Auto-discovery and Auto-configuration of Routers in an Autonomic Network

Arun Prakash, Alexej Starschenko, and Ranganai Chaparadza

Fraunhofer FOKUS, Competence Center MOTION,
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
{arun.prakash,alexej.starschenko,ranganai.chaparadza}@fokus.fraunhofer.de

Abstract. The domain of Autonomics and Self-Managing networks come with a number of self-* features, such as auto-discovery and auto-configuration to name a few. In this paper, we provide a novel approach to auto-discover and auto/self-configure routers for OSPF routing in an autonomic network. We present the enablers for realizing these self-* functionalities. This includes a framework for describing the network policies, objectives and router configuration, models to be followed by a node/device implementation when self-describing the capabilities of a node and tokens for enforcing security and access control during the auto-discovery and auto-configuration processes of a node. We also present the algorithms that the various entities should employ for realizing these self-* features in their autonomic networks.

Keywords: autonomics, auto-discovery, auto/self-configuration, GANA, self-managing networks.

1 Introduction

The domain of Configuration Management has been well studied over the past years. A number of problems ranging from configuration-related problems, such as the impact of configuration-errors on security and operation of the network [1], to problems due to manual configurations, to problems related to network configuration management frameworks, and the limitations of their associated approaches and protocols are all well-documented [2,3,4]. Traditional network management and configuration protocols such as SNMP [5] and COPS (for policy configurations in Policy-Based Network Management (PBNM)) that still play a significant role in device and network configuration contribute their own set of problems [6]. Proprietary approaches to configuration management based on CLI (Command Line Interface), which are vendor-specific, are not free from problems either.

The domain of autonomic networking promises to alleviate the problems infesting the domain of configuration management. The auto-discovery and auto-configuration functionalities of self-* networks move ahead from the traditional scripting and automation techniques, to an advanced feedback-control based configuration m management. Several initiatives such as ANEMA [7], ANA [5], Self-NET [6], and **G**eneric **A**utonomic **N**etwork **A**rchitecture (**GANA**) [8,9] to name

R. Szabó et al. (Eds.): AccessNets 2010, LNICST 63, pp. 311–324, 2011.

a few, propose self-managing/autonomic architectures for network management and control. In this paper we focus on the auto-discovery and auto-configuration functionalities of GANA. The auto-discovery and auto-configuration functionalities of GANA are inbuilt with security features, and thus have explicit and far reaching influence on other autonomic functionalities in the network such as *autonomic-routing*, *self-organization*, etc.

The paper is structured as follows: In Section 1, the problems with state-of-the-art device and network configuration methods, and the need for advanced auto-discovery and auto-configuration techniques are discussed. Sections 2.1, 2.2, 2.3 showcase the enablers of auto-discovery and auto-configuration in GANA conformant networks. The algorithms to auto-discover and auto/self-configure routers for realizing OSPF [10] routing in an autonomic network are provided in Section 3. Finally, in Section 4, the conclusion, insights and future research directions are deliberated.

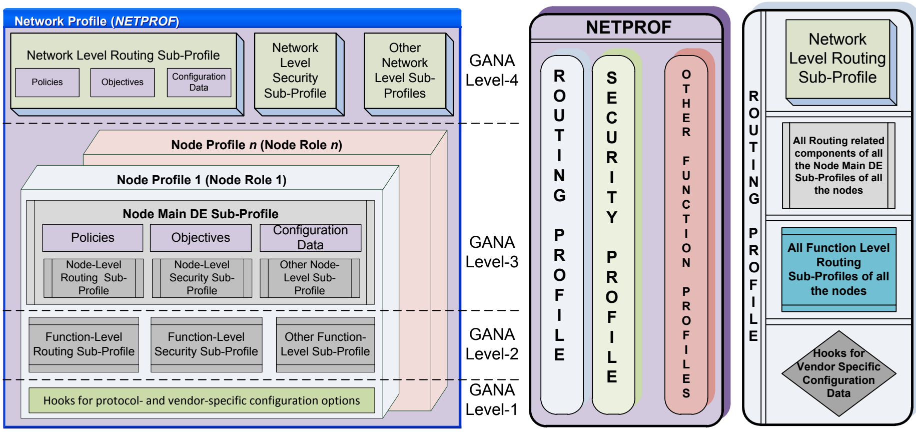
2 Auto-discovery and Auto-configuration Enablers

The auto-discovery and auto-configuration mechanisms in GANA require the concepts: *GANA Network Profiles*, *GANA Capability Description Model*, *GANA Tokens* and *ONIX* [11] as enablers for their complete range of functionality, potential and dynamism. These self-* enablers, with the exception of ONIX are discussed in here. **ONIX** (**O**verlay **N**etwork for **I**nformation e**X**change) [11] can be considered as a scalable, fault-tolerant and secured information and knowledge exchange system, with its own set of protocols and functions which facilitates seamless data and information distribution in an autonomic network.

2.1 GANA Network Profiles

A *Profile* is defined as a composition of *Policies*, *Functional Objectives* and *Configuration Data* required for defining overall goals and specific objectives for networking functions such as routing, security, etc of a network. A *Goal*, in the context of GANA, is defined as the overall target of the network as described by an network operator/administrator. Thus *Goals* delineate the *Policies*, *Objectives* and *Configuration Data* required for a network and its devices. In the context of GANA, the *Profile* is called a *GANA Network Profile* (**GANA NETPROF**) and is designed to provide:

- A structured and monolithic framework with a common data structure for specifying the policies, objectives and configuration data for an autonomic network and its nodes.
- A flexible framework for (re)configuring nodes, based on the dynamic roles they are computed to play in the network, and
- A mechanism to separate a node's role and functionality from its vendor specific configuration requirements.



(a) Vertical Decomposition **(b)** Horizontal Decomposition **(c)** Routing Profile

Fig. 1. *Network Profile* - NETPROF

A GANA NETPROF thus consists of a *Network Profile* (**NETPROF**), *GANA Configuration Options Map* (**MAP**) and several *Vendor Specific Node Configurations Options* (**CONFIG**). The NETPROF is composed of *Policies*, *Objectives*, *Configuration Data* and hooks for importing vendor specific CONFIG, and is structured along the GANA hierarchy [8,9] as shown in Figure 1a. At each GANA level, sub-profiles encapsulating the *Policies*, *Objectives* and *Configuration Data* for a GANA Decision-Element (DE) (≈ networking function) and its Managed Entities (MEs) are provided. The NETPROF can also be viewed to be composed along the abstracted functionalities of the GANA architecture, i.e., along functionalities such as *routing*, *security*, etc. Thus each of these functionalities can be considered to be contributing a profile of their own, as shown in Figure 1b. Thus, there is an inherent relationship between the GANA levels and abstracted functionalities as reflected Figure 1c.

The NETPROF is designed to accept different vendor CONFIGs for a node, as such an arrangement provides a number of advantages. The network operator/administrator can use existing configuration files without major changes, thus avoiding potential errors during the migration from their traditional networks to a GANA conformant network. They can continue to use their configuration files that are vendor specific for the nodes/devices. Finally, the *hooks* in the NETPROF do not confine a node *role* to be vendor specific at design time, allowing dynamic role switching and re-configuration 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 of a parameter such as *Area ID*, in the case of OSPF [10] routing is dynamic, as the network gets autonomically partitioned and merged into new OSPF areas due to failures, new routing nodes being added and changing network conditions. The problem arises when such configuration parameters are expressed in different vendor specific semantics for their devices. In order to enable a vendor-free implementation of a specific DE, we provide a solution with the MAP. The MAP is a tabular structure that maps configuration parameters used in GANA in their standard form (e.g. IETF RFCs) to vendor specific formats. The MAP holds the semantics of both name and value of a configuration parameter. It is used by the *Network-Level Routing-Management DE* (**NET_LEVEL_RM_DE**) [8,9] for the generation of the *GANA Node Configurations* (**NODECONF**) from the NETPROF. The NODECONF is used by a node for its configuration. The GANA NETPROF is formalized through the well known industry de facto XML standard. XML provides a formal and standardized approach to the design and engineering of GANA NETPROFs. The use of GANA NETPROFs for network governance is fully described in [12].

2.2 GANA Capability Description Model

A self-managing network needs a way to know the entities composing the network and the *Capabilities* of individual functional entities of the nodes in the network. The auto-discovery functionality of the network should gratify this need. In GANA, self-description, self-advertisement, topology-discovery and support for solicitation of *Capabilities* belong to the auto-discovery functions of a node.

Self-Description of *Capabilities.* is the ability of a functional entity to describe itself, i.e., to describe its *Capabilities* such as hardware and software specifications, supported protocols, services and tools, interface information, etc, its *current role* and the possible *potential roles* it can play in the network. These compose the *GANA Capability Description Model* of a functional entity. The *Capability Description* is formalized through the industry de facto XML standard.

Self-Advertisement of *Capabilities.* is the process by which a functional entity spontaneously disseminates its *Capability Description* to other functional entities either inside a node or in the network. The dissemination may be done over a distributed repositories such as ONIX [11].

Support for solicitation for *Capabilities.* is ability of a functional entity to respond to requests for its *Capability Description* by initiating its self-description and self-advertisement functions. This is vital for the self-organization functionality of a network.

In GANA, the auto-discovery mechanism is initiated by the Node-Main DE (**NODE_MAIN_DE**) [9,8] of a node. The NODE_MAIN_DE generates the *Capability Description* of the node by triggering the iterative self-description process as shown in Figure 2. The *Capabilities* of individual DEs and its MEs are obtained in a recursive manner. The aggregated *Capabilities* of the node are then advertised to the network, thus completing the auto-discovery process of a node.

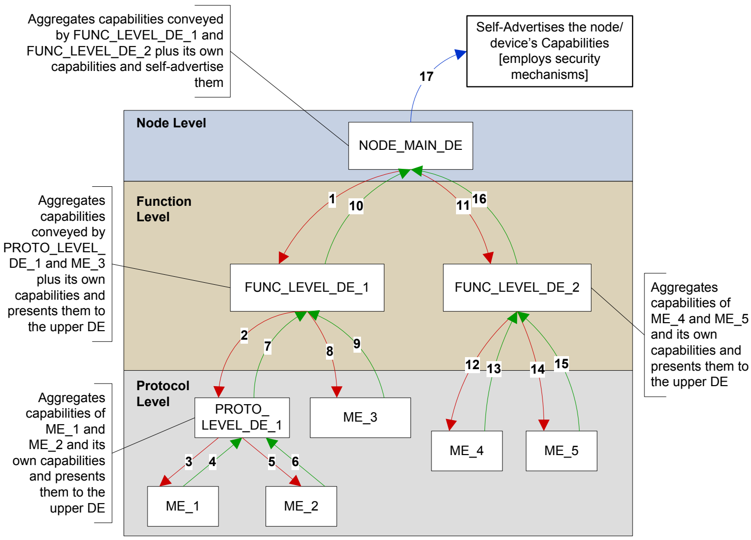


Fig. 2. Auto-Discovery Functionality in a GANA Conformant Node

Table 1. GANA Tokens Types and Token Information - Abridged Example

	Token Type	Type Description
	0	Entity Role Token
Info	0	Network Operator/Administrator
	1	NET_LEVEL_SEC_MNGT_DE
	2	Network-Level DE
	3	GANA Node
	1	Permitted ONIX Operations Token
Info	0	On-Behalf Subscribe
	1	On-Behalf Update
	2	ONIX Information Access Token
Info	0	Network Policy
	1	Monitoring Data

2.3 GANA Tokens

A *GANA Token* can be considered as an object that holds the token type and token information. Three different types of *GANA Tokens* have been defined,

– **Entity Role Token:** This identifies the token holder as a certain type of network entity / network role.
– **Permitted ONIX Operations Token:** This allows the token holder to have privileged ONIX related operations.
– **ONIX Information Access Token:** This allows the token holder with privileged access to information and data stored in ONIX.

A *GANA Token* is always encapsulated by a security key issued by the *Network-Level Security-Management DE* (**NET_LEVEL_SM_DE**) [8,9]. A key can have any number of ONIX related tokens, but only one *Entity Role Token*. The *Token Information* (Info) can be configured manually by the network operator/administrator or can be dynamically set by the NET_LEVEL_SM_DE. The NET_LEVEL_SM_DE issues *GANA Tokens* to the nodes to be used during the above mentioned operations in the network. A brief illustration of GANA Tokens is given in Table 1.

3 Auto-discovery and Auto-configuration Algorithms

The auto-discovery and auto-configuration algorithms of a GANA Node and the NET_LEVEL_RM_DE are discussed here. The algorithms focus on the mechanisms for realizing OSPF [10] routing in an autonomic network.

3.1 Network-Level Routing-Management Decision-Element

The NET_LEVEL_RM_DE [8,9] belongs to Level-4 of the GANA control-loop hierarchy. Thus it has the overall view on the routing functionality of the network, required for the configuration of the routers. In the context of this paper, the DE supports the network operator/administrator by assisting in network topology design and area-partitioning for OSPF routing during the network topology design phase. Further, autonomic network topology discovery and generation of router NODECONF with respect to the network design and goals are carried out at runtime without any manual intervention. Additionally, the DE triggers the reconfiguration of the routers to adapt to changing routing goals and dynamic network conditions. These functionalities of the NET_LEVEL_RM_DE are realized through a number of modules and functions, as discussed below.

Partitioning Parameters (*Partitioning_Parameters*) influence the partitioning of a network topology into OSPF areas and are provided by the operator through the *Desired_Topology* element of the NETPROF. If no parameters are explicitly set, default values are used for generation of OSPF areas. The parameters of interest are:

- *Maximum/minimum number of routers in an area* - It determines the size of an area during network design and partitioning.
- *Maximum/minimum number of areas* - It specifies the number of areas during network design an partitioning.
- *Threshold* - It expresses the maximum size of an area permissible for OSPF routing.

OSPF Desired Topology Partitioning Module partitions the network topology designed (desired) by the network operator/administrator into OSPF areas with respect to constraints imposed by the partitioning parameters. The behavior of the module is specified in **Algorithm 1**. The outcome of the

Algorithm 1. OSPF Desired Topology Partitioning

Require: UPDATES of *Desired_Topology*, *Planned_Topology* and *Partitioning_Parameters*
Ensure: *Planned_Topology* computed according to *Partitioning_Parameters*

```
loop
  if Desired_Topology available then
    if not Planned_Topology generated from Desired_Topology then
      if not Partitioning_Parameters published then
        Partitioning_Parameters = DEFAULT
      Planned_Topology = OSPF_Area_Partitioning(Desired_Topology, Partitioning_Parameters)
      if Partitioning FAILED then
        REPORT FAILURE
      else
        PUBLISH Planned_Topology
  LISTEN Desired_Topology updates
```

partitioning process (*Planned_Topology*) is published to ONIX as an update to NETPROF. The *Planned_Topology* is applied to the network by the *OSPF Actual Topology Partitioning and Configuration Module*. Additionally the operator may view and modify the *Planned_Topology* with new data.

OSPF Actual Topology Partitioning and Configuration Module handles a number of tasks relevant to the actual network, such as:

1. To apply the *Planned_Topology* (if one is present) when it matches with the actual network topology.
2. To partition the network, each time the number of routers in an OSPF area reaches the set threshold.
3. To compute and distribute NODECONFs that contain the required routing related configuration for each node.

Algorithm 2 reflects the behavior of this module. The *Actual_Topology* encapsulates the current network graph, i.e, the nodes and their interconnections. It is required in order to compute the *point-of-attachment* of the routers in the network, and thus determine their network roles and corresponding router NODECONFs. It may be generated by a topology-discovery function using IPv6's Neighbor-Discovery (ND) protocol, wherein each router publishes the state of its neighbors to the NET_LEVEL_RM_DE facilitating the computation of the IP-layer topology. The generated *Actual_Topology* is published into ONIX as a update to the NETPROF.

The NETPROF contains several sub-profiles (Node Profiles) for various node roles. For OSPF routing, the node roles are classified as: *Core Router* (CR), *Area Border Router* (ABR) and *Autonomous System Border Router* (ASBR). Using the *Area_Assignment* computed by the partitioning function, *Capability Description* and the *current role* of the node, the role of each router in the

Algorithm 2. OSPF Actual Topology Configuration and Partitioning

Require: UPDATES of *Desired_Topology*, *Planned_Topology*, *Actual_Topology*, *Capability_Descriptions*

Ensure: $MAX(AREA\_SIZE(Actual\_Topology)) <= THRESHOLD$, *Planned_Topology* applied to network, routers configured

```
if not Partitioning_Parameters published then
  Partitioning_Parameters = DEFAULT
loop
  if Actual_Topology changed then
    if Planned_Topology and Actual_Topology match then
      for all Router in Actual_Topology do
        COMPUTE NodeConf using Planned_Topology
        PUBLISH NodeConf to ONIX
        SUBSCRIBE Router to receive NodeConf
        WAIT for ACK from Router
    else
      if SIZE(Actual_Topology) > SIZE(Planned_Topology) then
        Partitioning_Parameters = DEFAULT
      if MAX(AREA_SIZE(Actual_Topology)) > THRESHOLD then
        Topology = OSPF_Area_Partitioning(Actual_Topology,
        Partitioning_Parameters)
        if Partitioning FAILED then
          Partitioning_Parameters = DEFAULT
          Topology = OSPF_Area_Partitioning(Actual_Topology,
          Partitioning_Parameters)
          if Partitioning FAILED then
            REPORT FAILURE
        for all Router in Actual_Topology do
          COMPUTE NodeConf using Topology
          PUBLISH NodeConf to ONIX
          SUBSCRIBE Router to receive NodeConf
          WAIT for ACK from Router
      else
        COMPUTE NodeConf of new router using Capability_Description and
        Actual_Topology
        PUBLISH NodeConf to ONIX
        SUBSCRIBE Router to receive NodeConf
        WAIT for ACK from Router
  LISTEN Actual_Topology updates
```

network is computed. The appropriate node sub-profile of the NETPROF is chosen for the generation of the NODECONF, as shown in **Algorithm 3**.

OSPF Area Partitioning Module contains functions to partition a network into OSPF areas and is used by both the modules specified in sections above. It uses a modified version of the *DDOA-Algorithm* [13], adapted to fit with the GANA requirements. The behavior is realized through three separate functions as depicted in **Algorithm 4**.

Algorithm 3. GANA NODECONF Computation

Require: *Capability_Description* of *Router*, *Configuration_Options*, *Configuration_Options_Map*, *Topology*, *Area_Assignment*, *NetProf*
Ensure: *NodeConf* for *Router* computed

```
Router_Vendor = read vendor from Capability_Description
RECEIVE Configuraion_Options for Router_Vendor from ONIX
READ RouterID from Capability_Description
SET RouterID in Configuration_Options using Configuration_Options_Map
for all Interface of Router in Topology do
    COMPUTE AreaID for Interface using Topology and Area_Assignment
    SET AreaID for Interface in Configuraion_Options using
    Configuration_Options_Map
NodeRole = role of router in AreaAssignment and Toplogy
GET NodeConf element for NodeRole from NetProf
SET Policies,Objectives,Configuration for FUNC_LEVEL_RM_DE in NodeConf
PASTE Configuration_Options for Router_Vendor into NodeConf
if NodeConf already published on ONIX then
    NodeConf_Current =RECEIVE NodeConf
    REPLACE classified under Routing with the elements from NodeConf
    RETURN NodeConf_Current
else
    RETURN NodeConf
```

Node-Grouping Function applies the *node grouping* constraints (if any) specified in the *Partitioning_Parameters*. The constraint ensures that the grouped nodes remain in the same OSPF area after the partitioning. Nodes are grouped by replacing them with a single node in the network topology graph. The weight of this single node is the aggregate weight of the group. After partitioning, the area-assignment of the representative node is applied to the entire group.

Graph-Partitioning Function partitions a network-graph into interconnected partitions under the constraints imposed by the *Partitioning_Parameters*. For graph partitioning the *Chaco* [14] partitioning tool is used. The *number of desired areas* (computed from *Partitioning_Parameters*), the weighted network topology graph, and optionally the geographical coordinates of the vertices (nodes/routers) serve as inputs. The graph is partitioned into the desired number of areas while balancing the number of vertices in each area (obtained from *Partitioning_Parameters*). Each vertex is then assigned an area-number based on the area it belong. The output (*Area_Assignment*) cannot be directly used for OSPF area assignment as *Chaco* does not guarantee interconnected areas. This function extends *Chaco* by reassigning disconnected areas with new area numbers.

After graph partitioning, the function checks whether the resulting graph satisfies the partitioning requirements, namely *minimum size of an area*, *maximum number of areas*, etc, enclosed within *Partitioning_Parameters*. The function returns the computed area assignment if the requirements are satisfied. If the *number of areas* is beyond the *maximum*, the function tries to merge some partitions without violating the *minimum number of areas* and the *maximum number of nodes per area* constraints. If the merging is successful, the new area-assignment is returned, else a failure is indicated.

Algorithm 4. OSPF Area Partitioning Module

```
Require: Topology, Partitioning_Parameters
Ensure: Area_Assignment for routers that fulfills partitioning-constraints
Graph = graph-representation of Topology
Topology = Node_Grouping(Graph, Partitioning_Parameters)
N_Areas = minimum number of areas
while N_Areas <= maximum number of areas do
  Area_Assignment = Network_Partition(Graph, Partitioning_Parameters,
  N_Areas)
  if SUCCEED Partitioning then
    Area_Assignment = Area0_Design(Graph, Partitioning_Parameters,
    Area_Assignment)
    if SUCCEED Area0-Design then
      Area_Assignment = Area_Assignment "+" Area_Assignment for grouped
      nodes
      return Area_Assignment
  N_Areas = N_Areas++
return F AILURE
```

Area0-Design Function computes the OSPF ABR candidates and builds the OSPF *Area-0* for the partitioned network-graph. The inputs to this function are the network-graph, partitioning-parameters and area-assignment. The output of this function is the *Area_Assignment* with *Area-0* defined. Nodes that are connected to each other across partition boundaries are chosen as the *ABR-Candidates* for the network. *Area-0* is computed by choosing a predefined number ($default = 1$) of nodes from each area' ABR candidate-list. If the network operator/administrator choose some nodes as preferred ABRs, they are chosen first. Otherwise the candidates with the highest sum of edge-weights are chosen. The chosen ABRs are used for constructing *Area-0*. If *Area-0* is discontinuous, more candidate ABRs are added or virtual links [10] are used to construct a connected *Area-0*. If this operation succeeds, the resulting area0-assignment for the selected ABR candidates is saved. Other possible combination of ABR candidates are tried, and the most *optimal solution* (minimal, robust, etc) is chosen. If no solutions are found, a failure is indicated.

3.2 Node Main Decision-Element

The NODE_MAIN_DE [8,9] is the top element in the DE hierarchy of a GANA node. It provides several functions and manages the node as a whole. The auto-discovery and auto-configuration functionalities of a node are managed and orchestrated by the NODE_MAIN_DE. They are discussed below.

Security. In GANA, some security aspects are inbuilt into the auto-discovery and auto-configuration functionalities of the node and the network. *GANA Tokens* (see Section 2.3) are used for authentication of the node and access control of ONIX information. All communication between the nodes, Network-Level DEs and ONIX make use of *GANA Tokens* for network security. The *GANA Tokens* are provided by the NET_LEVEL_SM_DE [8,9].

Node Bootstrap. When the node/router is turned on by the network operator/administrator, the NODE_MAIN_DE is invoked and performs the initialization and orchestration of node parameters and DEs. For the initialization and orchestration operations, default/initial DE and protocol configurations are used. During the bootstrap phase the NODE_MAIN_DE also triggers the address-auto-configuration of the interfaces of the node/router. For the address-auto-configuration the Autonomic DHCP Architecture (ADA) [15] is employed. ADA enables the address auto-configuration of node/router interfaces by providing a set of extensions to DHCP to primarily support interface bootstrapping and zero-conf DHCP relaying [15].

Auto-Discovery. The *Self-Description* process of the node involves the generation of *GANA Capability Description Model*, a concept introduced in Section 2.2. The computation of *Capabilities* of the DEs and its underlying MEs is an iterative process, as shown in Figure 2, triggered by the NODE_MAIN_DE. The aggregated *Capability Description* is augmented with the node attributes such as *class-of-device* and hardware information. The *Self-Advertisement* process of a node involves the publication of the unified *Capability Description Model* into ONIX and to on-link nodes subject to security policies. The self-description and self-advertisement functions are repeated when the *Capabilities* of the device changes.

As described in Section 3.1, neighbor information of each device is required by the NET_LEVEL_RM_DE for topology-discovery. This information is provided by the NODE_MAIN_DE by publishing and updating a list of on-link routers on each interface to the NET_LEVEL_RM_DE. **Algorithm 5** provides the algorithm for *Node Bootstrap* and *Auto-Discovery*.

Auto-Configuration. As described in Section 2.1, the NODECONF computed by the NET_LEVEL_RM_DE and distributed through ONIX, is used in the auto-configuration of the node. Every time a node receives a NODECONF, its NODE_MAIN_DE parses it, and configures itself and Function-Level DEs. Further, each DE also receives the sub-profiles containing the configuration data for their MEs from the NODE_MAIN_DE. This is reflected in **Algorithm 6**.

3.3 Function-Level Routing-Management Decision-Element

The *Function-Level Routing-Management DE* (**FUNC_LEVEL_RM_DE**) [8,9] is responsible for the configuration and management of routing protocols and mechanisms inside a GANA node/router. The functionalities in the context of the paper include:

Algorithm 5. NODE_MAIN_DE - Auto-Discovery

```
Require: GANA_Token
Ensure: Auto-Discovery, Security
for all DEs on Node- and Function-Level do
   START DE
DISCOVER NET_LEVEL_SEC_MNGT_DE
REQUEST GANA_Token
RECEIVE GANA_Token
DISCOVER ONIX
Capability_Description "=" NODE_MAIN_DE capabilities
Capability_Description "+=" device capabilities
for all MEs of NODE_MAIN_DE do
   Capability_Description "+=" ME capabilities
WAIT for global IP-address auto-configuration
Capability_Description "+=" global address
PUBLISH Capability_Description on ONIX
COMPUTE Neighbour_List using the IPv6 Neighbour Discovery Mechanism
DISCOVER NET_LEVEL_RM_DE
PUBLISH Neighbour_List to NET_LEVEL_RM_DE
loop
   LISTEN Event
   if Event == UPDATE of ME capabilities then
      Capability_Description "+=" ME capabilities
      PUBLISH update of Capability_Description on ONIX
   else if Event == UPDATE Neighbour_List then
      PUBLISH Neighbour_List to NET_LEVEL_RM_DE
```

Algorithm 6. NODE_MAIN_DE - Auto-Configuration

```
Require: NodeConf
Ensure: Auto-Configuration of DEs and protocols
loop
   LISTEN Event
   if Event == RECEIVE NodeConf then
      for all DE on Node- and Function-Level do
         PUSH DE_Config derived from NodeConf to DE
         PUSH Protocol_Config derived from NodeConf to DE for DEs function
```

1. Publication of the *Capabilities* of itself and its MEs (routing protocols and mechanisms such as OSPFv3) when requested by the NODE_MAIN_DE.
2. (Re-)Configuration of the MEs according to the configuration provided by the NODE_MAIN_DE, and its acknowledgment and validation.

The *Capabilities* of the MEs contain information such as protocol/ME version, supported protocol features and services, computational cost of running the protocol, etc. As described in Section 2.1. the configuration of MEs in GANA is vendor specific. Thus the DE uses the MAP defined in Section 2.1 to understand the semantics and syntax of the configuration parameters for the various vendor specific implementations of OSFPv3. The management interface (CLI or SNMP [5]) is used by the DE to configure OSPF. The configuration is validated by checking the value of the various parameters of OSPF with the configuration parameters provided in the NODECONF. Along with the NODE_MAIN_DE, the FUNC_LEVEL_RM_DE ensures that the configurations for OSPF (routing protocol) computed by the NET_LEVEL_RM_DE and distributed through ONIX, are applied by the individual routers, completing the auto-configuration process.

4 Conclusion

In this paper, we presented the enablers and the algorithms required for realizing the auto-discovery and auto-configuration features in a GANA conformant network. The approach outlined is realistic and moves ahead from the traditional scripting and automation techniques used for configuration management. Further, the profound issue of network security, often sidelined during a discourse on configuration management, is given paramount importance. Thus several security issues, such as authentication, access control and trust are considered during the design of the enablers and algorithms. In conclusion from our ongoing implementation, we believe that the approach provided here is pragmatic and effective.

In the future, we aim to replace the MAP with a vendor configuration Ontology that captures the semantics and grammar of the configuration parameters used by different vendors. Every vendor conforming to GANA standards may provide their vendor specific configuration ontology which would be integrated with the ontology designed as part of the GANA oriented design methodology [16]. The unified ontology would enable the Network-Level DEs to manipulate complex configuration parameters, of different vendor specific MEs providing the same functionality, in a dynamic fashion. Further, we also intend to extend the current algorithms to auto-configure other routing protocols, such as RIP, BGP etc. Finally, a case study evaluating the proposed algorithms would be completed and published.

Acknowledgments. This work is partially supported by EC FP7 EFIPSANS project (INFSO-ICT-215549) [17].

References

1. Tayal, A.P., Patnaik, L.M.: An Address Assignment for the Automatic Configuration of Mobile Ad Hoc Networks. Personal Ubiquitous Comput. 8(1), 47–54 (2004)
2. Sanchez, L., McCloghrie, K., Saperia, J.: Requirements for Configuration Management of IP-based Networks. RFC 3139 (Informational) (June 2001), `http://www.ietf.org/rfc/rfc3139.txt`

3. Yoo, S.M., Ju, H.T., Hong, J.: Web Services Based Configuration Management for IP Network Devices. In: Dalmau Royo, J., Hasegawa, G. (eds.) MMNS 2005. LNCS, vol. 3754, pp. 254–265. Springer, Heidelberg (2005), http://dx.doi.org/10.1007/11572831_22
4. Choi, H.M., Choi, M.J., Hong, J.: Design and Implementation of XML-based Configuration Management System for Distributed Systems. In: IEEE/IFIP NOMS 2004: Proc. of Network Operations and Management Symposium, vol. 1, pp. 831–844 (2004)
5. Harrington, D., Presuhn, R., Wijnen, B.: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard) (December 2002), http://www.ietf.org/rfc/rfc3411.txt; updated by RFCs 5343, 5590
6. Chatzimisios, P.: Security Issues and Vuluerabilities of the SNMP Protocol. In: ICEEE 2004: Proc. of 1st International Conference on Electrical and Electronics Engineering, June 2004, pp. 74–77 (2004)
7. Derbel, H., Agoulmine, N., Salaün, M.: ANEMA: Autonomic network management architecture to support self-configuration and self-optimization in IP networks. Comput. Netw. 53(3), 418–430 (2009)
8. Chaparadza, R.: Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments. International Engineering Consortium (IEC), Annual Review of Communications 61 (2008)
9. Chaparadza, R., et al.: Towards the Future Internet - A European Research Perspective. In: Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering, pp. 313–324. IOS Press, Amsterdam (2009)
10. Coltun, R., Ferguson, D., Moy, J., Lindem, A.: OSPF for IPv6. RFC 5340 (Proposed Standard) (July 2008), http://www.ietf.org/rfc/rfc5340.txt
11. Chaparadza, R., et al.: IPv6 and Extended IPv6 (IPv6++) Features that enable Autonomic Network Setup and Operation. In: SELFMAGICNETS 2010: Proceedings of the International Workshop on Autonomic Networking and Self-Management in the Access Networks. ICST ACCESSNETS 2010 (November 2010)
12. Lozano, J.A., Gonzalez, J.M., Chaparadza, R., Vigouraux, M.: Engineering Future Network Governance. Journal of ICT Future Technologies 36(204), 23–30 (2010)
13. Galli, S., et al.: A Novel Approach to OSPF-area Design for Large Wireless ad-hoc Networks. In: ICC 2005: Proc. of 2005 IEEE International Conference on Communications, May 2005, vol. 5, pp. 2986–2992 (2005)
14. Hendrickson, B., Lelandy, R.: The Chaco User's Guide Version 2.0. Tech Report SAND95-2344, Sandia National Laboratories, Albuquerque, NM 87185-1110 (July 1995)
15. Németh, F., Rétvári, G.: The Autonomic DHCP architecture for IPv6 (September 2010), http://qosip.tmit.bme.hu/~retvari/autonomic_dhcp.html
16. Prakash, A., Chaparadza, R., Theisz, Z.: Requirements of a Model-Driven Methodology and Tool-Chain for the Design and Verification of Hierarchical Controllers of an Autonomic Network. In: CTRQ 2010: Proc. of International Conference on Communication Theory, Reliability, and Quality of Service, June 2010, vol. 0, pp. 208–213. IEEE Computer Society, Los Alamitos (2010), ISBN: 978-0-7695-4070-2
17. EC FP7-IP EFIPSANS Project (2008-2010), http://www.efipsans.org, INFSO-ICT-215549