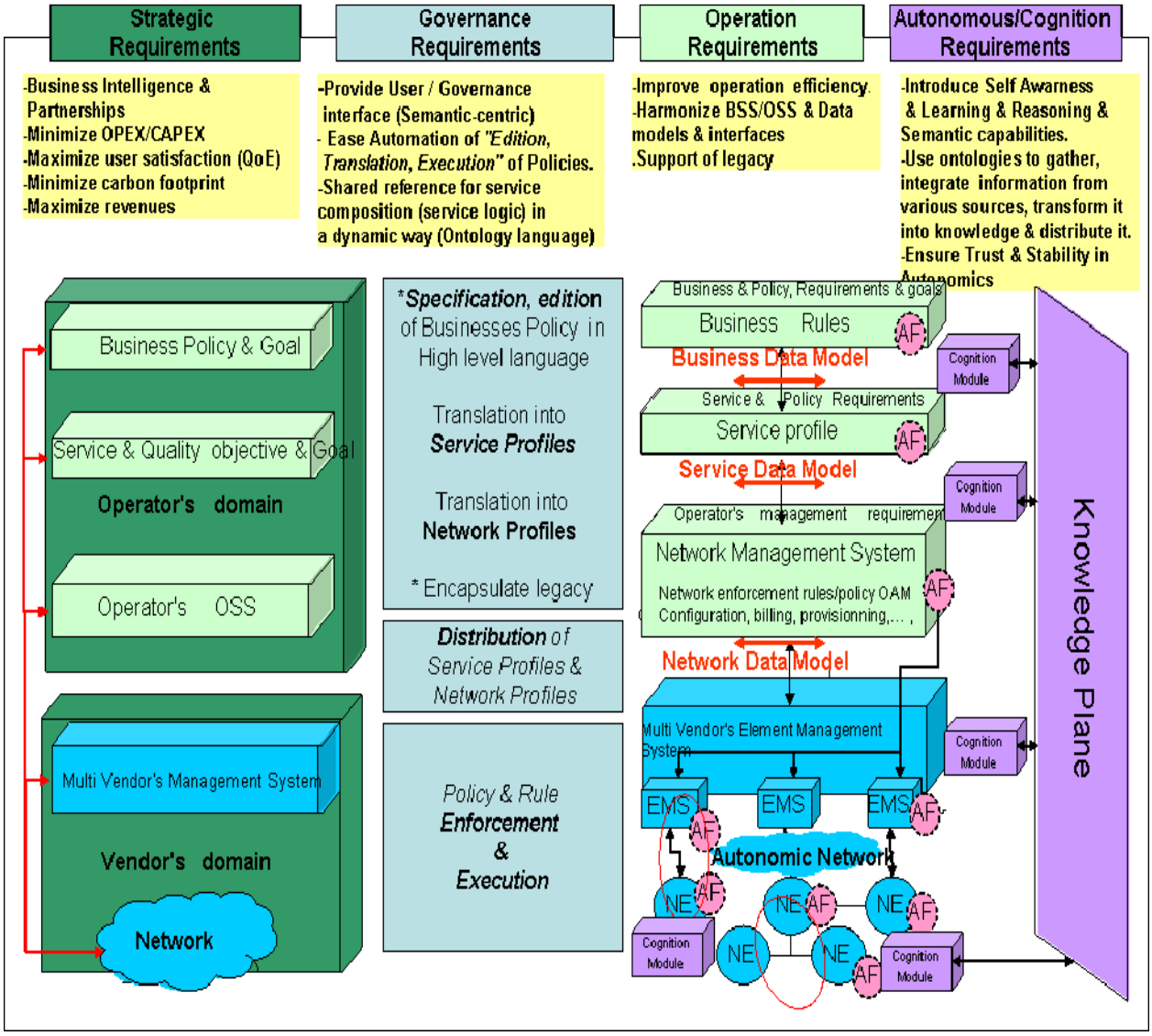
From management architecture perspective, Figure 68 maps these "Autonomic & Cognition requirements" to the real management architecture.

The cognition aspect is shown by "cognition module" (purple boxes) in the right part of the figure. These cognition modules are used to retrieve relevant knowledge from data/information and it allows the cross-control loop interactions accordingly. These cognition modules can also be seen as "brokers" where various knowledge sources store the knowledge in a controlled and secured way, while, the users of these knowledge retrieve required and relevant knowledge through a subscription model. In this context, a dissemination plane is used to disseminate knowledge and decision.

NOTE 2: A **Decision Element (DE)** embeds (implements) a "**cognition module**", and the Knowledge Plane is seen as Cognition at higher level than cognition in nodes since cognitive properties of the Knowledge Plane are properties exhibited by Network-Level-DEs.

NOTE 3: In the present document, in the clause "**Cognitive Networking and Knowledge Plane as part of the GANA Decision Plane, and Information/Knowledge Sharing**", the Knowledge Plane is defined. In the section the following aspect is reflected:

- The Autonomic capabilities are introduced in this management architecture thanks to Autonomic Functions (AFs) (pink boxes) which are realized by Decision-making Elements (DEs) at different layers. AFs may be decomposed into "sub-AFs" thereby creating the notion of nesting of Control-Loops as illustrated by the GANA Model. AFs could be embedded in different parts of the network (network element, management system, OSS, etc). The implementation of these AFs in the autonomic network architecture is driven by operator's policy. In this perspective, we represented the 3 autonomic network architectures schemas symbolized by red circles as specified by 3GPP/SA5 for SON related OA&M:
  - Distributed manner: the AFs are embedded in the NE only.
  - EMS-Centralized manner: the AFs are embedded in EMSs and NEs.
  - NMS-Centralized manner: the AFs are embedded in the operators' OSS and NEs.



NOTE: The indicated Cognition Modules may all be implemented within the Knowledge Plane's DEs.

Figure 68: Requirement framework for a Policy - based management of an Autonomics Network

11.9 Network self-management based on capabilities of the network as described to the overlying OSS processes

Nowadays, the technologies often run far ahead of the ability to manage them. This may result in complex networked systems requiring manual configuration and dedicated management provided by very expensive experts. Such a model is obviously cost ineffective and it cannot scale. One of the aspects that seems to be missing in terms of the desirable behaviours is any consideration of what is the **cost of ownership** of a given behaviour in the context of planning, fulfilling/configuring and assuring this behaviour in the end to end system. It is very easy to create a network feature that is bespoke and therefore difficult to integrate and from an OSS point of view the need would be to understand how the capabilities of network behaviours can be described in a common standardised way that makes them easy to integrate and describe to the overlying OSS processes.

For this reason there is an urgent need for automation in terms of planning, configuring and assuring network behaviours. Next generation Operations Support Systems (OSS) need to be able to manage a diverse set of network devices offering different capabilities and exposing various behaviours. In particular some of the devices might be equipped with capabilities allowing for certain **cooperative** and **non-cooperative** behaviours.

Currently developed next generation networks often assume the possibility of having devices able to route data in a cooperative manner to increase the QoS offered by the system. Such devices are preliminarily to be deployed by network operators as fixed, movable and mobile ones, while the mobile relay nodes may be also user devices in a bit longer perspective. It puts then certain requirements on the system to be able to discover the capabilities of distinct devices and use the acquired information for the purposes of exploiting these behaviours whenever beneficial for the overall performance. This task involves other OSS level routines such as **inventory management** and **service assurance**.

The aforementioned cooperative behaviour can be highly beneficial from the system performance perspective, however, it will be coming at a certain cost. This cost may result from the fact that either:

- additional network operator devices may need to be deployed so they can be automatically configured to expose cooperative behaviours when there is a need;
- user devices may be needed to instantiate cooperation and in that case users may want to trade what they can offer for some other privileges or benefits.

There is then a need for a logic that would be able to analyse applicability of such behaviour for specific deployment and in given circumstances (longer term perspective) and consequently make use of them by imposing certain policies. It might turn out that in some cases cooperative behaviours can be translated into revenue, as well as they can add to service assurance.

In different circumstances different cooperative behaviours may be exposed:

- Base Station/Access Point can instantiate cooperation with Relay Node(s).
- Relay Nodes can instantiate cooperation among themselves.

Otherwise non-cooperative behaviour may be advantageous.

11.10 Network-intrinsic autonomic management (in-network management) via DE-to-DE interactions across nodes/devices

As seen on Figure 20, the GANA facilitates for "network intrinsic management" or "in-network management", which is a kind of network management or control that does not require coordination by the vertical extension of the Decision (Management Plane) to the Network Level Managers or Element Managers (i.e. Network-Level-DEs in GANA terms), BUT that the control and coordination is orchestrated by the network nodes/devices/elements themselves. The peer relations that can be established by DEs at the various levels can be seen as the formation and management of "overlay compartments" in which policies for both "membership" and "information/knowledge exchange" may be applied. On the same figure (Figure 20), the DE-to-DE interfaces provide the means by which the network nodes/devices may exchange control information that results in the nodes/devices of some network scope co-operatively converging on performing behaviour such as updating some state in their local Knowledge repositories, in a similar fashion to how say OSPF works. Since the control plane is seen as an extension to the management plane, the DE-to-DE control information exchange across nodes can be regarded as "network-intrinsic management" or "in-network management". The figure provided earlier in the present document, ***Hierarchical, Peering, Sibling relations and interfaces of DEs in GANA,*** illustrates how this kind of management can be realized.

In-Network Management is aiming at performing supervision activities in a highly distributed way, by exploiting the self-organization among DEs controlling single nodes, resources (e.g. protocols, protocol stacks and mechanisms) that constitute Managed Entities (MEs) of specific FUNCTION-LEVEL-DEs, or network subsystems, and gossiping protocols to exchange and aggregate data among them. According to In-Network Management, each DE executes a local logic (e.g. implemented according to an autonomic control loop) in charge of supervision of node capabilities (i.e. the NODE-MAIN-DE takes care of this aspect). In GANA, Function-Level-DEs are in charge of the supervision of capabilities of resources (e.g. protocols, protocol stacks and mechanisms) that constitute Managed Entities (MEs) of specific Function-Level-DEs, and should communicate the capabilities of their MEs to the NODE-MAIN-DE that aggregates and reasons about the overall capabilities of the whole node. However, some local logic is also required at each level in the GANA Decision Plane within the node. The local logic can cooperate, e.g. by means of exchanging data. In general, the maximum "distance" over which two peer DEs can communicate effectively is assumed to be small compared with the size of the entire system: each DE communicates only with a few nearby neighbours, interconnected through a (self-managed) overlay network.

By exchanging data, by means of gossiping protocols, DEs can create an approximated knowledge of the (dynamically changing) global properties of the overall system, and use it to perform local supervision decisions and perform self-adaptive behaviours.

The distribution of the logic and the interactions through overlay networks guarantee the development of scalable and robust algorithms, also in multi-domain contexts.

In particular, In-Network Management relies on the following mechanisms.

Embedded autonomic behaviour: It is in charge of monitoring its internal behaviour (and that of the features of the supervised node) and adapting it to internal events and to the changes in its external environment; this would remove the strict distinction between managers and managed entities, where the (autonomic) supervision logic is located outside the node; embedded autonomic behaviour of different nodes can interwork, according to a peer-to-peer interaction model among the DEs involved, in order to cooperate and provide a coordinated supervision logic.

Self-organized overlays: Overlays constraint the interactions among few nearby elements, in order to keep the communication localized (e.g. to avoid flooding): only elements interconnected though the overlay can directly interact. Moreover, overlays rely on self-organization capabilities (implemented by the elements participating to the overlay) for their creation, maintenance and optimization; in this way, they provide a scalable and robust environment for interconnecting elements in a distributed system. Cluster algorithms can be used in order to create and maintain overlays among elements sharing the same properties (e.g. all the elements managing the same type of network resource/function e.g. the Routing-Management-DEs shown on the figure above autonomically manage and control the routing protocols and mechanisms of the node/device). In GANA, DEs at the Function-Level in particular are assigned to autonomically manage and control specific MEs, and may require forming peers with similar DEs on the same level and hosted by different nodes/devices in the network. In GANA, the Function-Level-Routing Management DE is responsible of autonomically managing all the routing protocols (e.g. OSPF, RIP, etc.) and associated mechanism hosted by the node/device. DEs interconnected through overlay links can interact in order to cooperate in executing distributed supervision algorithms and in exchanging information/data.

Gossiping protocols and self-aggregation [i.27**]:** These mechanisms can be adopted for dissemination and aggregation of information across elements in a distributed context. Gossiping protocols are based on iterative information exchange: during each protocol step, a node exchanges to (a small subset of) its neighbours (in an overlay) a small amount of data: they can be used to spread information, by limiting the flooding of data. Self-aggregation algorithms exploit gossiping protocols for disseminating combined information: during each step, a node combines the data received from its neighbours during the previous step, with its local state, updates its state and forwards the combined information to (some of) its neighbours; Self-aggregation algorithms can be used to create non-local/global knowledge from local information, by using only local information.

Knowledge Field: Knowledge is an essential element of cognitive and autonomic loops. The word field has to be intended according to the Physics meaning. For instance, the knowledge field could look like a sort of gravitational field emitted by each autonomic component i.e. a DE (e.g. representing the current amount of available resources): by following the local shape of the fields components could perform decision-making and actuation processes (e.g. concerning the discovery of available resources). In other words, autonomic components i.e. DEs in the GANA Model could sense the knowledge field and act accordingly to specified behavioural patterns or plans. Knowledge field models could potentially be implemented, as an overlay network such as the ONIX system, on any middleware providing basic support for data storing, communication and event-notification. What is required is to provide simple storage mechanisms (to store field values), communication protocols and primitives (to efficiently propagate field values among the peers), and basic publish-subscribe mechanisms. The ONIX distributed system facilitates these kinds of services. The knowledge field could be stored, in a decentralized way, either on DEs in network nodes/devices, or on specialized nodes in charge of generating, handling and distributing the knowledge across the system. Gossiping can achieve an efficient way for distributing and aggregating the information. Reasoning techniques and data correlation features might also be necessary in effective exploitation of the "knowledge field".

Explicit Control Information Exchange between a set of DEs: Some mechanisms may be required for explicitly exchanging control messages between a number of DEs. For different requirements for DE-to-DE communications across node/devices, GANA DEs, residing in two or more nodes/devices, may require the ability to perform the following operations:

- Negotiating the way their Managed Entities (MEs) e.g. Protocols should be configured. The need to negotiate can be in the case of lack of a centralized coordinator and the need for peer DEs to **self-organize**.

- Soliciting for Capabilities of the peer or multiple peers based on the features supported by their associated MEs's Capabilities (i.e. capabilities of the managed protocols and mechanisms).

- Self-Advertising Capabilities to peers on the link or selected peers by policy.
- Exchange Trust related info. This could be done by the Node-Main-DE (autonomically manages and controls the behaviour of the whole node) or also at lower level DEs within the node.
- Expose "Views" to peer DEs, such as detected incidents, misbehaviours, etc., which can be used by the peers for managing the configuration of their associated MEs.
- Request "Views" from peer DEs.

All the nodes that are involved in In-Network Management shall ensure that as their DEs at the three levels (but most importantly level-2 and level-3) shall have the ability to form peers with similar DEs in other nodes, the DEs shall implement the mechanisms mentioned above as well as having the properties described below. In particular, the DEs that address certain aspects related to in-network management can be structurally viewed according to two autonomic control loops:

- *Node-Level overall internal control loop realized by the NODE-MAIN-DE and its internal sub-DEs*: it adapts the behaviour according to the planned and unplanned internal events and/or events of the controlled node capabilities, e.g. for improving performance of internal functions or for detecting and recovery from internal faults.
- *At each of the levels: Node-level, Level-2 and Level-1 in GANA (especially the Node-level i.e. Level-3 and the Function-Level i.e. Level-2) we consider having also what can be perceived as an external control loop: Such a control-loop* should react to environmental changes, e.g. to the events/messages sent by DEs interconnected through an overlay at a particular level in GANA, by tuning the internal behaviour and coordinating it with those of the neighbours DEs interconnected through the overlay network. The external control loop provides features for participating to self-organized overlays (e.g. joining, leaving, reorganize links after a failure, etc.); moreover, it can be used to implement features for achieving gossiping and self-aggregation of information through simple cooperative interactions (e.g. inspired to biological or social metaphors). The NODE-MAIN-DE shall decide on the overall policies of the node regarding the behaviour of the lower level DEs in participating in overlay formations and exchanging information/knowledge with peer DEs that are members of a particular overlay.

All the DEs involved in an In-Network Management interact according to a peer-to-peer model according to the Peer Relations between DEs hosted by different GANA compliant nodes/devices.

The definition of In-Network Management solutions requires the specification of the messages used for these interactions. The DEs could be, instead, implemented according to different autonomic models/frameworks (this is particularly useful in multi-domain contexts). In particular, it is necessary to define the following suites of messages/protocols including the control information exchange requirements described above on "*Explicit Control Information Exchange between a set of DEs*":

- overlay management: protocols for discovering/advertising and joining to an overlay (according to the node/DE properties), for maintaining the overlay, in case of failures and/or DEs leaving the overlays (possibly in an unexpected/unplanned way), for optimizing the links topology, etc.;
- information exchange: protocols for the exchange of information among the DEs interconnected through an overlay;
- distributed supervision messages: protocols for implementing cooperative behaviour for achieving distributed supervision algorithms, such as for fault detection and recovery, load balancing, optimization in resource usage, etc. (examples are provided in [i.22]; each supervision area should identify a set of related messages, which should also include the exchange of policy information (e.g. to share some thresholds);
- negotiation: protocols for negotiating the information to exchange, the cooperation level through distributed algorithms, and the "views" exposed by DEs;
- election of lead member of an overlay e.g. a DE hosted by one of the nodes involved may need to be elected to play some kind of co-coordinator role for some tasks.

It is important to consider that in In-Network Management, nodes/DEs are not forced to cooperate in algorithms and information exchange; this is particularly in case of multi-domain contexts, where nodes/DEs could belong to different administrative domains. In fact, each DE could implement a selfish behaviour; the other DEs could be able to detect non-cooperative DEs and in case punish it (e.g. by removing it from the overlay, or by not considering its messages).

In-Network Management can co-operate with autonomic solutions based on hierarchy of supervised/supervisor nodes (e.g. according to MAPE-K model). For instance, a supervisor such as an upper DE shall be able to react to events that are produced by a lower DE when it is not able to resolve certain situations by means of In-Network management mechanisms. For examples, when a DE is not able to recover a fault by using the distributed supervision algorithm defined by the In-Network management, it can inform its supervision system/DE, which elaborates and returns the corrective actions.

12 Federation in GANA

Future Internet will be characterized by the existence of autonomic systems for (specialized in) managing particular technologies/segments/networks. Hence, federation of these autonomic systems will be the key concept in order to enable secure, trusted and optimal end-to-end service provisioning.

12.1 Definition of federation

The definition of the term "federation" is rather vague. Some definitions can be found in the literature, for instance:

- "Federation is considered to be a persistence organizational agreement which enables multiple autonomous entities to share capabilities in a controlled way" [i.47].
- "A federation is a set of domains that are governed by either a single central authority or a set of distributed collaborating governing authorities in which each domain has a set of limited powers regarding their own local interests" [i.48].
- "Federation is a model for the establishment of a large scale and diverse infrastructure for communication technologies, services, and applications and can generally be seen as an interconnection of two or more independent administrative domains for the creation of a richer environment and for the increased multilateral benefits of the users of the individual domains" [i.49].

Federation can be seen as:

- the interoperation between different kinds of administrative domains; or
- the agreement between different Decision Elements (DEs) belonging to different domains. Such an agreement may concern the negotiated way the peer DEs shall configure the behaviour of their assigned Managed Entities (MEs) e.g. protocols to fulfil the required network behaviour, and such DE-to-DE negotiations may be governed by some policies (the reader is referred to the Network Governance clause of the present document).

12.2 Approach on federation

The steps that need to be followed for addressing the implication and requirements that are imposed with respect to federation are:

- Definition, identification of domains.
- Identification of interfaces for which the notion of domain boundaries can be identified i.e. two or more entities connected by the interface are considered as belonging to different domains.

A "domain" can be defined based on administrative or technological aspects. For example, the following comprise a collection of different kinds of "domains" and domain categories as appearing in research and/or standardization literature:

- Operator domain.
- Business domain.
- Service domain.
- Network domain.

- Fixed domain.
- Wireless domain.
- Routing Domain:
  - Intra-.
  - Inter-.
- QoS Provisioning related domains:
  - DiffServ, IntServ, MPLS, etc., domains.
- Collision domain.
- Security domain.
- Trust domain.
- Other type.
- etc.

Two or more Decision Elements (DEs) that form peers across a Reference Point of nature "DE-to-DE interface across multiple network elements/devices" need to discover the following types of information concerning their peers:

- **Domain Type(s):** whether technical or administrative, of which the DE is member and is willing to share this information (subject to security and trust policies). Regarding the discovery of a "technical" domain to which a DE belongs (or is able to orchestrate and autonomically manage and control), the GANA Model includes the concept of Capability Model and its self-description and publishing by a DE (refer to the clause on Network Governance), through which information about a "technical domain" is encapsulated. A Decision Element should aggregate capabilities of its assigned Managed Entities (MEs) and advertise the descriptive information when required (subject to security policies). For example, a *Function-Level Routing-Management-DE* in a routing device running two routing protocols e.g. OSPF and BGP, would indicate the routing protocols (its Managed Entities) under its control, in its Capability Model description.
- **Domain Identifier(s):** Administrative domains may have Domain Identifiers assigned by the governing authority following some scheme of choice.

Decision Elements (DEs) use such information to verify against security and trust policies, and to behave accordingly in the way they configure their respective Managed Entities (MEs) to fulfil the required network behaviours across domain boundaries.

However, the exchange of such information may possibly be restricted to the Node-Main-DEs discovering and exchanging such information on-behalf of the lower level (Function-Level) DEs that then use the information to behave accordingly in the way they provision a service across domain boundaries.

13 Reference Points relevant in GANA compliant networks

This clause, provides a Table describing all the Reference Points/Interfaces in GANA.

The GANA Reference Model defines the three core concepts listed below, apart from defining enablers for autonomicity, cognition and self-management as well as design and operational principles of autonomic-manager elements (i.e. Decision-Elements (DEs)):

- Functional Blocks (e.g. the different types of DEs and special types of Information-Sharing Components/Systems).
- Reference Points (Rfps).
- Characteristic Information communicated on Reference Points.

The GANA Model defines a set of basic Reference Points whose Characteristic Information and the actual mechanisms or protocol(s) that can be employed to convey the information shall be further elaborated when the GANA Model is instantiated onto a target Reference Architecture such as the NGN architecture, 3GPP architecture, BBF architecture, etc. Figure 69 shows a general architectural model composed of three functional blocks. A reference point between two blocks shall be named as well as the Characteristic Information crossing the particular reference point. GANA DEs such as QoS-Management-DE, Security-Management-DE, Routing Management-DE, and Forwarding-Management-DE, need to exchange information in order to collaboratively drive the autonomic control of various types of resources (e.g. protocols, stacks and mechanisms)-each of which shall be managed and regulated by a specific DE. As described in the GANA Model, each of such DEs when designed to operate at the "network-level", has a "mirror" DE at the "function-level" (i.e. Level-2 in GANA) that operates in a faster time scale for those types of adaptive behaviours for which node/device intelligence is dedicated to the ability of a node/device to react based on decisions made at a low level (i.e. "Function-Level"). For example, "Network-Level-Routing-Management-DE" should have a mirror "Function-Level-Routing-Management-DE" that operates within a routing node/device and implements a faster control-loop (though the two mirror DEs and their associated control-loops interwork with each other).

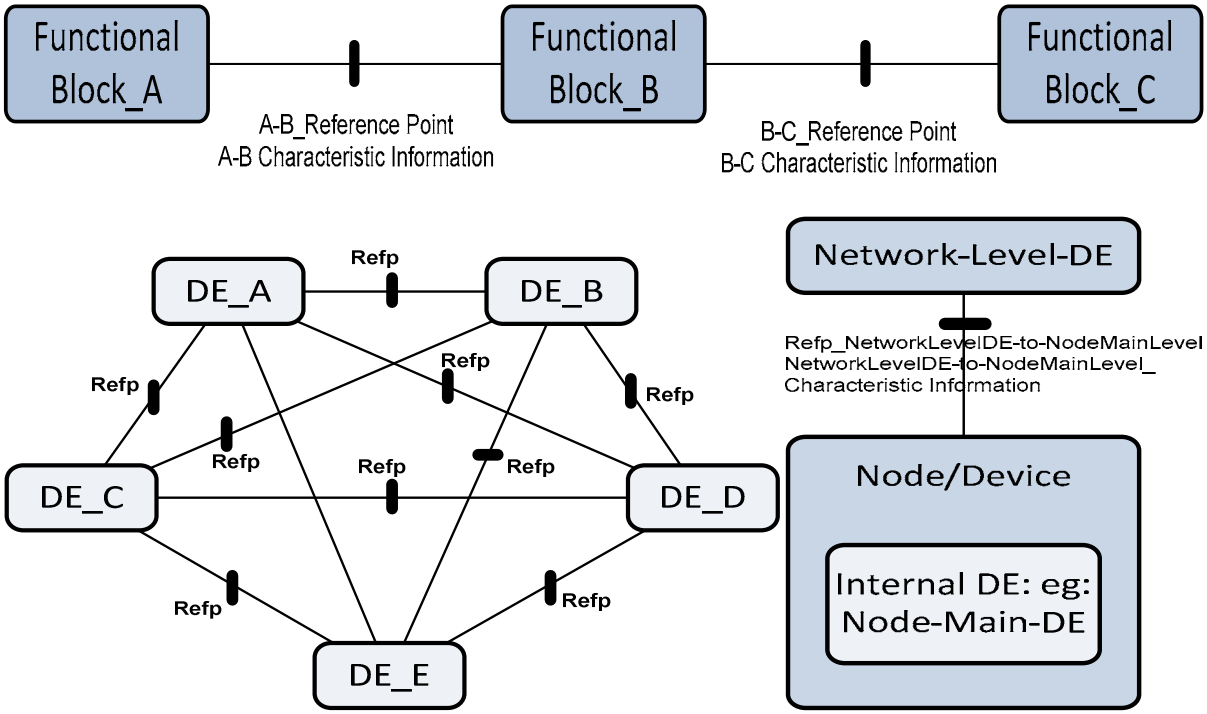


Figure 69: Illustration of some of the Reference Points and Characteristic Information

Table 4: Summary of all Reference Points in the GANA Reference Model

Reference Point Name (see note 1)	Characteristic Information communicated over the Reference Point	Additional Comments, Figure and Section where the Reference Point is described
Rfp_GANA-Level2-AccessToProtocolsAndMechanisms (see note 2)	• **Views** e.g. event notifications, monitoring data are communicated to Function-level-DEs by their specifically assigned Managed Entities (MEs)-i.e. Protocols, Stacks and Mechanisms (see clauses 9.11.5 and 9.11.6 on assignment of DEs to specific types of MEs). • **Commands** are issued by a specific Function-Level-DE e.g. Function-Level-Routing-Management-DE, to its specifically assigned Managed Entities (i.e. protocols and mechanisms such as routing protocols and mechanisms) in order to (re)-configure and regulate the behaviour of the ME(s).	This node/device internal interface is meant to enable the loading of DEs coming from other parties other than the device vendor. The DEs would access and autonomically manage and control the Protocols and Mechanisms of the device. See clause 9.6. See Figure 19.
Rfp_FunctionLevelDE-to-FunctionLevelDE	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Domain Type(s)** to which a DE involved is bound need to be exchanged. **Domain Identifier(s)** of DEs hosted by entities belonging to different administrative domains need to be exchanged. • **Views** can be communicated by a particular Function-Level-DE to other peer Function-level-DEs on other nodes/devices, especially concerning events or issues a function of a node e.g. Routing-Function cannot resolve (by performing some action) without jeopardizing network integrity (objectives). • **Control Information** exchange between Function-Level-DEs via the DE-2-DE interactions to achieve a "network-intrinsic management and control". Such interactions may include the notion of "compartment formation, policies of operation and compartment management" by DE-2-DE communication in a distributed fashion.	See clauses 9.8 and 11.10. See Figure 20 and Figure 34. For Domain Type(s) and Domain Identifier(s) refer to the section on Federation in GANA.
Rfp_NodeMainDE-to-NodeMainDE	Similar types of Characteristic Information as in the case of the Reference Point "Rfp_FunctionLevelDE-to-FunctionLevelDE". The difference being the scope for which the Characteristic Information applies i.e. this case applies to the scope of the node/device level than a particular Function-Level (lower level).	See clauses 9.8 and 11.10. See Figure 20 and Figure 34.

Reference Point Name (see note 1)	Characteristic Information communicated over the Reference Point	Additional Comments, Figure and Section where the Reference Point is described
Rfp_NetworkLevelDE-to-NodeMainDE	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Domain Type(s)** to which a DE involved is bound need to be exchanged. **Domain Identifier(s)** of DEs hosted by entities belonging to different administrative domains need to be exchanged. • **Views** are communicated to Network-level-DEs, especially concerning events or issues a node/device cannot resolve (by performing some action) without jeopardizing network integrity (objectives). • **Commands** may be issued by a Network-Level-DE to the node or to a Function-Level-DE via the Node-Main-DE.	See clauses 9.8, 9.9 and 9.13.5. See Figure 21, Figure 22 and Figure 34. For Domain Type(s) and Domain Identifier(s) refer to the section on Federation in GANA.
Rfp_ModelBasedTranslationService-to-NodeMainDE	• This is a refinement of the Reference Point "Rfp_NetworkLevelDE-to-NodeMainDE" to involve a case whereby Network-Level-DEs communicate with a Node-Main-DE via a Model-Based-Translation Service (MBTS) that translates COMMANDS from Network-Level-DEs and RESPONSES from nodes/devices to a form usable by the targeted entity.	See clauses 9.13.5 and 11.7. See Figure 64 and Figure 34.
Rfp_ModelBasedTranslationService-to-ONIX	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving Information from the ONIX system.** [For example, the MBTS can use the publish/subscribe services of the ONIX that enable Advanced Auto-Discovery of Information and Resources, to retrieve Information about Network Elements/Nodes, such as Capability Description Models of individual nodes/devices, self-advertised/published by an individual node/device upon initialization. Capability Models of a node/device include technological features supported, including management protocols supported and Information about Managed Objects (MOs) of the technologies (e.g. protocols, etc.). Capability Models may include apart from technological features, vendor information].	See clauses 9.13.5 and 11.7. See Figure 64 and Figure 34.

Reference Point Name (see note 1)	Characteristic Information communicated over the Reference Point	Additional Comments, Figure and Section where the Reference Point is described
Rfp_NetworkLevelDE-to-NetworkLevelDE	• **"Views"** such as **Policy changes** by the human operator; **challenges to the network's operation** from the perspective of a particular DE e.g. detected faults, threats, etc; "views" communicated from lower-Level DEs in nodes/devices that require Net-Level-DEs to share and act upon if necessary. • **Domain Type(s)** to which a DE involved is bound need to be exchanged. **Domain Identifier(s)** of DEs hosted by entities belonging to different administrative domains need to be exchanged. • **Negotiations and Synchronization of Actions and Policies.**	This Reference Point between Network-Level-DEs is independent of the types of Network-Level-DEs and so should be considered as a common type of Reference Point between any Network-Level-DEs. See clauses 9.9, 9.13.5 and 11.7. See Figure 22, Figure 34 and Figure 64. For Domain Type(s) and Domain Identifier(s) refer to the section on Federation in GANA.
Rfp_NetworkLevelDE-to-Data_Storage	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages from the DE for retrieval of Data or (Knowledge created out of raw data)** from a storage such as Data Collector that gathers data such as: IPFIX Data, SNMP BulkStats Data, NetFlow Data, Flow Traces, Traffic Matrix, etc, OR Any Data that is not suitable to be stored and shared through the ONIX system. • **Knowledge created out of raw data by Algorithms running on the Data Storage, that operate on raw data and create Knowledge for export to the Knowledge Plane (i.e. to Net-Level-DEs).** • **Data that may need to be communicated by the Storage to the particular DE**	See clauses 11.7 and 9.13.7. See Figure 64 and Figure 38.
Rfp_NetworkLevelDE-to-ONIX-System	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving Information from the ONIX system.**	See clauses 9.13.4; 10 on the Use of Network Profiles, Policies, Objectives, Config-Data, and Capabilities of network elements; and 11.7. See Figure 64.
Rfp_NodeMainDE-to-ONIX-System	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving Information from the ONIX system.**	See clauses 9.13.4; 10 on the Use of Network Profiles, Policies, Objectives, Config-Data, and Capabilities of network elements; and 11.7. See also the figure in clause 9.13.5. See Figure 64 and Figure 34.
Rfp_OSS-to-ONIX-System	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving (mainly) Information from the ONIX system.**	See clauses 11.7 and 11.8. See Figure 67 in particular, Figure 64 and Figure 68.

Reference Point Name (see note 1)	Characteristic Information communicated over the Reference Point	Additional Comments, Figure and Section where the Reference Point is described
Rfp_OSS-to-Network-Level-DEs	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Management COMMANDS normally sent to the network by an OSS** through the so-called "network-adapter interface" need to be rather sent directly to the Network-Level-DE (considering that they are ones that take the full responsibility of performing Autonomic Management of the Network), and NOT to the network directly.	This case applies to configurations where OSS systems are integrated to co-exist and interwork harmoniously with Network-Level-DEs in the overall management of the network. The current OSS-Network Interface would need to be "re-directed" towards Network-Level-DEs (assuming that Network-Level-DEs take full responsibility for network management and control). See clauses 11.7 and 11.8. See Figure 67 in particular, Figure 64 and Figure 68.
Rfp_EMS_OR_NMS-to-NodeMainDE	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Management COMMANDS** targeting nodes/devices designed following GANA principles. A Manager in the sense of a traditional EMS/NMS, may create a "Wrapper packet/message that encapsulates a COMMAND" e.g. a SET/WRITE COMMAND on a Variable, and send the packet/message to the Node-Main-DE of a node/device where the Node-Main-DE extracts the COMMAND and relays it to the appropriate Function-Level-DE responsible for autonomically managing and controlling the ME targeted by the COMMAND. The DE then reasons about whether to apply the COMMAND, and if yes, the DE executes the COMMAND directly on the ME's management-interface OR issues the COMMAND via the "loopback interface" to the local Management Agent (on the node/device) for execution if the DE manages and controls the ME indirectly through the Management Agent (see note 3).	This case applies to configurations where today's management systems are integrated to co-exist and interwork harmoniously with Network-Level-DEs in the overall management of the network. See clauses 11.7 and 11.8. See Figure 64 and Figure 68, see also Figure 63.

NOTE 1: "**to**" in the name does not mean unidirectional communication only. In some of the cases communication may be initiated by either party and can be bi-directional.
NOTE 2: "GANA-Leve2" is also called "Function-Level".
NOTE 3: An alternative to this approach is presented in the corresponding section where this reference point is defined.

Annex A (informative): AFI Top-Down & Bottom-up Methodology

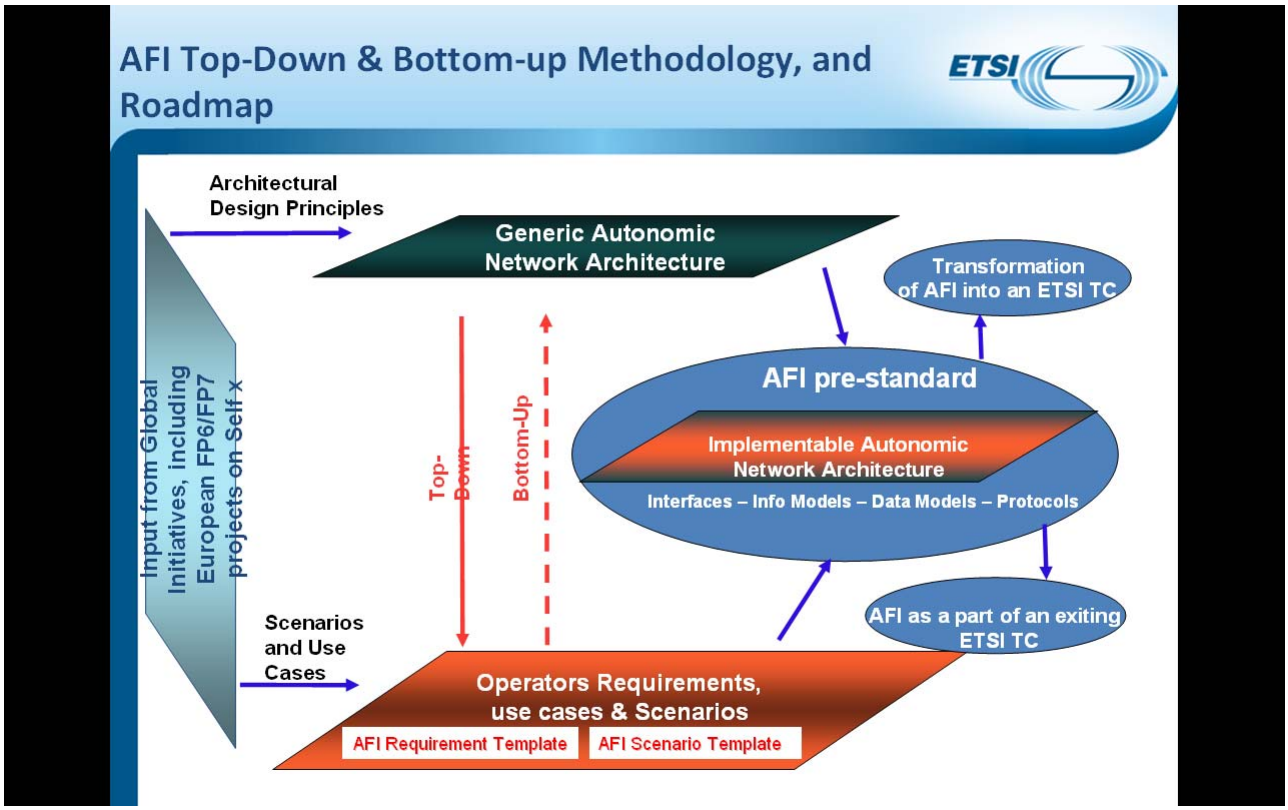


Figure A.1

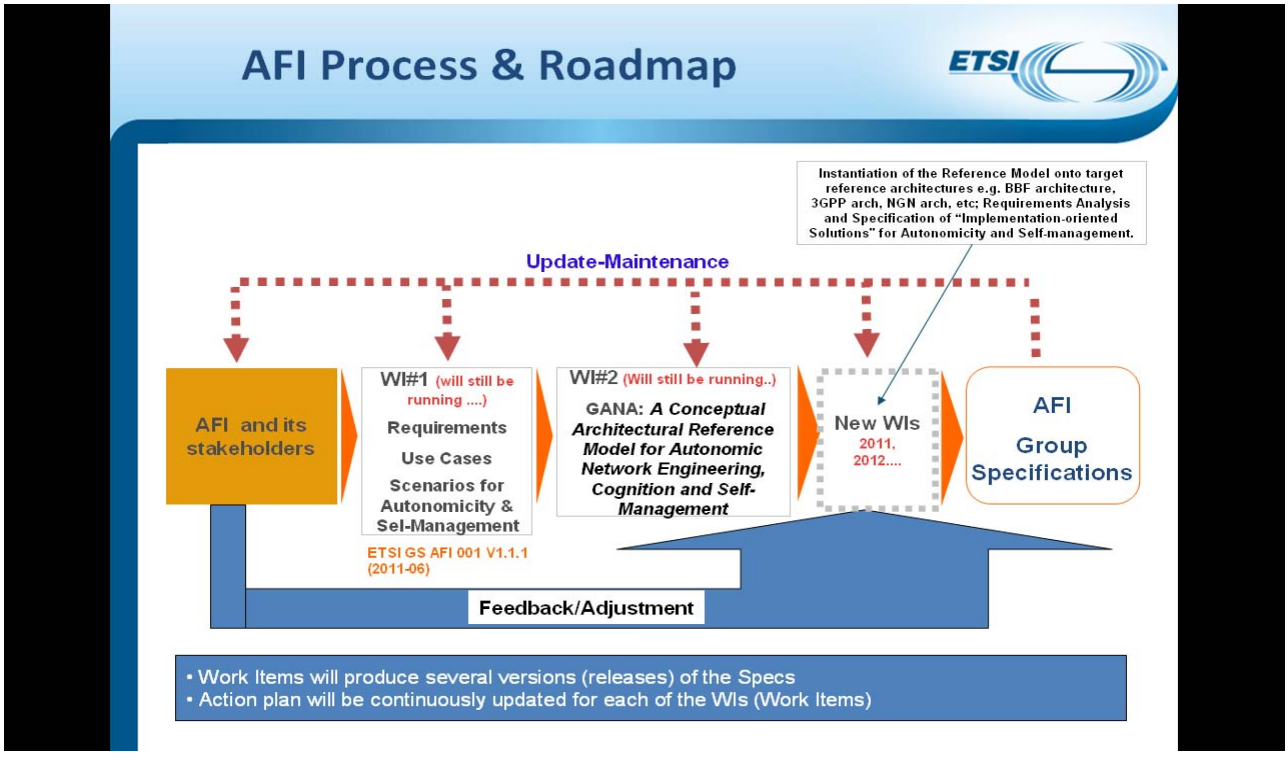


Figure A.2

Annex B (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:
Laurent, Ciavaglia, Alcatel-Lucent

Supporting Editor: Ranganai Chaparadza, Fokus, Fraunhofer; Tayeb Ben Meriem, France Telecom/Orange, Michal Wodczak, Telcorida/Ericsson

Other contributors:

First Name	Last Name	Company
Ranganai	Chaparadza	Fokus, Fraunhofer, IPv6 Forum
Laurent	Ciavaglia	Alcatel-Lucent
Martin	Vigoureux	Alcatel-Lucent
Tayeb	Ben Meriem	Orange/France Telecom
Benoit	Radier	Orange/France Telecom
José Antonio	Lozano López	Telefonica
Juan Manuel	González Muñoz	Telefonica
Beatriz	Fuentes Arenaz	Telefonica
Alfonso	Castro Escudero	Telefonica
Arun	Prakash	Fokus, Fraunhofer
Alexej	Starschenko	Fokus, Fraunhofer
Nikolay	Tcholtchev	Fokus, Fraunhofer
Razvan	Petre	Fokus, Fraunhofer
Antonio	Manzalini	Telecom Italia (TI)
Corrado	Moiso	Telecom Italia (TI)
Michał	Wódczak	Telcordia Technologies
Mick	Wilson	Fujitsu Labs of Europe
Alice	Zhaolin	Fujitsu Labs of Europe
Maurice	Duault	Cisco systems
Michael	Behringer	Cisco systems
Pascal	Thubert	Cisco systems
Andras	Toth	Ericsson
Tony	Jokikyyny	Ericsson
Said	Soulhi	Ericsson
Latif	Ladid	IPv6 Forum
Symeon	Papavassiliou	ICCS/NTUA, Athens (in collaboration with Fokus)
Timotheos	Kastrinogiannis	ICCS/NTUA, Athens (in collaboration with Fokus)
Mary	Grammatikou	ICCS/NTUA, Athens (in collaboration with Fokus)
Vassilios	Kaldanis	Velti (in collaboration with Fokus)
Shiduan	Cheng	BUPT, China (in collaboration with Fokus)
Wendong	Wang	BUPT, China (in collaboration with Fokus)
Yuhong	Li	BUPT, China (in collaboration with Fokus)
Jianguo	Ding	University of Luxembourg (in collaboration with Fokus)
Kevin	Quinn	Waterford Institute of Technology
Steven	Davy	Waterford Institute of Technology
Jesse	Kielthy	Waterford Institute of Technology
Alan	Davy	Waterford Institute of Technology
Brian	Lee	Athlone Institute of Technology
Gerard	Nguengang	Thales
Maurice	Israel	Thales
Szymon	Szott	AGH University of Science and Technology
Katarzyna	Kosek-Szott	AGH University of Science and Technology
Marek	Natkaniec	AGH University of Science and Technology

First Name	Last Name	Company
Athanassios	Liakopoulos	GRNET
Anastasios	Zafeiropoulos	GRNET
Panagiotis	Gouvas	GRNET/ UBITECH
Nancy	Alonistioti	National and Kapodistrian University of Athens
Apostolos	Kousaridas	National and Kapodistrian University of Athens
Tilemachos	Raptis	National and Kapodistrian University of Athens
Vangelis	Gazis	National and Kapodistrian University of Athens
Alexandros	Kaloxylos	National and Kapodistrian University of Athens
Panagis	Magdalinos	National and Kapodistrian University of Athens
Panagiotis	Demestichas	UPRC
Kostas	Tsagkaris	UPRC
Maria	Akezidou	UPRC
Marios	Logothetis	UPRC
Andrej	Mihailovic	King's College London
Taesang	Choi	ETRI
George	Kormentzas	NCSR Demokritos
Lambros	Sarakis	NCSR Demokritos

Annex C (informative): Acknowledgements

The AFI very much acknowledges the European Commission (EC), which is funding a number of FP7 projects under the umbrella "Network of the Future", like EFIPSANS http://www.efipsans.org (some partners of which founded AFI together with some non-EFIPSANS partners), and many other FP7 projects that are represented in AFI and are making contributions to AFI, such as Self-NET http://www.ict-selfnet.eu/, E3 https://www.ict-e3.eu, "Future Internet Engineering" http://www.iip.net.pl/, and others to be added. Also, non-EC funded projects contributing to AFI are to be acknowledged.

Various projects are contributing to the work of AFI in different ways:

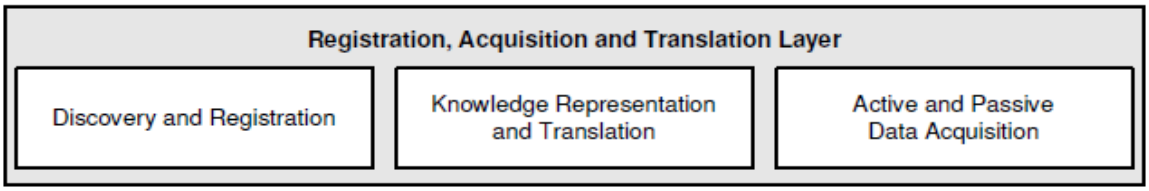
1) by reviewing the AFI work and concepts being specified and providing useful comments on the basis of their own work being done in the current research projects (or even in future projects);

2) mapping their own architectural components validated in the projects, to the concepts and architectural principles specified in the AFI Frameworks such as the GANA Reference Model and its Instantiations/Use Cases in various target reference architectures, in order to identify gaps or concepts and properties that may be missing and should be added to either evolving the Reference Model or its Instantiations/Use Cases.

It is acknowledged that in the field of autonomic networking and computing, various research projects do come up with different kinds of architectural frameworks. In order for the industry to achieve inter-operable autonomics with realistic costs of standardization and implementation, now and in the long-term, results from various projects should be contributed to developing and relying on a commonly-shared (i.e. common baseline) unified standardized framework(s), by adopting, enhancing and evolving the frameworks emerging in AFI (i.e. the Reference Model and the Autonomicity-Enabled Reference Architectures resulting from the Instantiation of the Reference Model onto target implementation-oriented architectures).

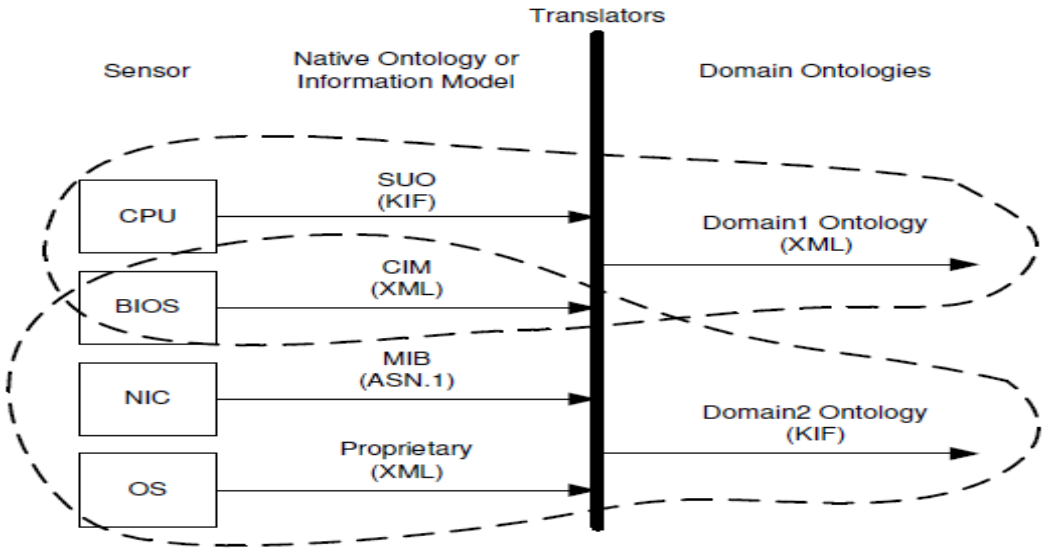
Therefore, AFI Members will continue to add all the projects that are represented in AFI.

Annex D (informative): Other Useful Information relevant in Knowledge Derivation, Representation and Presentation to the GANA Knowledge Plane



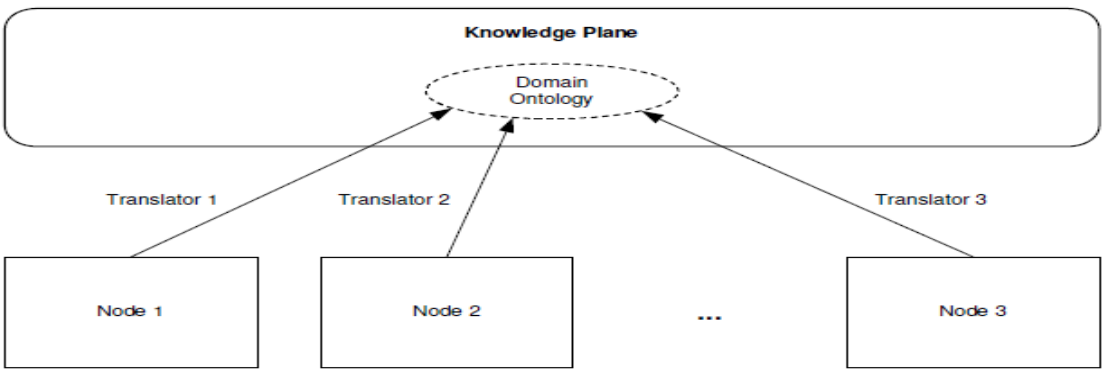
NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure D.1: RAT layer



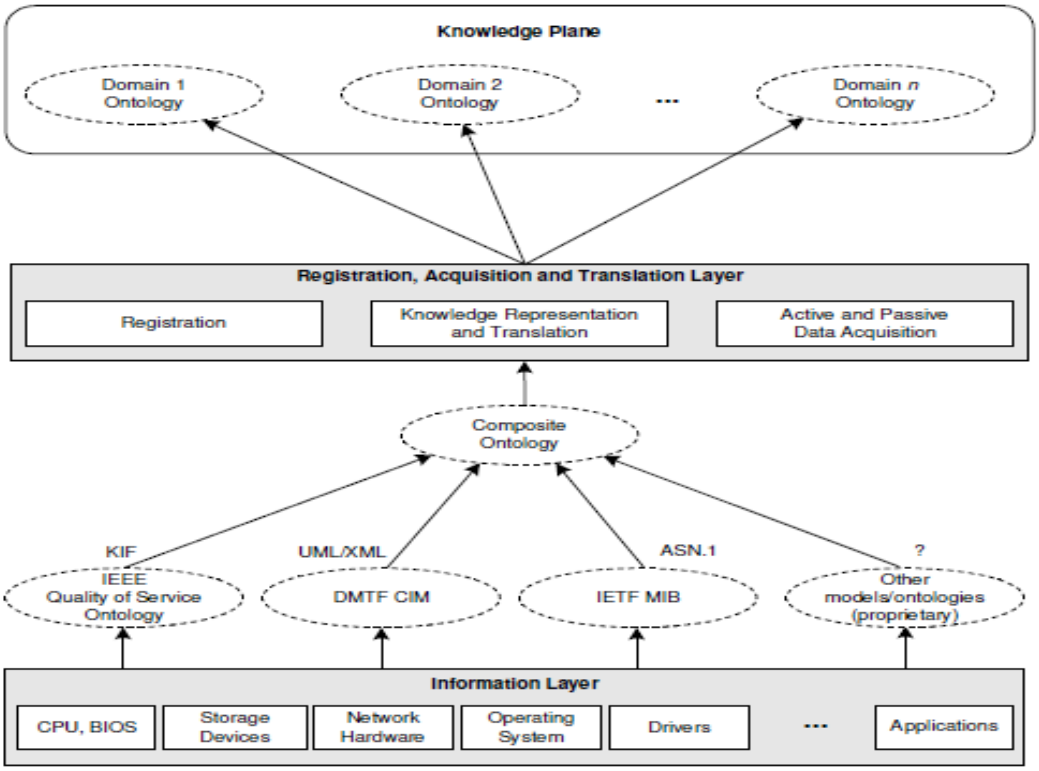
NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure D.2: Translations of domain ontologies



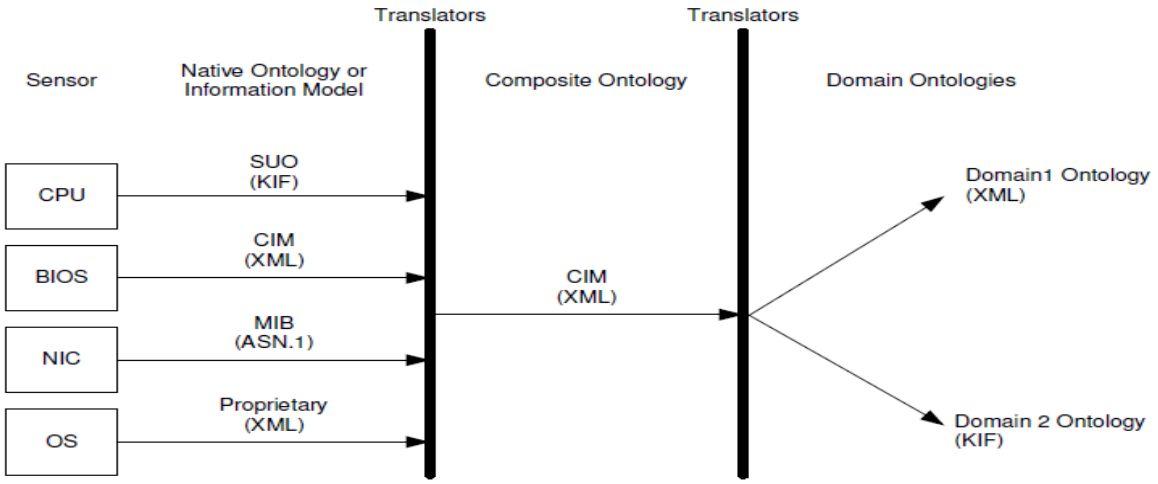
NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure D.3: Multiple translators for heterogeneous nodes



NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure D.4: Use of composite ontology for deriving domain ontologies



NOTE: This figure was extracted from the NIST Draft [i.36]: Copyright of the figure belongs to the authors of [i.36] (Stephen Quirolgico et al) and/or NIST (USA).

Figure D.5: Composite ontology translation

Annex E (informative): Bibliography

- Jennings, B., Brennan, R., van der Meer, S., Lewis, D. et al.: "Challenges for Federated, Autonomic Network Management I the Future Internet" ManFI workshop, New York, U.S.A. (5th June 2009), pp 87-92.
- Hans-Jörg Vögela, Benjamin Weyl, Stephan Eichler: "Federation solutions for inter- and intra-domain security in next-generation mobile service platforms", Int. J. Electron. Communications, (2006), pp 13-19.
- Wai-Khuen Cheng, Boon-Yaik Ooi, Huah-Yong Chan: "Resource federation in grid using automated intelligent agent negotiation", Future Generation Computer Systems 26, (2010), pp 1116-1126.
- Kevin Feeney, Rob Brennan, John Keeney, Hendrik Thomas, Dave Lewis, Aidan Boran, Declan O'Sullivan: "Enabling decentralised management through federation", Computer Networks, (2010).
- M. Serrano, S. van Der Meer, V. Holum, J. Murphy, and J. Strassner: "Federation, a Matter of Autonomic Management in the Future Internet", IEEE Network Operations and Management Symposium (NOMS), 2010, pp 845-849.
- Hongyong Zang, Yuzhong Sun, Kuiyan Gu: "Optimizing Inter-Domain Communication", Parallel and Distributed Systems (ICPADS), (2009), pp 355-360.
- Brennan R., Feeney K, Keeney J, O'Sullivan D, Fleck J.J. II, Foley S, Van der Meer S.: "Multidomain IT architectures for next-generation communications service providers", IEEE Communications Magazine 48, (2010), pp 110-117.
- Sajal K. Das, Arthur A. Reyes: "An approach to integrating HLA federations and genetic algorithms to support automatic design evaluation for multi-agent systems", Simulation Practice and Theory, Volume 9, Issues 3-5, (15 April 2002), pp 167-192.
- Wentong Cai, Guangya Li, Turner, S.J., Bu-Sung Lee, Li Liu: "Implementation of Federation Management Services over Federation Community Networks", Seventeenth Workshop on Parallel and Distributed Simulation, (2003), pp 50-57.
- Thomas Magedanz, Florian Schreiner, Sebastian Wahle: "Service-Oriented Testbed Infrastructures and Cross-Domain Federation for Future Internet Research", IM '09. IFIP/IEEE International Symposium, (2009), pp. 101-106.
- Govindan, R., Reddy, A.: "An Analysis of Internet Inter-Domain Topology and Route Stability", INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Vol. 2, (1997), pp. 850-857.
- K. I. Calvert, M. B. Doar, and E. W. Zegura: "Modelling Internet Topology", IEEE Communications Magazine, Vol. 35, No. 6, (June 1997), pp.160-163.
- Wakikawa R., Ohara, Y., Murai, J.: "Virtual Mobility Control Domain for Enhancements of Mobility Protocols", INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Vol. 4, (2005), pp. 2792-2797.
- Heeseok Lee, Jae Lee: "Analyzing Business Domain: A Methodology and Repository System", System Sciences, Vol. 3, (1998), pp. 409-419.
- G. Huston: "Analyzing the Internet's BGP routing table", The Internet Protocol Journal, (2001) -Citeseer.
- IETF RFC 3324 (November 2002): "Requirements for Network Asserted Identity", M. Watson.
- IETF RFC 3221 (May 2004): "Requirements for Inter Domain Routing", A. Doria, E. Davies, F. Kastenholz.

- Xuxian Jiang and Dongyan Xu: "VIOLIN: Virtual Internetworking on Overlay Infrastructure". In ISPA, pages 937-946, 2004.
- Mike Wawrzoniak, Larry Peterson and Timothy Roscoe: "Sophia: An Information Plane for Networked Systems". In HOTNETS-II. Intel, Nov 2003. IRB-TR-03-048.

History

Document history		
V1.1.1	April 2013	Publication