Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 20, 2012

R. Chaparadza
R. Petre
Fraunhofer Fokus
H. Mahkonen
Ericsson
L. Xin
BUPT
M. Behringer
Cisco Systems

July 19, 2011

IP based Generic Control Protocol (IGCP)
draft-chaparadza-intarea-igcp-00.txt

Abstract

This document presents a proposal for a multi-purpose Generic Control Protocol (IGCP) for IP based networks. There is a growing need for a generic control protocol framework that can be further customized to specific usage contexts in which certain types of control information exchange messages and behavior among some functional entities hosted by different nodes or devices is desired. For example, the growing area of self-management, self-organization and autonomic networking introduces functional entities into the node/device and network architectures that need to exchange control information in order to implement self-adaptive behavior by dynamically configuring and optimizing the network. In this Draft we capture a number of control message exchange types of contexts (semantics) that can be selectively applied in the exchange of control information, which can form the basis of a generic control protocol, while at the same time defining the part in the message format that can be further customized according to the needs of specific functional entities designed to use the generic control protocol for exchanging control information. In this Draft, we present our proposal for such a generic control protocol, whose message format is divided into two parts: a Common Part and a Generic Data Part. The Common Part defines a set of a variety of selectable control semantics (e.g. simple one-way control information flow, indications of whether an acknowledgement is needed or not, solicitations for information or push/pull behaviors, negotiations for parameter value settings, etc). The Generic Data Part can be further customized and structured according to some specific use case of conveying control information carried by the Data Part that need to be parsed and used by some entities designed to interpret the Data Part according to their own specific customization and structuring of the Data Part. We also give an example domain of application of the IGCP, namely the domain of autonomic adaptive control of network behaviours, of which we

illustrate further by providing an example Use Case that customizes the Data Part of the IGCP for use by special functional entities residing in different nodes in exchanging information using the IGCP messages.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 20, 2012.

Copyright Notice

Table of Contents

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

The Draft proposes a Generic Control Protocol Framework that is customizable for different types of usage contexts (use cases) in which certain types of control information exchange messages and control behavioral semantics among the involved Functional Entities hosted by different nodes/devices are required. Being "Generic" means having the following properties:

1. Having a Common Part i.e. a Header in the Message Format that defines a set of a variety of control semantics (e.g. simple one-way control information flow, indications of whether an acknowledgement is needed or not, solicitations for information or push/pull behaviors, negotiations for parameter value settings, etc). The control semantics are meant to be diverse and selectable by functional entities exchanging control messages.

2. Having a Generic Data Part as payload part of the Message Format that can be customized and further structured according to the identified requirements for control communication between two or more types of Functional Entities that need to parse and interpret the Data Part in their own way customized for them by design.

3. Problem Statement

The trend in designing protocols for control information exchange between two or more functional entities residing in different nodes/devices in the network has often resulted in control protocols that are specific only to the needs of the functional entities involved. Multiple control protocols exist today and yet share very similar control semantics and in most cases they are difficult to extend or apply for other requirements for control semantics introduced by the evolving networking paradigms. Different types of a variety of control semantics are often used to design a control protocol, such as (1) simple one-way control information flow with or without requiring acknowledgement of receipt of information by the receiver; (2) solicitations for information or push/pull behaviors by the functional entities involved; (3) negotiations by the entities involved in the process of performing parameter value settings, etc). Some functional entities in network nodes/devices need to exchange control information in order to implement adaptive network behavior and dynamic network configuration and optimization that is co-ordinated by the functional entities collaboratively. Such functional entities can be an application, protocol or some other type of an entity that communicates with other peer entities hosted by other nodes/devices.

In the emerging networking paradigms, the growing area of self-management, self-organization and autonomic networking introduces functional entities into the node/device and overall network architectures that need to exchange control information in order to implement adaptive network behavior and dynamic network configuration and optimization that is co-ordinated by the functional entities collaboratively. Such functional entities can be, for example, distributed management components playing the role of "micro-manager elements" embedded in two or more nodes/devices, which co-operatively work together to realize distributed management and adaptive control of resources such as protocols, stacks and mechanisms. This because some degree of intelligence can be introduced or enhanced within a network element e.g. a router by introducing embedded "micro-manager elements" that monitor events reported by local entities such as protocols (for example by accessing a management MIB(s)) and react by adaptively provisioning resources or regulating the behaviour of the managed protocols in order to fulfill some objective (e.g. reducing the amount of control traffic transmitted by the network element). This means the "micro-manager elements" configure, apply policies, monitor and dynamically adapt the behaviour of the Managed Entities (MEs) such as protocols, stacks and mechanisms. The "micro-manager elements" can be developed to operate at a level of abstraction that is above protocol stacks and mechanisms of a device, since such a level of abstraction does not necessarily require changes to existing

protocols. Such micro-manager elements can be perceived as "Decision-making-Elements" controlling (regulating) or managing (in general sense) some aspects of a node/device and protocols since the control-plane is viewed as a kind of horizontal extension of the management plane within the network itself, outside of the notion of the vertical management plane that involves management systems. The "micro-manager elements" may require the ability to perform the following operations: (a) Negotiating the way their assigned Managed Entities (MEs) e.g. Protocols, Stacks and mechanisms of the node/ device, should be configured or dynamically re-configured. The need to negotiate configuration settings e.g. parameter value settings applies to the case where there is no need to involve a centralized coordinating entity (e.g. Network Management System), and so the peer "manager elements" in nodes/devices need to self-organize certain aspects of the network e.g. the way some protocols must behave according to some policies. (b) Soliciting for Capabilities of the peer manager element or multiple peers based on the features supported by their associated MEs(i.e. capabilities of the managed protocols, stacks and mechanisms--managed resources). (c) Self-Advertising Capabilities of functional entities such as a node/device as a whole to peer "manager elements" hosted by devices on the link, or to selected peers by policy. (d) Exchanging Trust related information/data. (e) Exposing "Views" to peer "manager elements", such as detected incidents, misbehaviors, etc, which can be used by the peers for adaptive (re)-configuration of their associated MEs. (f) Requesting for "Views" from peer "manager elements".

A Generic Control Protocol Framework that is extensible, and defines a set of a variety of control semantics that can be selectively used by functional entities intending to communicate or exchange control information, if defined for IP networks, would serve as a "multi-purpose" Generic Control Protocol in IP based networks and would cost-effectively ease the development of adaptive behaviours of diverse types of functional entities. An example of a control protocol that exists today that enables nodes/devices in IP networks to communicate to each other some control information and is extensible is ICMPv6. A number of proposals have emerged recently, regarding information sharing between nodes/devices. For example in [RFC4620] two messages are defined: Node Information Query and Node Information Reply, but they are designed to be used only for discovering information about names and addresses. ICMPv6 messages are subdivided into two classes: Error messages and Information messages. ICMPv6 (like a number of IPv6 protocols) offers some room for Extensibility depending on the need for exchanging Control Related Information and/or Use Case Context in which some Extension is required to the base ICMPv6 protocol. Therefore, ICMPv6 offers a base for developing a Generic "multi-purpose" Control Protocol as an Extension to ICMPv6. But, de-coupling the generic control protocol

from being an explicit extension to ICMPv6, means the generic multi-purpose control protocol could be developed as a standalone protocol that can be used in both IPv4 and IPv6.

Proposal in brief (more details are provided in the next sections): An IP based Generic Control Protocol(IGCP) Framework that is customizable to specific usage contexts in which certain types of control information exchange messages and control behavioral semantics among some Functional Entities hosted by different nodes is desired. Being "Generic" means having the following properties: 1. Having a Common Part i.e. a Header in the Message Format that defines a set of a variety of control semantics (e.g. simple one-way control information flow, indications of whether an acknowledgement is needed or not, solicitations for information or push/pull behaviors, negotiations for parameter value settings, etc). The control semantics are meant to be diverse and selectable by functional entities exchanging control messages. 2. Having a Generic Data Part as payload part of the Message Format that can be customized and further structured according to the identified requirements for control communication between two or more types of Functional Entities that need to parse and interpret the Data Part in their own way customized for them by design. The Data field must be left to be customized and defined according to the specific needs of specific types of functional entities that need to exchange control information.

Later, at the end of the Draft, we also give an example domain of application of the IGCP, namely the domain of autonomic adaptive control of network behaviours, of which we illustrate further by providing an example Use Case that customizes the Data Part of the IGCP for use by special functional entities residing in different nodes in exchanging information using the IGCP messages.

4. The Proposed Solution (IGCP: IP-based Generic Control Protocol) and its application (Usage Scenarios)

For all the requirements for control information exchange listed in the problem statement, we propose to introduce some generic IGCP messages that can be used by different types of functional entities requiring communicating with each other across nodes/devices. The IGCP header consists of the first part that can be used for different purposes as allowed by the fields defined later in this draft, and a Data part. The Data part can be further structured according to the identified requirements of different types of functional entities that need to exchange control information. We propose that the Data field be left to be customized and defined according to the specific needs of specific types of functional entities that need to exchange control information. An example of how the Data field can be customized for a particular Use Case Scenario of the IGCP is presented later.

4.1. Generating an IGCP message/packet

Entities generating an IGCP packet (e.g. DEs) should reason about:

1. When to generate the IGCP packet.

2. The IP addressing for the IGCP packet: Unicast, Multicast or Anycast.

3. The required forwarding behaviour for the IGCP packet. In IPv6, this may involve selectively combining Hop-By-Hop Options Header; Routing Header (this may require a new Routing-Type to be added to existing ones to support this behaviour), and/or Destination Options Header whenever applicable now and in the future evolution of IPv6.

4. In IPv6 networks, Neighbor Discovery (ND) protocol events may be used as triggers to the generation of an IGCP packet e.g. an entity that could leverage Neighbor Discovery related events may listen for such events and generate an IGCP message to some other peer entities in other nodes/devices in order to send information or update peers.

5. Conventions used in this document

In the future revision.

6. Message type and format for IGCP

We propose three types of generic IGCP messages for addressing the requirements outlined earlier in the problem statement, namely: ({Information Request}, {Information Offer} and {Information ReceiptACK}) and two messages for addressing the need for "negotiations" discussed earlier in the problem statement ({Negotiation Offer}, {Negotiation Reply}).

The messages are of a somewhat a similar category to informational messages defined by ICMPv6.

6.1. IGCP Information Exchange Messages

In [RFC4620] two messages are defined: Node Information Query and Node Information Reply, but they are designed to be used only for discovering information about names and addresses. We propose three new types of IGCP messages for facilitating the exchange of generic control information: {Information Request}, {Information Offer} and {Information ReceiptACK} messages, that share the same general format presented below Figure 1.

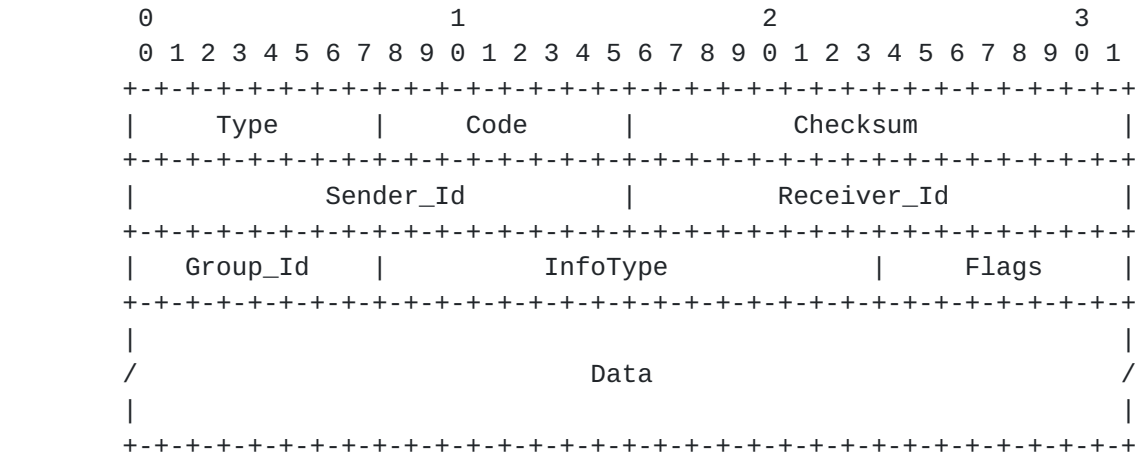

Figure 1: Information Request/Offer/ReceiptACK Messages.

Type
IGCP message type.

Code
For Information Request it can be used to further distinguish between different categories of information: capabilities of a functional entity e.g. a node/device , views regarding incidents and policies, etc.
For Information Offer:

* 0 - Indicates that a confirmation is needed (the receiver should send an Information ReceiptACK message in response to this Information Offer message). It means that the Information Offer message is not a response to an Information Request, but that the push model is being used instead.

* 1 - Indicates a successful reply to an Information Request message.

* 2 - Indicates that no confirmation is needed (no Information ReceiptACK message should be sent in response).

* 3 - Indicates that the receiver of an Information Request message refuses to provide the requested information (maybe because of prohibiting local policies)

* 4 - Indicates that the InfoType field is unknown to the responder or the Data field could not be decoded accordingly to the InfoType field.

For Information ReceiptACK:

* 1 - Indicates that the information was successfully received and processed.

* 3 - Indicates that the InfoType field presented in the Information Offer is unknown to the responder or the Data field could not be decoded accordingly to the InfoType field.

Checksum
IGCP checksum.

Sender_Id
A 16-bit field used to identify the sender functional entity. The identifier must be unique among all the functional entities inside a node since there can be multiple types of a functional entity inside a node/device that use the IGCP, each type of which requires its own customized use of the Data Part of the IGCP.

Receiver_Id
This field is similar to the Sender_Id, but it is used for identifying the receiver inside the node/device.

Group_Id
There can be a case when an IGCP message is of interest to more than one functional entity inside a node. For that purpose we envision that different groups of interest can be set up, and different functional entities inside the node can be part of a

specific group. This field should be used for identifying the group of functional entities for which the message is addressed. IDs and Group-ID need to be discovered, say in a similar fashion to the way Neighbor Discovery (ND) in IPv6 works.

InfoType

A 16-bit field that indicates the type of information requested in an Information Request or supplied in an Information Offer. Its value in a reply should be always copied from the corresponding received message (for an Information Offer with code different from 0 it should be copied from the corresponding Information Request message and for Information ReceiptACK it should be copied from the Information Offer message). The Information types need to be further researched.

Flags

There are specific flags for each InfoType defined for Information Requests or Offers. When no flags are defined for a given InfoType, this field must be zero on transmission and ignored on reception, and must not be copied from a Request to a Reply. An example of how the flags can be used is presented later on.

Data

We propose to keep the data field as "generic" as possible. The Data field must be left to be customized and defined according to the specific needs of specific types of functional entities that need to exchange control information (e.g. different types of functional entities called Decision Elements (DEs) in the case of a GANA based network described in [Chaparadza-FIA-Prague2009]). An example of how the Data field can be used is presented later in this draft.

The messages described above provide just the basic fields that can be used by any functional entity intending to exchange control information. The Data part of the message can be customized to have different fields and structure according to the specific needs of the functional entities involved in the information exchange process. The structure of the Data field is determined by the value of the InfoType field.

The Information Request message is used for requesting different types of information such as capabilities of functional entities (including protocols and mechanisms), "Views" (e.g. regarding incidents in the network or link state) etc. The Information Offer message is primarily meant to be sent in response to an Information Request message, but there are cases when the push model needs to be used and then the targeted functional entity could send directly an Information Offer message. The sender of an Information Offer

message can choose to request an acknowledgment or not by using the Code field inside the message. If an acknowledgment is required then the receiver of the Information Offer message must construct and send an Information ReceiptACK message.

6.2. IGCP Negotiation Messages

The {Negotiation Offer} and {Negotiation Reply} messages have the same structure presented as in Figure 2.

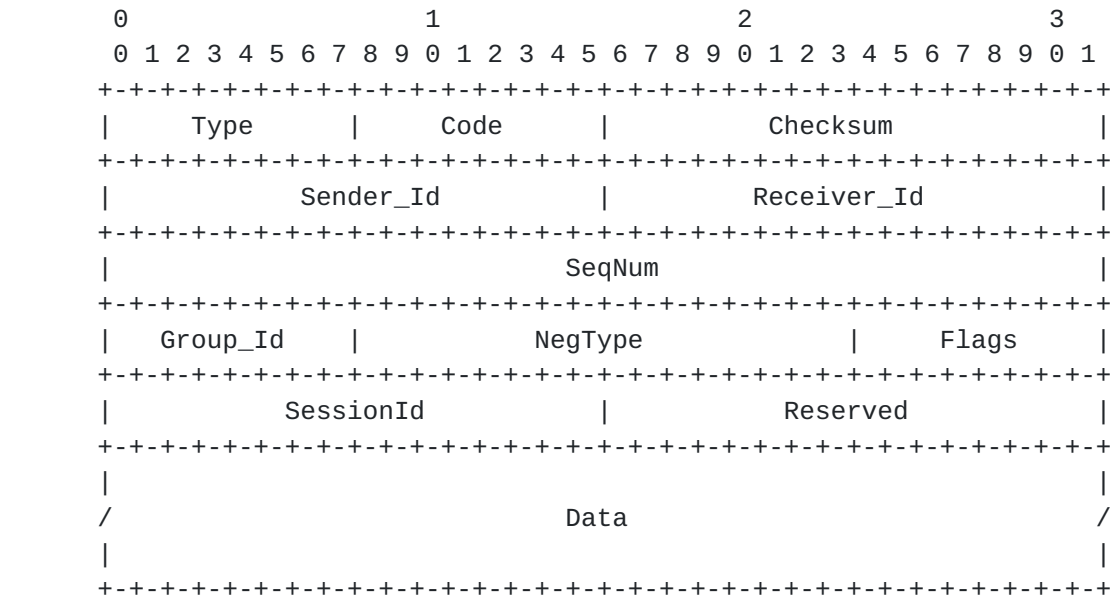

Figure 2: Negotiation Offer/Reply Messages.

Type
IGCP message type.

Code
Not used set to 0.

Sender_Id
A 16-bit field used to identify the sender functional entity. The identifier must be unique among all the functional entities inside a node since there can be multiple types of a functional entity inside a node/device that use the IGCP, each type of which requires its own customized use of the Data Part of the IGCP.

Receiver_Id
This field is similar to the Sender_Id, but it is used for identifying the receiver inside the destination node/device.

SeqNum
The 32-bit sequence number of the negotiation message inside a negotiation session. This field is important for mapping a Negotiation Reply message to the corresponding Negotiation Offer Message. Its value must be set by the sender in a Negotiation Offer message and should be copied by the receiver into the Negotiation Reply message.

Group_Id
There can be a case when an IGCP message is of interest to more than one functional entity inside a node. For that purpose we envision that different groups of interest can be set up, and different functional entities inside the node can be part of a specific group. This field should be used for identifying the group of functional entities for which the message is addressed.

NegType
Negotiation Type is a 16-bit field that indicates the negotiation protocol used for the current negotiation session. Its value is set in the Negotiation Offer message by the initiator of a negotiation session and must remain unchanged during the whole session.

Flags
There are specific flags for each NegType defined for Negotiation Offer or Negotiation Reply. When no flags are defined for a given NegType, this field must be zero on transmission and ignored on reception, and must not be copied from a Request to a Reply.

SessionId
An 16-bit field used to identify the messages that correspond to a negotiation session. Its value must be set up by the initiator of the negotiation session and this value must remain unchanged during the whole session.

Reserved
16 bits of reserved space

Data
Negotiation data.

The Negotiation Offer and Negotiation Reply messages (Figure 2) presented in the previous section are meant to be generic messages for facilitating the negotiation process between two or more functional entities. They provide the basic fields that may be required during the negotiation. The Data field should be used for carrying the additional information necessary in a specific case of negotiation. The negotiation process could be done in more than one

step, as illustrated in Figure 3.

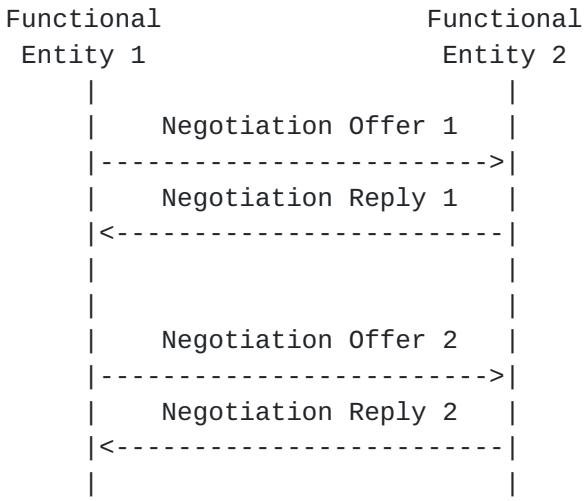

Figure 3: Negotiation Process.

The negotiation process can be based on a simple mechanism like in the case of HTTP content negotiation [RFC 2616], [RFC 2295] or more sophisticated negotiation algorithms may be developed when necessary.

7. Considerations of Usage in IPv4 Networks

Fragmentation support discussion in future revision.

8. Security Considerations

In the future revision.

9. IANA Considerations

In the future revision.

10. Acknowledgements

A number of people have contributed to the text and ideas. The list of these people include Shiduan Cheng (BUPT), Yuhong Li (BUPT), Tony Jokikyyny (Ericsson), Jan Melen (Ericsson).

In addition to people who have themselves contributed many ideas have come from the EU project consortium [EFIPSANS]. Our apologies to anyone whose name is missing.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4620] Crawford, M. and B. Haberman, "IPv6 Node Information Queries", RFC 4620, August 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

11.2. Informative References

[Ballani-Francis-ACM-SIGCOMM-vol37]
Ballani, H. and P. Francis, "CONMan: A Step Towards Network Manageability", ACM SIGCOMM Computer Communications Review, Volume 37(4), pages 205-216, 2007.

[Chaparadza-FIA-Prague2009]
Chaparadza, R., Papavassiliou, S., Kastrinogiannis, T., Vigoureux, M., Dotaro, E., Davy, A., Quinn, K., Wodczak, M., Toth, A., Liakopoulos, A., and M. Wilson, "Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering", FIA Prague 2009 Conference, Published in the Future Internet Book, 2009.

[Chaparadza-IEC-v60-Dec2007]
Chaparadza, R., "Evolution of the current IPv6 towards IPv6++ (IPv6 with Autonomic Flavours)", Published by The International Engineering Consortium (IEC) in the Review of Communications, Volume 60, Dec 2007.

[Chaparadza-IEC-v61-Dec2008]
Chaparadza, R., "Requirements for a Generic Autonomic Architecture (GANA), suitable for Standardizable Autonomic Behaviour Specifications of Decision-Making-Elements (DMEs) for Diverse Networking Enviroments", Published by The International Engineering Consortium (IEC) in the Review of Communications,

Volume 61, Dec 2008.

[EFIPSANS]
EFIPSANS, "EC funded- FP7-EFIPSANS Project", <http://efipsans.org/>.

[Greeberg-ACM-SIGCOMM-vol35]
Greenberg, A., "A clean slate 4D approach to network control and management", ACM SIGCOMM Computer Communications Review, Volume 35(5), pages 41-54, 2005.

[Retvari-MACE2009]
Retvari, G., Nemeth, F., Chaparadza, R., and R. Szabo, "OSPF for Implementing Self-adaptive Routing in Autonomic Networks: a Case Study", in proceedings of the 4th IEEE International Workshop on Modeling Autonomic Communication Environments (MACE2009), Venice, Italy, Oct 2009.

Appendix A. Example domain of application of IGCP: The growing concept of Self-Management and Adaptive Control

A.1. The growing concept of Self-Management and Adaptive Control

Self-Management and Adaptive Control is a new growing concept for networks in which so-called Autonomic Manager Entities (i.e. Decision-making-Elements) are introduced into the node/device architectures and the overall network architecture (refer to Figure 4 and Figure 5). The manager entities are required for "autonomically" configuring and controlling i.e. regulating and dynamically adapting the behaviour of network protocols and mechanisms by setting and re-setting parameters(variables) i.e. invoking management operations (in general) exposed by the "management interfaces" of the individual protocols and mechanisms based on some dynamic views analyzed by the manager entities. Such views can be goals and policies governing the way the protocols, stacks and mechanisms should be configured and adaptively adjusted, or incidents observed and processed by the manager entities. Autonomic management and control as the paradigm may be called aims at automating management operations/functions while at the same time ensuring self-adaptation of individual systems and the network through interacting control-loop structures designed to operate at various levels of abstraction of functionality. The notion of "control" is associated with a combination of "observing, supervision/regulating/adapting behaviour of managed entities i.e. something a controller from control-theory does.

The manager entities need to be able to communicate with each other some control types of messages depending on their co-operative goal as discussed later in this draft. We shall refer to the Autonomic Manager Entities as Decision-Making-Elements. Recently, an example of an evolvable, standardizable architectural Reference Model for Autonomic Networking and Self-Management within node and network architectures, dubbed the Generic Autonomic Network Architecture (GANA), has emerged [EFIPSANS] [Chaparadza-IEC-v60-Dec2007]. The concept of Autonomicity - realized through control-loop structures operating within network nodes/devices and the network as a whole, is an enabler for advanced and enriched self-manageability of network devices and networks. 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) e.g. protocols, stacks and mechanisms. An advanced Decision-making-Element (DE) may exhibit cognitive behaviour. Therefore, in GANA, self-* functionalities such as self-configuration, self-healing, self-optimization, etc, are functionalities implemented by a Decision Element(s). The "generic nature" of GANA lies in 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 must be supported on the management-interfaces of the individual MEs. In [Chaparadza-IEC-v60-Dec2007] different types of "instantiation" of GANA for autonomic network management and adaptive control of protocols for different types of network environments are illustrated.

The Generic Autonomic Network Architecture (GANA) [Chaparadza-IEC-v61-Dec2008] introduces "autonomic manager components/elements" known as Decision-Making-Elements (DMEs or in short - DEs) meant to operate at four different abstraction levels of functionality. These "autonomic manager components" are designed following the principles of "hierarchical", "peering", and "sibling" relationships among each other within a node or network. Moreover, these components are capable of performing autonomic control of their associated Managed-Entities (MEs), as well as co-operating with each other in driving the self-managing features of the Network(s).

In GANA, four basic hierarchical levels of abstractions for which DEs, MEs, Control-Loops and their associated dynamic adaptive behaviours can be designed (refer to Figure 4 and Figure 5). The levels of abstractions are as follows:

o Level-1: protocol-level (the lowest level) by which self-management is associated with a network protocol itself (whether monolithic or modular). This applies to those kinds of protocols that need to intrinsically embed control-loops similar to the type of control-loops embedded in TCP or OSPF. There is growing opinion, however, that future protocols need to be simpler (i.e. with no decision logic embedded) than todays protocols which have become too hard to manage due undesired emergent behaviour during operation and interaction with other protocols. This means, there is a need to rather implement decision logic at a level higher (i.e. outside the individual protocols)[Refer to sources such as CONMan [Ballani-Francis-ACM-SIGCOMM-vol37], 4D [Greeberg-ACM-SIGCOMM-vol35], etc.]

o Level-2: the abstracted function-level (directly above the protocol(s)-level) that abstract some protocols and mechanisms associated with a particular function e.g. routing, forwarding, mobility management, etc-whereby we can reason about autonomic routing (see example instantiation of GANA for realizing autonomic routing in [Retvari-MACE2009]), autonomic forwarding, autonomic fault-management, autonomic configuration management;

o Level-3: 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;

o Level-4: the level of the network's overall functionality and behaviour (the highest level). The figure below (Figure 4) illustrates that at node/system level of self-management (autonomic) properties, the lower level Decision-Making-Elements operating at the level of abstracted networking functions become the Managed-Entities of the main Decision-Making-Element (DME) of the system (node). This means the node's main DME has access to the "views" exposed by the lower level DMEs and uses its overall knowledge to influence (enforce) the lower level DMEs to take certain desired decisions, which may in turn further influence or enforce desired behaviours on their associated Managed-Entities, down to the lowest level of individual protocol behaviour. A "Sibling" relationship simply means that the entities are created or managed by the same upper level Decision-Making-Element (DME/DE).

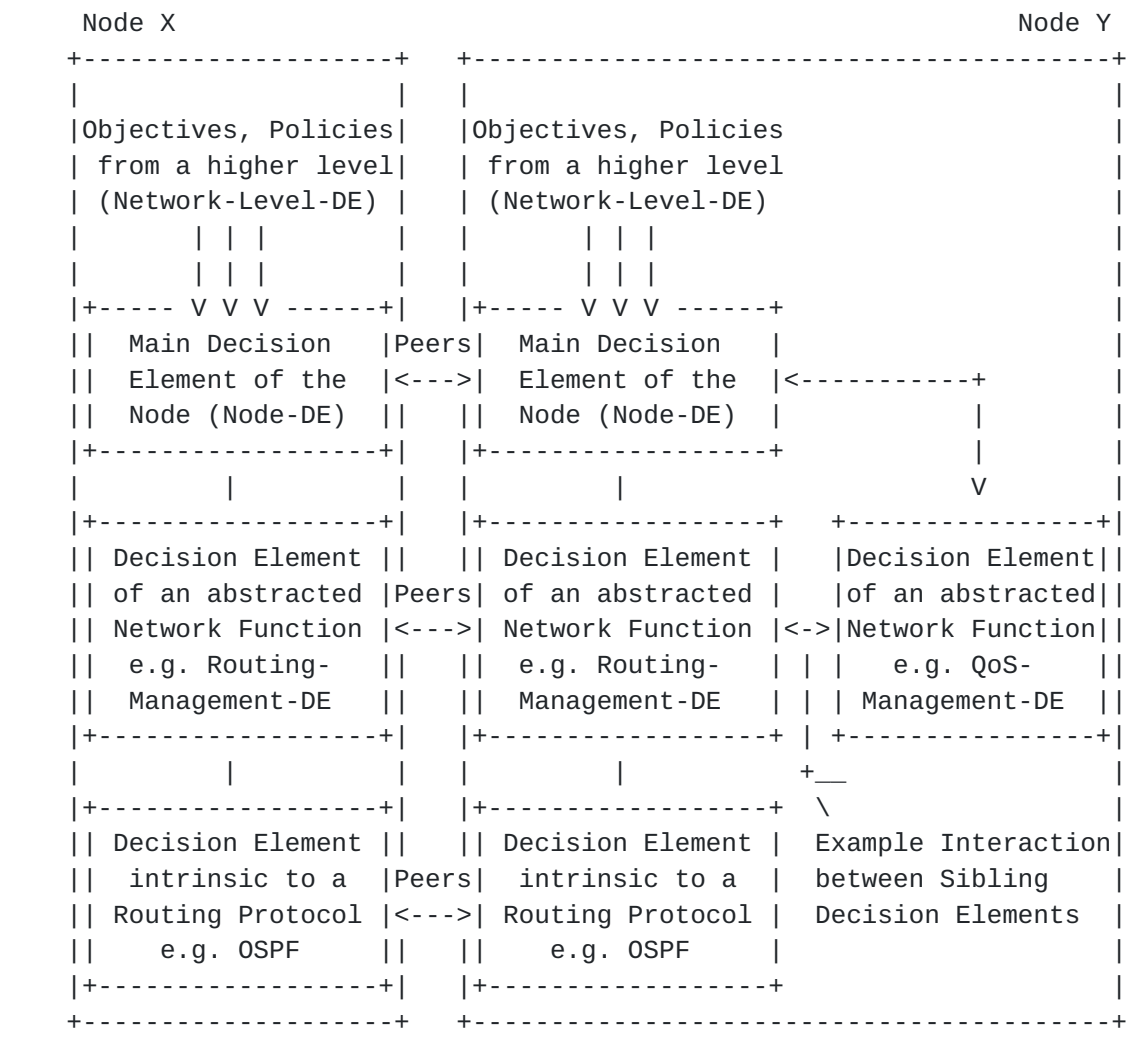

Figure 4: Hierarchical, Peering, Sibling Relations and Interfaces of DEs in GANA.

This means that the entities having a sibling relation can still form other types of peer relationship within the autonomic node or with other entities hosted by other nodes in the network, according to the protocol and communication needs defined for their needs to communicate with other DEs.

The GANA Level-4 i.e. the "network-level" represents the last level of autonomicity i.e. the highest level of self-manageability (determined by some associated control-loops for autonomicity). There may exist a logically centralized network-level Decision-Making-Element(s) or the kind of DEs proposed in the 4D network architecture [Greeberg-ACM-SIGCOMM-vol35] belonging to an isolated "overlay decision cloud" outside the nodes/devices (refer to Figure 5), which is considered to know (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 of the nodes of the network that are covered by the logically centralized network-level DME(s) in the "overlay decision cloud", export "views" such as events and state information to the logically centralized DME(s). This may happen in order for the logically centralized DME to influence or enforce the DMEs 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 DMEs of individual nodes/devices i.e. down to protocol level decisions. A distributed network-level control-loop may be implemented following the above set-up, while another case of implementing a distributed control-loop would involve the main Decision-Making Elements of nodes/devices working co-operatively to self-organize and manage the network without the need or intervention of a "specially instrumented" logically centralized higher level network-level DME(s) in an isolated "overlay decision cloud". Such a logically centralized network-level DME(s) 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, data mining and management operations that must work in harmony with autonomous self-management at node-level down to the protocol-level. The second case implies that the nodes/devices need to be designed in such a way as to have the possibility for the nodes/devices themselves to self-organize and perform "intrinsic" management and control-which can only be achieved to a certain extent due to resource limitations of the nodes/devices and the problem of longer convergence time with some distributed decision-making algorithms. All DEs from an stack with a "vertical view" and a "horizontal view" of interfaces and interworking with each to form the Decision-Plane of the network. The vertical view would replace in the long term, the traditional Management Plane.

In the GANA Reference Model:

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

o 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).

In [Retvari-MACE2009] an instantiation case for autonomic management and control of IPv6 routing protocols and mechanisms is illustrated.

The GANA Reference Model can be viewed as a holistic architectural

Reference Model for autonomic network engineering and self-management. It is holistic in the sense of the four basic levels of abstraction of functionality at which "inter-working nested control-loops for self-management" can be introduced. Moreover, it is meant to also address what may simply be called "intrinsic management control and within a node/device and collaboratively among network devices" as well as depicting "boundaries and constraints" for this type of intrinsic management and control. This means that the issues that require centralization of some of the autonomic decision-making processes of the network should be captured and defined by such the GANA Reference Model. In addition, appropriate interfaces of the kind of Network- level Decision Elements (DEs) that take care of the sophisticated centralized decision-making-processes must be defined by GANA. Network-Level DEs must allow for some "network-intrinsic management and decision-making-processes" to take place harmoniously at a lower level within the network nodes/devices themselves. In GANA, Network-level DEs, which implement the "slower control-loops", need to interact with the lower level (Function-Level DEs) that implement "faster control-loops" within the nodes/devices. We refer the reader to [Chaparadza-FIA-Prague2009] for more information on the subject.

DEs operating at a certain level within GANA hierarchy of DEs may need to exchange control information, as described in the next section.

A.2. The need for a multi-purpose Generic Control Protocol in IP based networks

In IP-based node and network architectures that introduce such manager entities that need to exchange different types of control messages for the purposes outlined below, there is a need to introduce a generic control protocol in order to address the requirements outlined below. The protocol (IGCP) should capture the parts that must be generic in the types of messages while identifying the items that must be mandatory in those types of messages. IGCP can be used by any entities of a node that require to exchange control messages, not just Manager Entities (i.e. DEs) presented by the GANA Reference Model.

In the case of Manager Entities such as the GANA DEs, residing in two or more nodes/devices, the entities may require the ability to perform the following operations:

(a) Negotiating the way their assigned Managed Entities (MEs) e.g. Protocols, Stacks and mechanisms of the node/device, should be configured or dynamically re-configured. The need to negotiate configuration settings e.g. parameter value settings applies to the

case where there is no need to involve a centralized coordinating entity (e.g. Network Management System), and so the peer "manager elements" (i.e. DEs) in nodes/devices need to self-organize certain aspects of the network e.g. the way some protocols must behave according to some policies.

(b) Soliciting for Capabilities of the peer manager element (DE) or multiple peers based on the features supported by their associated MEs(i.e. capabilities of the managed protocols, stacks and mechanisms--managed resources).

(c) Self-Advertising Capabilities of functional entities such as a node/device as a whole to peer "manager elements" (DEs) hosted by devices on the link, or to selected peers by policy.

(d) Exchanging Trust related information/data. This could be done by the Node_DE (autonomically manages and controls the behaviour of the whole node) or also at lower level DEs within the node/device.

(e) Exposing "Views" to peer DEs, such as detected incidents, mis-behaviours, etc, which can be used by the peers for managing the (re)-configuration of their associated MEs.

(f) Requesting for "Views" from peer DEs.

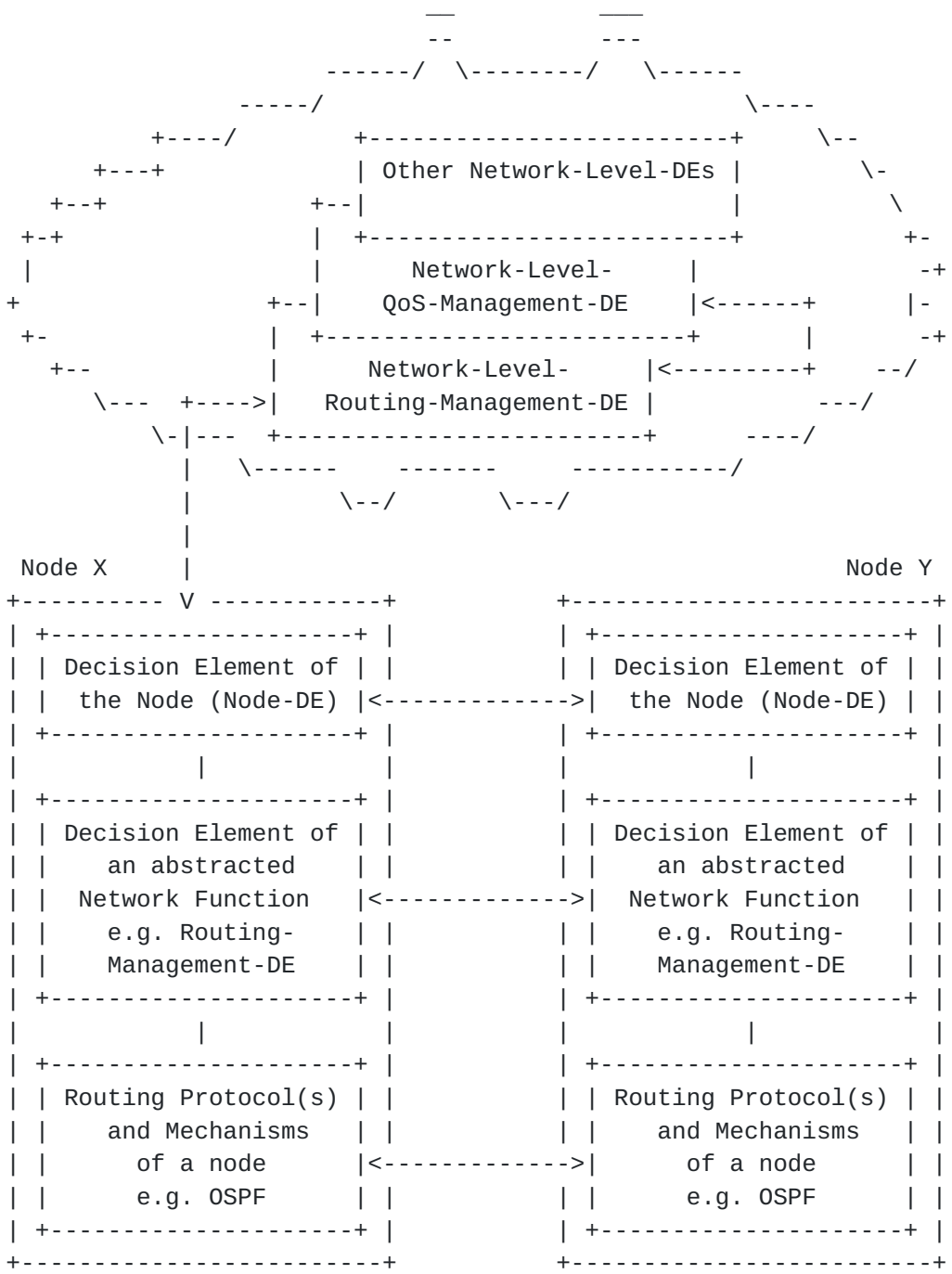

Figure 5: DEs communicating using IGCP.

A.3. Example Customization of the Data Part of IGCP and Use Case Scenario of the IGCP Information Exchange Messages

IGCP is meant to be used by any functional entities intending to exchange control messages for any of the requirements outlined earlier in the problem statement. Here we focus on one example of a

usage case of IGCP. In the following paragraphs, we give a concrete example of how the Information Offer message can be used for exchanging control information by two GANA DEs residing in different nodes/devices. Here, we focus on the FUNCTION_LEVEL_ROUTING_MANAGEMENT_DEs (FUNC_LEVEL_RM_DEs) that, according to the GANA Model, are sitting on top of their Managed Entities (MEs) i.e. the routing protocols and mechanisms of a node/ device and use specially customized messages to communicate with each other across nodes/devices. In [Retvari-MACE2009] where an instantiation case for autonomic management and control of IPv6 routing protocols and mechanisms is illustrated, it is argued that decision logic designed to operate outside and at a level of abstraction above protocols enables such instrumented decision logic to configure and dynamically adapt the behaviour of protocols based on information about the network and its goals that is not known or available to the individual protocols managed by such decision logic. Therefore, Function-Level-Routing-Management-DE (FUNC_LEVEL_RM_DE) inside a node is responsible for autonomic management and control of the routing protocols and mechanisms of a node by orchestrating the routing protocols, configuring them and applying policies, as well as listening for context changes and incidents and adapting the protocol behaviours. In order to efficiently, autonomically manage and control the Managed Entity (ME) e.g. the OSPF protocol based on context changes and incidents, by setting and resetting some parameters such as those defined by the Management Information Base (MIB) for the target protocol e.g. Timers and link-weights in OSPF, FUNC_LEVEL_RM_DEs in different nodes/devices in the network need to exchange control information. They need to encapsulate the control information into the Data field of the previously described messages. For this example case of control information exchange according to the requirements presented earlier, three messages from the point of view of the customizable Data Part of an IGCP message, are introduced and their structure and fields are described below.

Therefore, we illustrate how the Data field can be specially tailored i.e. customized for exchanging link state information that complements link state information exchange intrinsically built into OSPF. The FUNC_LEVEL_RM_DE of a router is responsible for autonomically managing and controlling the behaviour of all the routing protocols and mechanisms of the device e.g. OSPF (refer to [Retvari-MACE2009], [Chaparadza-FIA-Prague2009] for more detailed information on the description of the FUNC_LEVEL_RM_DEs). The adaptive control is realized in a distributed manner as the FUNC_LEVEL_RM_DEs residing on different routers exchange information in order to take the most appropriate decisions for the overall network, such as how to adjust OSPF's timers to control the convergence or use FRR (Fast-Re-Route) to protect destroyed paths. Three kinds of messages that customize the Data Part of the IGCP are

introduced for facilitating the information exchange between FUNC_LEVEL_RM_DEs and their structure is presented below.

The reason why the DEs need to exchange the same types of messages that are exchanged by OSPF is that DEs must detect the state of the network in advance. The FUNC_LEVEL_RM_DE is a kind of controller of OSPF, so it must discover the change of the network before OSPF and decide how to adjust OSPF's running status. Using the same types of messages is for the consistency with OSPF. The FUNC_LEVEL_RM_DEs should discover the failure of the link/node before OSPF finds it and decide what to do with the failure. For example, when a link in the network is disconected, the DEs will discover the failure first because of much higher hello sending frequency. If the DE does nothing with the failure, OSPF will discover and handle the failure in more than 30 seconds, and that is not acceptable if the broken link has a high importance. Rather, the DE will adjust the timer so that OSPF will become more sensitive and discover the failure soon, so the failure will be recovered in an acceptable time. At the same time, DEs should acquire and keep the same cognitive knowledge concerning the network as the OSPF so that they will be able to spot a failure and make a protective remediation decision. From the above we can conclude that DEs need to run a similar protocol as OSPF to detect the state of the network by exchanging the same types of messages that are exchanged by OSPF. The difference is that DEs send hello messages with a much higher frequency to be much more sensitive than OSPF.

The reason why the FUNC_LEVEL_RM_DEs do not use BFD (Bi-directional Forwarding Detection) as a usable option is that DEs run a link state detecting protocol similar to OSPF not only for detecting the links' state and discovering failures, but also for keeping the same cognitive knowledge concerning the network as the OSPF. DEs (FUNC_LEVEL_RM_DEs) build a link state database, the database is considered to be the same as OSPF's link state database. DEs use the link state database to calculate the importance of each link, and figure out the damage degree of the network after failures have occurred. Additionally, using an independent hello mechanism would be more general and facilitate the deployment of DEs, so DEs do not use options such as BFD to detect the link state.

FUNC_LEVEL_RM_DEs are NOT meant to replace OSPF but they configure OSPF and dynamically re-configure OSPF in trying to adapt the protocol behaviour according to events and incidents detected in the network's operation. The FUNC_LEVEL_RM_DE uses two methods to configure OSPF, one is adjusting OSPF's timers while the other is using FRR (Fast-Re-Route) to replace broken paths with backup routes. When OSPF converges, the DEs will calculate backup route for the important links. If the links are broken, the DE will activate the

backup route and use it to replace the broken route to guarantee communications(connectivity). When the network converges again, the DE will cancel the backup route and calculate the backup routes for important links again.

Neighbouring FUNC_LEVEL_RM_DEs discover each other and establish bidirectional communications by sending and receiving continuous Hello messages. After establishing the communication, they synchronize the link state database with each other by sending Link State Exchange message, so that the FUNC_LEVEL_RM_DEs in the network will have the same link state database. If a link state changes, such as a link failure, the FUNC_LEVEL_RM_DE that detects the change will send a Link State Advertisement message to all its neighboring FUNC_LEVEL_RM_DEs. Receiving the message, the neighboring FUNC_LEVEL_RM_DE will update its local link state database and relay the message to all of its neighboring DE, so that the change of the link state will be known by all the FUNC_LEVEL_RM_DEs in the network.

The Hello message (Figure 6) is used for establishing and maintaining connections to the neighbouring FUNC_LEVEL_RM_DEs. Hello messages are sent by each FUNC_LEVEL_RM_DE every Hello_Interval seconds from all router interfaces in order to establish a bidirectional communication with all its FUNC_LEVEL_RM_DEs neighbours. When two FUNC_LEVEL_RM_DEs have established a bidirectional communication, they begin to synchronize their topology databases.

The synchronization is performed by using Link State Exchange messages (Figure 7). This process takes place in two steps. First, the two FUNC_LEVEL_RM_DEs involved must negotiate which DE is the Master and which one is the Slave. This is done easily by selecting the FUNC_LEVEL_RM_DE that has the largest value in the Sender_Router_Id field as the master. Once the roles of the DEs have been determined, the asymmetric exchange of information begins. The master DE then sends its database in a sequence of Link State Exchange messages. The messages are sent one at a time and each message is acknowledged by the slave DE (by setting the bit A to one and keeping the same sequence number). The M bit must be set to 1 if there is more than one record in the database. When the master DE transmits its last database record, it will set the M bit to 0 to indicate that there are no more records to be sent. After receiving a message with M bit set to 0, the slave DE begins to send its database to the master DE. When the slave DE sends the last database record it will set also the M bit to 0 and the synchronization process completes.

When a link state changes (e.g. in case of a link failure), the FUNC_LEVEL_RM_DE that detects the change will send a Link State Advertisement message (Figure 9) to all its neighboring

FUNC_LEVEL_RM_DEs. A Link State Advertisement message may contain updates about the changes of more than one link. The sequence number from the advertisement message is compared to the value from the DE's local database. If the sequence number is new, the receiver DE will update the state of the link in the local database and it will issue a new Link State Advertisement message to all other interfaces in order to propagate the updates.

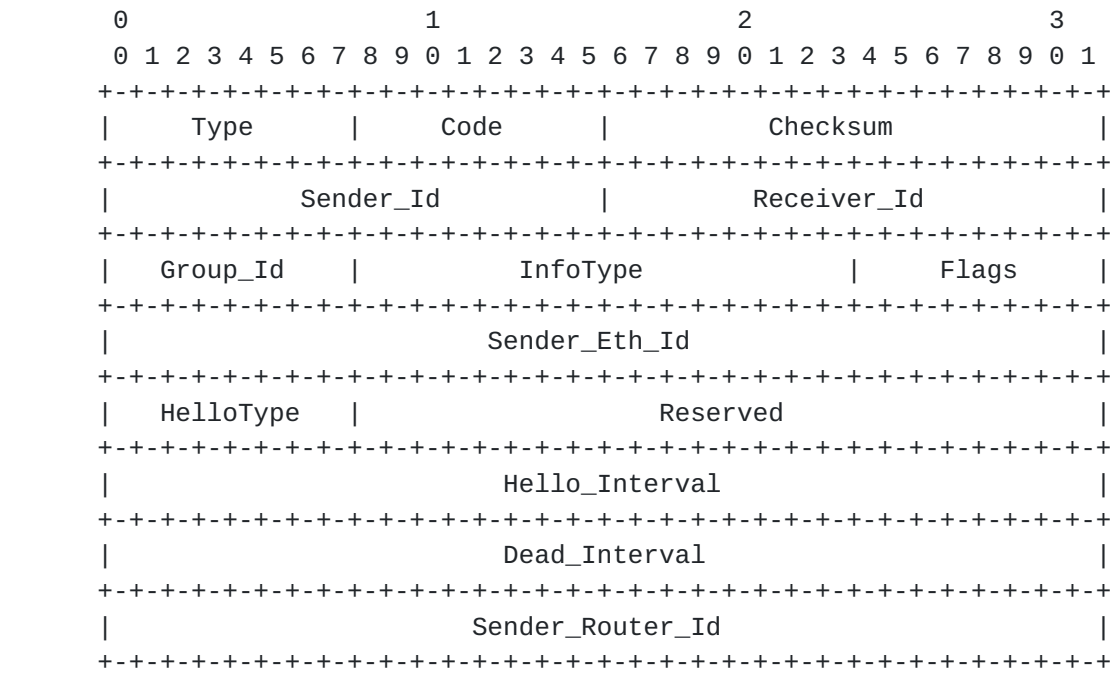

Figure 6: Hello Message.

Sender_Eth_Id
Interface identifier of the sender of the Hello message

HelloType
The type of the Hello message

Hello_Interval
The time between two consecutive Hello messages, expressed in seconds

Dead_Interval
The time for the dead interval, expressed in seconds

Sender_Router_Id
Router identifier of the sender of the Hello message

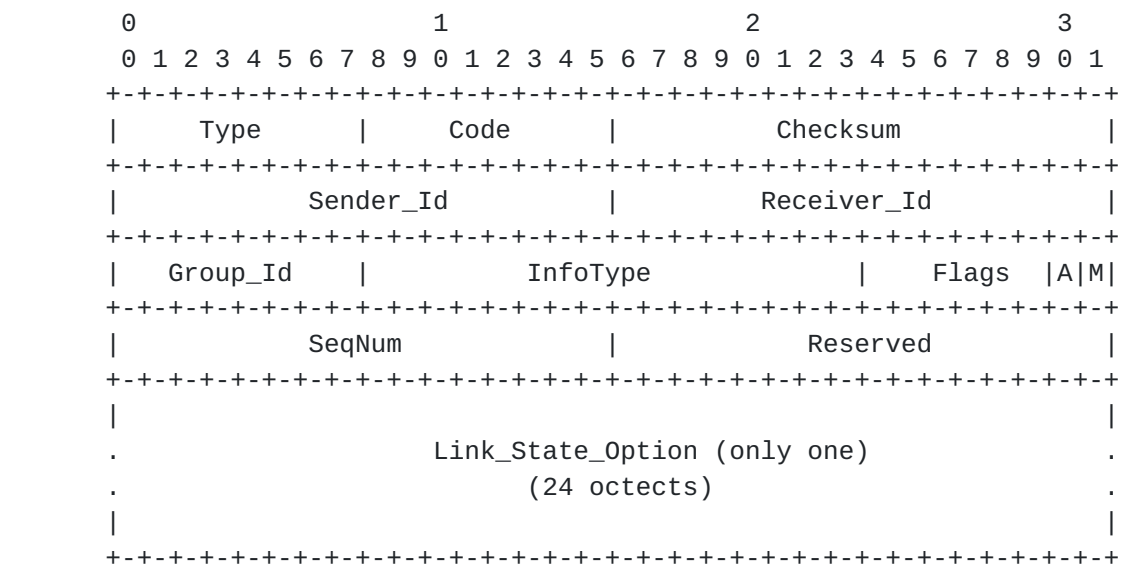

Figure 7: Link State Exchange Message.

Flags

A: Acknowledgement bit. This bit should be set to zero when sending Link State Exchange messages. If this bit is set to 1 then it indicates that the message is an acknowledgment (see next sub-section for more details).
M: If this bit is set to 1 more Link State Exchange messages will follow. When the last Link State Exchange message is sent, this bit must be set to 0 (see next sub-section for more details).
All the rest of the bits should be set to zero by any sender of a Link State Exchange message.

SeqNum

Sequence number of Link State Exchange message

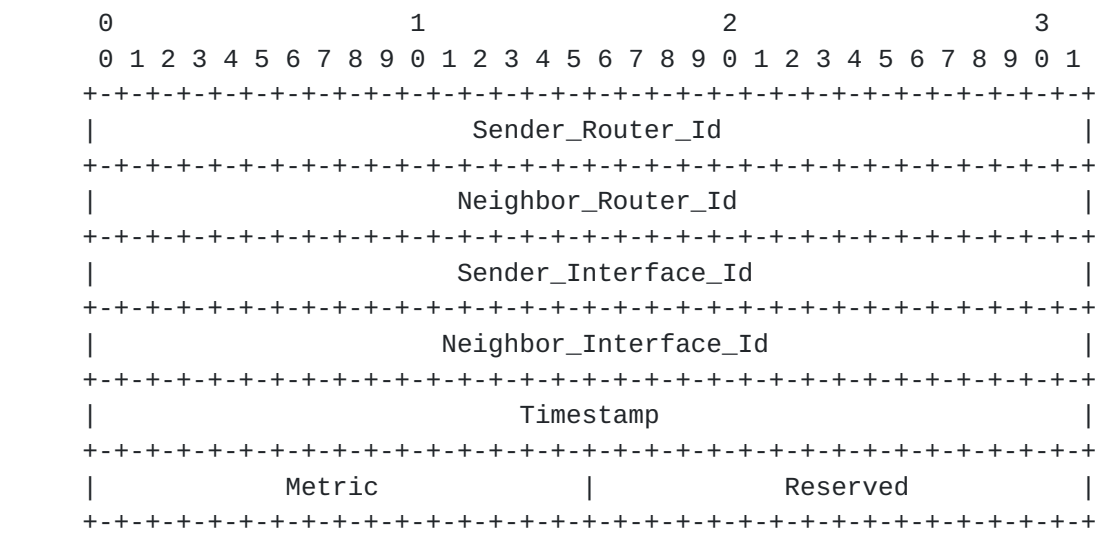

Figure 8: Link State Option.

Sender_Router_Id
   Router identifier of the sender

Neighbour_Router_Id
   Neighbour Router identifier

Sender_Interface_Id
   Interface identifier of the sender

Neighbour_Interface_Id
   Neighbour Interface identifier

Timestamp
   Timestamp at which this Link State Option was generated

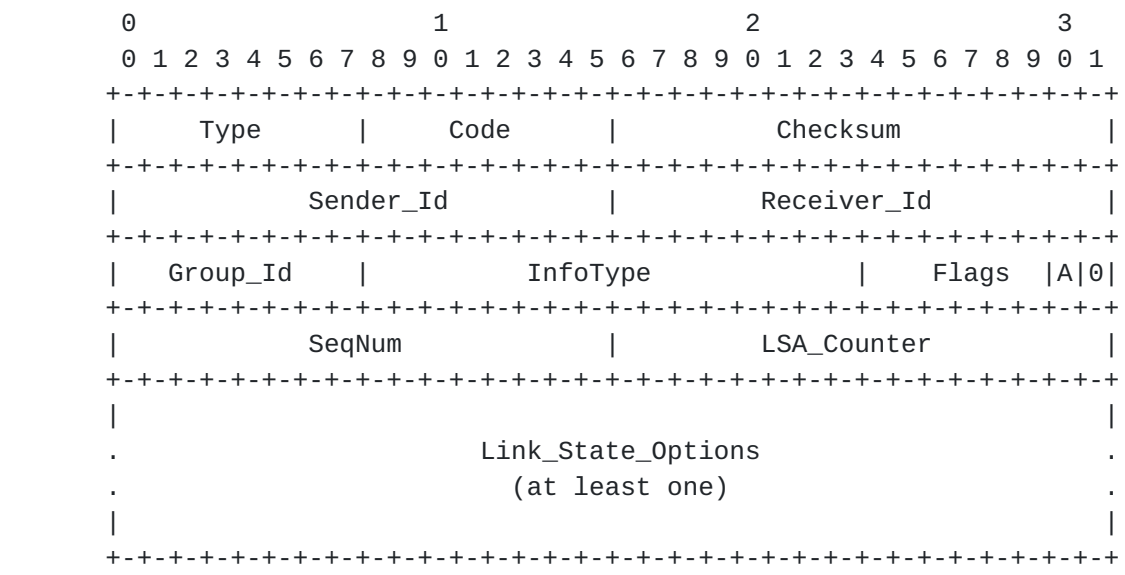

Figure 9: Link State Advertisement Message.

Flags

A: Acknowledgement bit. This bit should be set to zero when sending Link State Advertisement messages. If this bit is set to 1 then it indicates that the message is an acknowledgment (see next sub-section for more details).
All the rest of the bits should be set to zero by any sender of a Link State Exchange message.

SeqNum

The sequence number of Link State Advertisement message

LSA_Counter

This field indicates how many Link State Options (Figure 8) are present in the message.

Appendix B. Conclusions

The proposed generic control protocol (IGCP) and its generic Data Part offers an opportunity to accommodate different types of control information exchange contexts in the current and future IP networks. As an example domain of application of IGCP: the growing concept of self-management and autonomic networking, which can be applied to existing networking paradigms and architectures would benefit from such a multi-purpose control protocol as the network's Decision-making-Elements(DEs) defined by the GANA Reference Model presented in brief in this document, that are meant to dynamically (re)-configure, adapt and regulate the behaviour of protocols, stacks and mechanisms require exchanging control information. The IGCP protocol can be used by any types of functional entities intending to exchange control messages. When applied for use in the domain of autonomic networking, by Decision-Elements (DEs) defined by the GANA Model, the Data Part can be customized according to the needs of the different types of Decision Elements. In this draft we illustrated the customization of the Data Part of IGCP for the needs of one specific type of a Decision Element, namely the Function-Level-Routing-Management-DE (FUNC_LEVEL_RM_DE). The customization of the Data Part for the needs of other types of DEs would require the definition of more Internet Drafts accordingly.

Authors' Addresses

Ranganai Chaparadza
Fraunhofer Fokus

Razvan Petre
Fraunhofer Fokus

Heikki Mahkonen
Ericsson

Li Xin
BUPT

Michael Behringer
Cisco Systems