On Self-healing Based on Collaborating End-Systems, Access, Edge and Core Network Components

Nikolay Tcholtchev and Ranganai Chaparadza

Fraunhofer-FOKUS Institute for Open Communication Systems, Berlin, Germany
{nikolay.tcholtchev,ranganai.chaparadza}@fokus.fraunhofer.de

Abstract. Autonomic Networking, realized through control loops, is an enabler for advanced self-manageability of network nodes and respectively the network as a whole. Self-healing is one of the desired autonomic features of a system/network that can be facilitated through autonomic behaviors realized by control loop structures. Autonomicity, implemented over existing protocol stacks as managed resources, requires an architectural framework that integrates the diverse aspects and levels of self-healing capabilities of individual protocols, systems and the network as a whole, such that they all should co-operate as required towards achieving reliable network services. This integration should include the traditional resilience capabilities intrinsically embedded within some protocols e.g. some telecommunication protocols, as well as diverse proactive and reactive schemes for incident prevention and resolution, which must be realized by autonomic entities implementing a control loops at a higher-level outside of protocols. In this paper, we present our considerations on how such an architectural framework, integrating the diverse resilience aspects inside an autonomic node, can facilitate collaborative self-healing across end systems, access networks, edge and core network components.

Keywords: Autonomic Fault-Management, GANA-orientated architecture for Autonomic Fault-Management, Resilience, Self-Healing.

1 Introduction

Autonomic Computing as introduced by IBM [1] is based on *MAPE (Monitor→Analyze→Plan→Execute)* type of a control loop. Such a control loop is also meant to realize self-management features like self-healing, self-protection, self-optimization and self-configuration. According to [1], self-healing is defined as follows: *"To detect incidents such as adverse conditions, faults, errors, failures; diagnose, and act to prevent or solve disruptions"*. That is, on one hand a network equipped with self-healing mechanisms should aim at automatically preventing future fault activations, and on the other hand it should detect, diagnose and remove faults (provided that the detected faults can be automatically removed or have their impact reduced to a minimum). This corresponds to a number of concepts and approaches that have been investigated recently, such as Autonomic Fault-Management as well as reactive and proactive Resilience in autonomic networks. Autonomic Fault-Management [7][9] is understood as a control loop structure that facilitates the

R. Szabó et al. (Eds.): AccessNets 2010, LNICST 63, pp. 283–298, 2011.

interplay between the processes of Fault-Management as defined by TMN (Telecommunications Management Network) [3] namely: Fault-Detection – *"detect the presence of a fault"*, Fault-Isolation – *"find the fault (root cause) for the observed erroneous state"*, and Fault-Removal – *"remove or reduce the impact of the root cause"*. Moreover, resilience mechanisms have been developed for a variety of communication protocols and technologies, such as restoration schemes in SONET/SDH and recovery schemes in MPLS [RFC4427]. The introduction of control-loops that govern self-management and control of the existing protocols (as managed entities, i.e. managed resources of specific control-loops), calls for a framework that integrates an overall picture of the self-healing aspects and levels, at which to reason about self-management within a node/device architecture, and the network architecture as a whole. A first draft of such architecture was presented in our previous work [4] and is briefly described in section 3. Hitherto, the application of this framework to a single administrative domain of limited scope was taken into account. In this work, we present our further considerations on how this architectural framework could be used in a multi-domain environment consisting of end systems, access networks, edge routers, and core routers. Thereby, on the network provider's side we restrict the discussion to the internet service provider's (ISP) network, which enables the access to the internet backbone for its subscribers.

The rest of this paper is organized as follows: Section 2 presents the Generic Autonomic Network Architecture (GANA) which is the Reference Model for Autonomic Networking and Self-Management on which our self-healing framework is based. Section 3 presents the architectural framework (based on GANA) which reflects the current status of our research on self-healing/resilience mechanisms in autonomic networks. Section 4 presents the different aspects and mechanisms facilitating the collaboration of end systems, access and core network components towards implementing self-healing mechanisms across the different domains. Section 5 presents a case study and a scenario on how such collaboration can work in practice. Finally, section 6 provides some concluding remarks.

2 Generic Autonomic Network Architecture

The *Generic Autonomic Network Architecture (GANA)* is based on a set of requirements derived in [2]. The core autonomic concept in GANA is that of a *Decision Element (DE)*. A Decision Element (DE) implements a *control-loop* and manages a set of *Managed Entities (MEs) assigned to be autonomically managed and controlled*. That is, self-*/autonomic features are realized by Decision Element(s) implementing a control loop within the GANA reference model. Since control loops on different levels of functionality are possible, e.g. on node or network level, GANA defines the Hierarchical Control Loops (HCLs) framework. The HCLs Framework *fixes and establishes four levels of abstraction* for which DEs, MEs and associated control-loops can be designed: **Level-1:** *Protocol-Level*, i.e. control loops embedded within protocol modules (e.g. within some routing protocol). **Level-2:** *Abstracted Functions-Level*, i.e. DEs managing some abstracted networking functions inside a device e.g. routing, forwarding, mobility management, **Level-3:** *Node Level* - the node level consist of a Node_Main_DE that takes care of the management of aspects related to

the state/fitness of the overall node, e.g. Fault-Management and Auto-Configuration. **Level-4:** *Network Level*-DEs on that level manage different aspects that require to be handled at the network-level, e.g. routing or monitoring, of a group of nodes according to a network scope. Thereby, control loops (i.e. DEs) on a higher level manage DEs on a lower level down to the lowest-level "pure" MEs. Detailed information about all the presented concepts, examples, as well as discussions on the application of GANA to diverse aspects of Autonomic Networking can be found in [2].

3 Unified Architecture for Autonomic Fault-Management, Resilience, and Survivability in Self-managing Networks

Within the EFIPSANS [17] project, we introduced a *Unified Architecture for Autonomic Fault-Management, Resilience and Survivability in Self-Managing Networks* (UAFAReS). UAFAReS is based on the observation that the evolution of traditional Fault-Management towards Autonomic Fault-Management enables network devices to exercise self-healing and recover from faulty conditions. That is, the nodes of the network are then able to automatically self-heal (to some degree) without the need for human intervention. Hence, Autonomic Fault-Management has to interplay with concepts and mechanisms related to Fault-Tolerance, Fault-Masking, and Multilayer Resilience [6]. This implies harmonization (i.e. ordered time-scaling of reactions) to incidents, at different levels of autonomicity and self-management defined by GANA. UAFAReS is based on the GANA reference model and specifies a number of components which aim at realizing the interplay of the aforementioned aspects. The node components of the architectural framework are illustrated in Figure 1. The main UAFAReS entities in a device are the Fault-Management Decision Element (FM_DE) and the Resilience and Survivability Decision Element (RS_DE). The RS_DE is responsible for an immediate reaction to the symptoms of an erroneous state, while in parallel the FM_DE performs Fault-Isolation and Fault-Removal in order to eliminate the corresponding root cause(s). Both DEs are part of the Node_Main_DE, i.e. they are introduced on node level inside a GANA conformant device, in order to have exclusive access to all node functional entities (i.e. DEs and MEs) such that the overall autonomic behaviors of a node with respect to coping with incidents and alarms are synchronized to ensure node integrity. The UAFAReS DEs operate based on distributed control loops. The distributed nature of the UAFAReS control loops is enabled by a number of components that facilitate the incident information exchange across the network nodes. A *set of repositories for storing incident information* and an *Incident Information Dissemination Engine (IDE)* enable the synchronization of the faults/errors/failures/alarms knowledge known by UAFAReS DEs residing in different devices, and allow the DEs to perform Fault-Masking, Fault-Isolation and Fault-Removal in a node specific manner, based on the same information.

The FM_DE consists of four modules: 1) a component responsible for Fault-Isolation *(Fault-Diagnosis/Localization/Isolation functions* abr. *FDLI*), 2) *Fault-Removal Functions (FRF)*, 3) *Action Synchronization Functions (ASF)* – responsible for synchronizing (allowing and/or disallowing) tentative actions issued as by the RS_DE and the FM_DE control loops running in parallel, 4) *Fault-Removal Assessment Functions (FRAF)* – a component responsible for assessing and verifying the success of the fault removal actions issued as output of the FM_DE.

Specially instrumented monitoring entities, which have the capability to share incident information over the UAFAReS incident repositories, push descriptions of symptoms to the UAFAReS fault/error/failure/alarm registries such that the info gets conveyed (i.e. stored in the UAFAReS node registries) by the Incident Dissemination Engine (IDE) to the UAFAReS instances across the network scope, e.g. subnet/LAN. Once an incident description has been reported to the FM_DE over the UAFAReS incident repositories, it gets received and processed first by the FDLI functions. That is, the FDLI functions collect such events and correlate them in order to find the root cause for the observed faulty conditions. Algorithms that can be used for event correlation are presented in [8]. The correlation of incident events is realized by the FDLI functions based on a Causality Model that is kept in the *Causality Model Repository (CMR)* inside a node. The identified root cause(s) (faults) are then further submitted to the Fault-Removal Functions which implement an *"if-then-action"* logic that issues a reaction required to eliminate the faults, e.g. reconfiguration of an entity by using the corresponding command line interface (CLI). Since it is possible that the tentative reaction would interfere with other actions that are intended to be performed by either the RS_DE control loop (next paragraph), or would interfere with a parallel Autonomic Fault-Management control loop process (i.e. a thread in multi-threading environment), the ASF should be invoked in order to allow or disallow the tentative action in question. The ASF is based on techniques from the area of optimal control, and selects the optimal subset of tentative actions in order to better optimize the network performance reflected by the values of selected key performance indicators while ensuring integrity. Given that the ASF has allowed a tentative action, the FRF issues it on the MEs in question inside the device. Thereby the FRF can make use of information regarding the dependencies among protocol entities and services, kept in the *Dependability Model Repository (DMR)*. Finally, the success of the executed action is assessed by the FRAF functions, which may choose to notify the network operator in case when the UAFAReS mechanisms can't cope with the pending challenges.

The Resilience and Survivability DE contains the *Fault-Masking Functions (FMF)* component and a *Risk Assessment Module (RAM)*. The Fault-Masking Functions realize a reaction immediately after the symptoms of a faulty condition have been registered into the UAFAReS alarm/incident repositories. Thereby, the goal of the FMF is to implement a fault-tolerant behavior such that some fundamental level of service can be sustained in the face of a pending challenging condition. The FMF follow a similar logic as the Fault-Removal Functions of the FM_DE, and consult the Actions Synchronization Functions of the FM_DE to react first in order to ensure that the best possible set of actions is executed. The FMF, as the instance of first reaction, should also consider the aspect of Multilayer Resilience [6] while orchestrating a fault-tolerant/masking behavior. Multilayer Resilience is a model that deals with the capabilities of functional entities at different layers in the protocol stack to execute their own embedded resilience behaviors. For instance, in IP networks generated ICMP messages enable systems (especially end systems) to overcome issues occurring in the network, e.g. sudden changes of PMTU (Path Maximum Transmission Unit) during the lifetime of a connection. Thus, the FMF is expected to allow the protocol modules to recover based on their own intrinsic capabilities and should intervene only in the case when these mechanisms fail. [6] proposes the usage of "hold-off" timers specifying the time that should be given to a protocol to recover on

its own. Information on how to handle the resilience properties of a protocol module (e.g. protocol module ID and corresponding "hold off" timer) is kept in the *Multi-Layer Resilience Properties Repository*. In addition, the operation of the Risk Assessment Module (RAM) is based on monitoring information about diverse key performance indicators (e.g. CPU temperature) that are used to calculate the probability for failures in the future. This results in notifications to the FMF which consequently have to trigger mechanisms that help proactively avoid significant degradation in the QoS in the future.

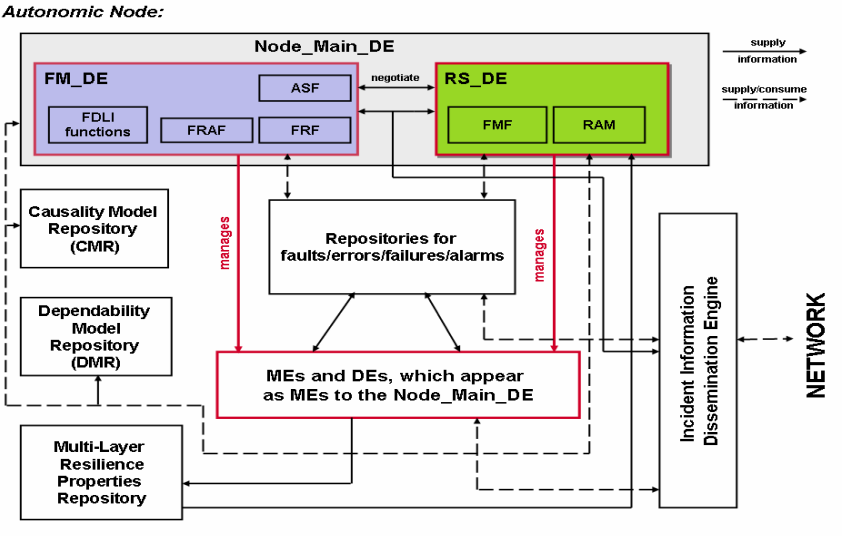

Fig. 1. The UAFAReS [4] architecture inside an Autonomic Node

Hitherto, we considered the application of UAFAReS to a single domain of limited scope. In this work, we present our view on how UAFAReS can be used in a specific multi-domain environment consisting of end systems, access networks, edge routers, and core routers implementing UAFAReS. Thereby, we have identified that what differs from domain to domain is the type of knowledge and models that the UAFAReS instances are supplied with. Thereby, it is imperative that on one hand, effective collaboration, i.e. knowledge/information exchange, is facilitated, and on the other hand, the confidentiality and integrity of the information exchanged between the domains must be ensured.

4 Aspects of Collaboration

In this section, we present on the different aspects of collaboration among end systems, access network entities, as well as core network components. These aspects should be considered in implementing self-healing based on collaboration of network entities across multiple network segments. Furthermore, we make an attempt to identify situations and issues where the collaboration can be beneficial for enabling fault

resolution. Thereby, we assume that the devices across the different networks are equipped with UAFAReS components and we specify the collaboration aspects as interactions between the UAFAReS instances.

4.1 Means for Auto-collaboration

In general, the base for UAFAReS collaboration over the domains in question is provided by the exchange of different types of information, either during the device/terminal subscription phase for the end systems, or during the phase in which they are utilizing the ISP's network. The information used by a UAFAReS instance can be classified as follows: **1)** Runtime information about detected incidents - stored in the *incident repositories* of a UAFAReS implementing device, **2)** Causality Model that describes information about potential chains of events (fault→error … →failure) – stored in the *Causality Model repository* of a node, **3)** Information about the resilience properties of involved network components – stored in the *Multi-Layer Resilience Properties Repository*, **4)** Policy configurations for the FRF, FMF and the FRAF modules of the FM_DE and the RS_DE. Figure 2 gives an overview of the information flows required to facilitate collaborative self-healing. In the following subsections, we discuss the importance of the different types of information outlined in Figure 2, thereby distinguishing between the subscription phase of a device to the ISP network and the operating phase when the network is utilized by the end system for services and applications.

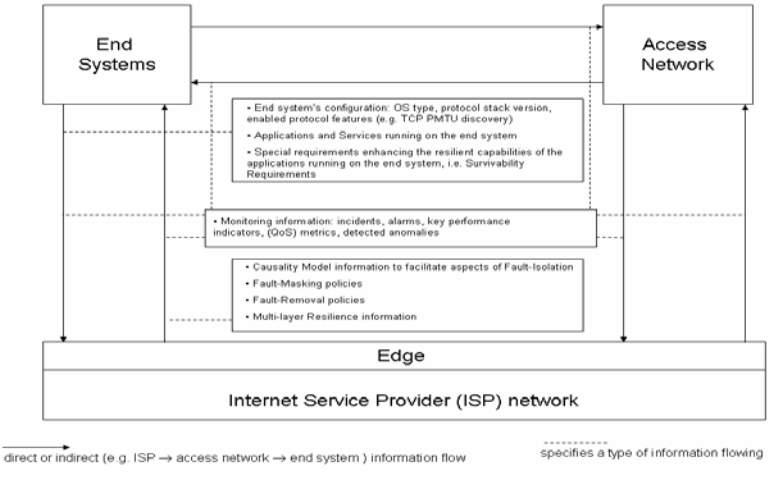

Fig. 2. Aspects of collaboration

4.2 Auto-configuration During the Subscription Phase for a Terminal: Enabling UAFAReS Based Self-healing

The *end systems* are expected to share with the operator's network, information about their settings and configurations, e.g. OS (Operating System) type and characteristics,

type and version of the employed protocol stack including enabled protocol features such as PMTU discovery, applications and services hosted on the end system, etc. This information is denoted as *device capabilities* and is provided during the subscription phase of the device into the operator's network. The device capabilities can be seen as accumulated capabilities of the plethora of software and hardware components of an end system. As described in Figure 3, the capabilities are required by the ISP network in order to select the appropriate UAFAReS related configurations which are sent back to the end system. This information may include (as illustrated in Figure 2) a Causality Model related to potential problems that can occur in the network such that the end system can better understand certain anomalies, Fault-Removal/Masking policies which enhance the set of such policies already in place on the end-system, information about Multilayer Resilience aspects related to the core network, which allow the UAFAReS instance on the end system to take into account the intrinsic resilience mechanisms of the lower communication layers of the network. An interesting research issue is that of the information model that facilitates the processes in Figure 3. The current trends in telecommunications are striving to reduce the usage of proprietary models and promote the usage of a single standardized model such as or CIM [14]. For the purpose of self-description by end systems, one has to be aware that due to different standards and technological domains, the exchanged information will suffer major drawbacks because of the lack of a standardized terminology. Hence, the usage of semantic based model concepts such as ontologies is imperative for the process of self-description and auto-configuration for UAFAReS. That is, the end system must submit (to the network) its capabilities in an ontological format such that the obstacle of different terminologies across software/hardware vendors, standardization bodies, etc. can be avoided.

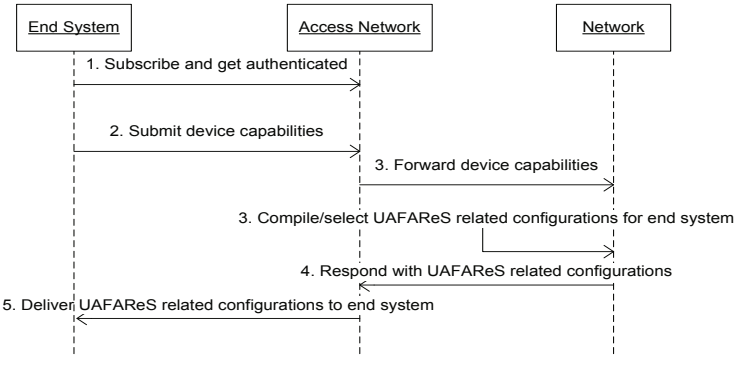

Fig. 3. UAFAReS auto-configuration of an end system

Another aspect that must be addressed when specifying the Causality Model as well as Fault-Removal/Masking policies to be supplied to an end system, is that of confidentiality of the exported information. If not designed properly, the aforementioned models and policies may contain information that directly or indirectly reveals

some sensitive information about the operator's network. Such information can be easily exploited by hackers. Therefore, it may be useful to use only some cryptographically generated strings (i.e. avoid using full descriptions of the events) in the version of the Causality Model which is given to an end system, or to delegate as many as possible reactions (Fault-Masking/Removal) of the control loops to the edge or access network components.

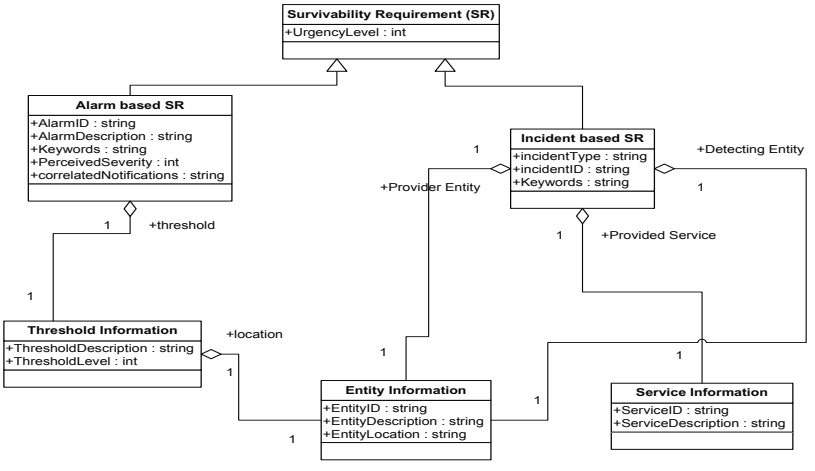

Fig. 4. Information Model for Survivability Requirements

Additionally, after getting to know the capabilities of the new subscriber, the autonomic mechanisms of the network have to prepare the information flows from the network to the subscriber such that the end system can access information about incidents in the ISP network and can correspondingly react according to the aforementioned Fault-Masking policies. This means that the network mechanisms formulate Survivability Requirements (SR) for the applications and services on the newly subscribed end system. A survivability requirement expresses time frame within which an entity requires to be notified of incidents, to enable it to employ its own mechanisms to avoid failure or degraded service beyond unacceptable service. These SRs define "filters" and specify the incident information that should be conveyed from the network to the end system. Furthermore, the end system can explicitly express its SRs to the network UAFAReS mechanisms as mentioned in [7]. The SRs are used as filters by the IDEs across the ISP's network in order to enable the automatic notifications towards the end system upon the detection of matching incident events. Figure 4 depicts the information model of a Survivability Requirement that we propose. This model is based on alarm/incident event descriptions as recommended by ITU-T [15] and CIM [14]. Due to space limitations, we omit the detailed description of the different attributes which are also self-explanatory to a large extent.

4.3 Collaboration and Information Flow during the Operation Phase

After an end system has subscribed to the ISP over the access network, selected types of monitoring information are expected to flow between the end system, the access network and the ISP's network. Figure 5 describes a generic scheme of how an end system can mask the local impact of an erroneous state that has been detected in the core network (step 1). The node in the core network, on which the erroneous state was detected, reasons about the need to send the incident description to the corresponding end system, e.g. the decision could be based on a previously expressed/generated Survivability Requirement and/or on the end system being one of the end points in a flow in which some traffic anomalies were detected. Moreover, the Edge of the ISP network intercepts (step 3) the message (after it has been sent in step 2), and checks whether the information inside can be sent to the end system based on the security policies in place. Obviously, such an incident message can contain information which the operator does not want to share in order to keep certain structures and configurations of her network a secret. Hence, the edge should be equipped with a policy that allows/disallows, or manipulates incident descriptions being sent to end systems. Given that the message has been allowed or anonymized, step 4 and 5

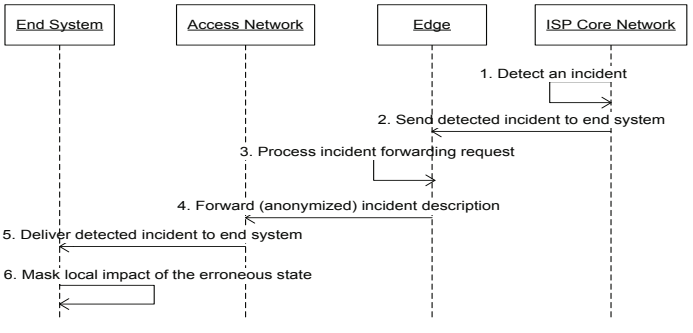

Fig. 5. An end system masking the local impact of an erroneous state in the ISP network

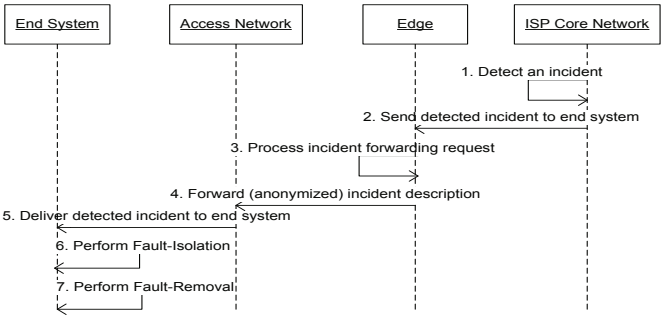

Fig. 6. An end system performing Fault-Isolation and Fault-Removal in case the incident, detected in the ISP network, indicates a fault (root cause) on the end system

deliver it to the end system. Based on the Fault-Masking policy, which has been supplied during the subscription phase of the end system (Figure 3), the end system is expected to react and mask/mediate the local impact of the erroneous state indicated by the received incident description.

In addition to the mediation behavior of the end system, which is realized by the Resilience and Survivability DE of UAFAReS, we also consider a behavior as the one described in Figure 6. In this case the end-system FM_DE gets activated and performs Fault-Isolation, and consequently Fault-Removal in case the root cause for the traffic anomaly observed in the core network is located on the end-system. The logistic that facilitates these processes includes the Causality Model and the Fault-Removal policies which must have been supplied during the subscription phase of the end-system. Furthermore, monitoring information, i.e. metric values or incident description of observed anomalies, may flow from the end systems to the ISP core network thereby enabling self-healing in the access network, the edge and core network. The signaling that realizes fault tolerant behaviors in the core network is described in Figure 7. In addition, the sequence of interactions and actions towards eliminating the root cause(s) of a faulty condition in the core network are illustrated in Figure 8.

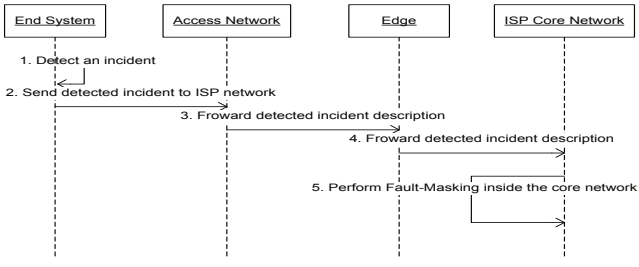

Fig. 7. Fault-Masking in the ISP network based on information supplied by an end system

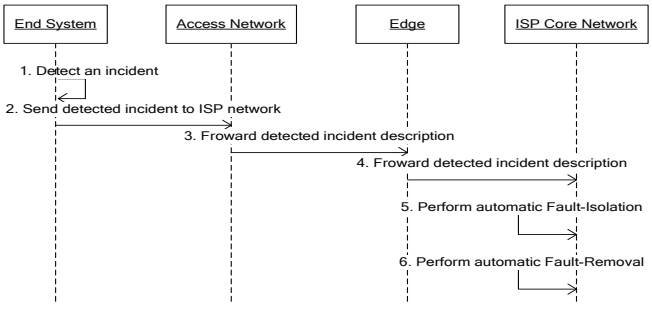

Fig. 8. Autonomic Fault-Management in the ISP network, including automatic Fault-Isolation and Fault-Removal, based on information supplied by an end system

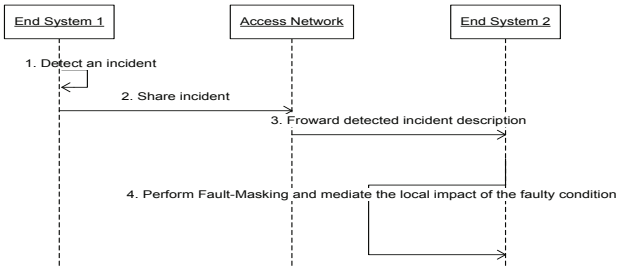

Fig. 9. Information sharing between end systems and consequent Fault-Masking of the local impact of a faulty condition

The sharing of incident event descriptions among end systems can enable UAFAReS based resilient behaviors on the subscribed end systems as illustrated in Figure 9. This sharing can be realized over the SOHO network or directly over the communication medium of the access network. It is also possible that different access network components, serving different subscribers collaborate and enable the exchange of incident information between the subscribed end systems. An example of such autonomic behavior, facilitated by the UAFAReS incident sharing mechanisms among end systems, is given in the next subsection.

4.4 Faulty Conditions Resolvable by UAFAReS Based Auto-collaboration

In this section, we give some examples of issues which can be resolved in case different access/core nodes and end systems implementing UAFAReS collaborate.

Black Holes: The term Black Hole represents a family of packet delivery problems where the physical (and in some cases even logical) connectivity between two systems is present, but however the packets sent between the two nodes (e.g. hosts) don't reach their destination. The erroneous state results from the fact that the systems, even having the capabilities to react to the fault activation, do not get notified of packet delivery failures and can not even localize the fault. That is, the sender may continue sending packets without detecting the packet loss problem and it can not react. The forwarding nodes are also not aware of the problem and do not adapt their behavior correspondingly. The phenomenon of Black Holes has been extensively studied and its relevance for ISP operators is now well known [13] [12]. Potentially, Black Holes may occur due to: 1) Loss of connection because of an incorrect PMTU or broken tunnels, e.g. MPLS. 2) Software bugs, 3) Delayed routing protocol convergence etc. The UAFAReS architecture can remediate some Black Hole issues by detecting and isolating them, reconfiguring the corresponding core network components, and informing the end systems in question to adapt their behaviors.

Duplex mismatches: IEEE 802.3 Ethernet networks are an established communication environment for LANs. These networks support an auto-negotiation procedure that allows the Ethernet interfaces on the involved nodes to automatically obtain and setup the optimal parameters on a link. The aforementioned parameters include the speed of the interface cards (10 MBit/s, 100 MBit/s, 1000 MBit/s), the duplex mode, as weIl as

flow control information. Whilst this procedure makes the setting up of a network easier, it can lead to a mismatch in the settings assumed by both NICs (Network Interface Cards) involved. For instance, the failure of the auto negotiation procedure may result in a mismatch of the duplex configuration on two peer interfaces, i.e. the one operates in full and the other in half duplex mode. This can seriously cripple the network and lead to performance degradation. Such duplex mismatches can occur in the SOHO network attached to the access network, as well as in the core or access network. In [11], the symptoms of such Ethernet Duplex mismatches are investigated. The potential symptoms include duplicate ACKs in TCP flows, fluctuations in the CWND (congestion window) size of TCP connections, as well as increased collusions on the MAC layer and frame losses at nodes' NICs. Some of these anomalies can be detected only on the end systems, e.g. CWND size fluctuations. On the other hand, in case a duplex mismatch is identified somewhere along a path, the UAFAReS framework would disseminate the information to the involved network components and enable/implement diverse resilient behaviors. That is, the nodes attached to the misconfigured link would restart the corresponding NICs in order to reinitiate duplex auto-negotiation. Given that the involved end systems have been notified about the issues (Figure 5), the UAFAReS mechanisms could hold outgoing traffic for a period of time which should be enough for the forwarding nodes along a path to recover. That way, traffic is not unnecessarily (since it will for sure get lost during the fault removal phase) pushed into the network, and the user will probably experience just a short delay.

TCP MSS size problems: [RFC2923] describes a case that results in a smaller (then possible based on the actual PMTU) Maximum Segment Size (MSS) advertised and used in a TCP connection. This leads to a limited segment size of the TCP connection between end systems, and respectively to degradation in QoS of TCP applications using the affected TCP session. UAFAReS mechanisms (monitoring sensors) inside the network could detect such symptoms, identify the aforementioned root cause for the QoS degradation, and inform the UAFAReS instance on the end system in question to readjust its MSS size settings.

(D)DoS detection: We hold that the mechanisms of the UAFAReS architecture can also be employed for security related issues realizing a self-protecting/defending functionality. Recently, a lot of research has been conducted in the area of intrusion detection. The study and classification of traffic anomalies towards the detection of malicious software preparing a Distributed Denial of Service (DDoS) attack is an ongoing research topic the results of which can be easily integrated with the UAFAReS incident sharing mechanisms. In this way, end systems that detect such suspicious activities may send this information to the UAFAReS instance at the edge of the ISP's network. After having correlated these notifications, and having identified the threat of a DDoS attack, the UAFAReS instance on the edge can implement a policy in order to counter the intended attack.

Corrupted services, e.g. DNS: Given that an end system or any functional entity has detected a malfunctioning service, the UAFAReS incident sharing mechanisms can be employed to disseminate this information and inform other potentially affected end systems. For instance, an end system may detect the unavailability of a DNS server

and use the UAFAReS mechanisms in order to share this information and enable other end systems to switch to another DNS server as long as the primary one is offline.

Maintenance activities: According to [5], maintenance activities are responsible for around 20% of the failures observed in an operational IP backbone. The UAFAReS architecture can potentially detect the corresponding outage, identify the reason for it and apply, for example, some admission control policies in order to guarantee best QoS for the already subscribed end systems.

PMTU issues: Traditionally, IP protocols implement a PMTU (Path Maximum Transmission Unit) discovery procedure on end systems. PMTU discovery is performed before the data transmission has started. This enables the end systems to obtain a valid PMTU for a connection towards another end system using the services of an operator's network. However, it is possible that the PMTU towards a host decreases during the lifetime of a connection. In such cases, the router at which the PMTU has decreased will fail to forward a packet and is expected to respond to the end system with an ICMP message indicating the packet loss and the reason. The IP module on the sender end system is expected in turn to readjust its PMTU settings towards the receiver. However, this procedure is very often not possible due to reasons such as firewall ICMP suppression on routers, or simply because the traffic that cannot be forwarded is being tunneled (e.g. a VPN tunnel) and the ICMP messages are correspondingly not relayed. In such cases the sending host fails to adapt its fragmenting behavior and the packets towards the receiver fail to reach their destination even though there is an intact physical connectivity. Obviously, this constitutes a specific type of a Black Hole as previously described. We believe that a framework such as UAFAReS can enable the implementation of mechanisms which allow the collaborating end systems, access and network components to overcome such issues with PMTU changes.

5 Case Study: Overcoming Potential Problems with PMTU Changes in an IPv6 Network, Resulting from IPv4 to IPv6 Transitioning

Our scenario is based on the Path MTU problems described in [RFC2923]. [RFC2923] describes a specific type of a Black Hole phenomenon that can potentially occur in IPv6 networks wrongly configured with firewall configurations adopted from IPv4 firewall configuration rules that drop ICMP messages. The faulty condition gets activated by a change in the PMTU towards a host during the lifetime of a connection. This problem can become crucial and lead to the extensive loss of traffic in IPv6 networks, since in IPv6 packets do not get fragmented on the forwarding nodes due to performance considerations. The next paragraphs describe the outlined issues in detail.

The phenomenon in our case study is caused by the loss of packets at a router without any (ICMPv6) error notification being conveyed back to the sender host. We look at the specific case when packets are dropped at a forwarding node because of an incorrect PMTU (Path Maximum Transmission Unit), i.e. the packets of a flow are larger than the maximum size of a packet that can be transmitted. Such a situation can

occur in case of a link/node failure followed by the automatic (as done in OSPF) re-routing of the packets over a link with an MTU which is smaller than the initially obtained PMTU for a flow originating from an end system as illustrated in Figure 10 (the Gigabit Ethernet links have bigger MTU than the Fast Ethernet links). Usually, if a router fails to forward a packet with MTU greater than the one set for the egress interface for the packet, it is supposed to notify the sender of a flow via an ICMPv6 *Packet Too Big* message. Upon receiving such an ICMPv6 message, the sender is expected to adjust the PMTU of the corresponding flow in order to continue serving the end user's applications. However, it seems that administrators often suppress ICMP messages in their firewalls (security considerations). Therefore, this problem can easily occur whenever security concerns prevail inside an operator's network or simply in the case when an old (IPv4 relevant) firewall configuration has been adopted to an IPv6 network. The issue of ICMP message suppression is extremely critical in the context of IPv6, since ICMPv6 plays a significant role in IPv6, and suppressing ICMPv6 messages in routers could lead to major performance and connectivity issues, amongst others because intermediate nodes in an IPv6 network do not perform packet fragmentation due to performance considerations.

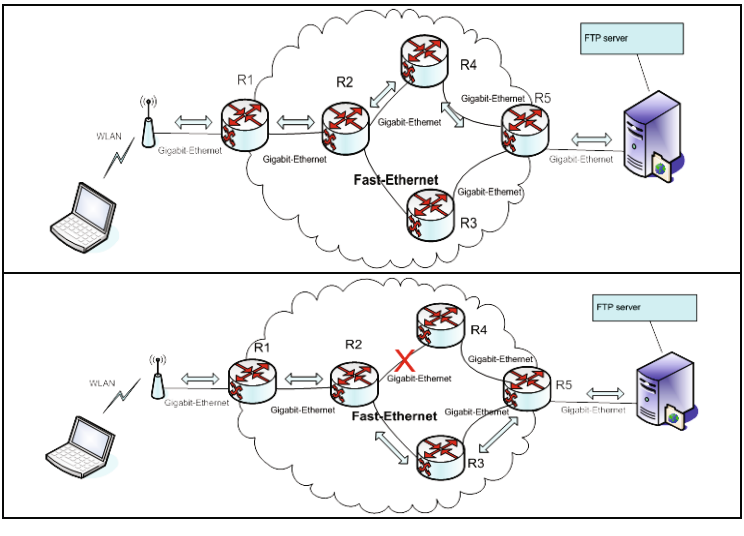

Fig. 10. Reference network and scenes for the IPv6 Black Hole case study

The reference network for our IPv6 PMTU scenario is illustrated in Figure 10. It consists of a WLAN access point which is connected to a service provider network of five routers with different types of links. On the services' side, there is an FTP server which is used by subscriber users to share files. Such a network can be seen as a part of a university campus network. Assuming that R2 is a router with a misconfigured firewall that suppresses ICMP messages, one can think of traffic between the FTP server and the host on the left over the path: **end system ↔ R1 ↔ R2 ↔ R4 ↔ R5 ↔ FTP server**. The flows running over this path would also have an established Path MTU of the size allowed by Gigabit Ethernet jumbo frames and WLAN at the same

time: 2346 bytes (the WLAN MTU). Assuming that at a particular point in time, the link **R2 ↔ R4** fails as illustrated in Figure 10, then the routing protocol would shift the traffic to the link **R2 ↔ R3**. This link has the Fast-Ethernet MTU of 1500 bytes. Indeed, the PMTU will go down from 2346 bytes to 1500 bytes. For that reason, R2 starts losing packets since no fragmentation is allowed on intermediate nodes in IPv6 because of performance issues. Because of R2 suppressing ICMP messages, the end system on the left won't get notified to readjust the size of the packets it sends out. This would normally lead to a time out of the connection on the transport layer in case of connection-oriented protocols. In the case of TCP, the connection could survive given that TCP PMTU discovery is implemented and activated on the end system, which is often not the case for the standard configuration of many operating systems. The application of UAFAReS in that context aims at extending the IPv6 capabilities of handling PMTU changes during the lifetime of a flow even in the presence of misconfigured firewalls, and eliminating the misconfigurations in the network. [RFC2923] provides packet flow descriptions that illustrate such a PMTU Black Hole as observed on an intermediate router, i.e. a router between one of the end systems and the black hole router. Such a router would be R1 in our case, because R2 is considered as the black hole router. [RFC2923] illustrates a case when large packets fail to traverse the network. Given that such symptoms are detected on the R1, the corresponding incident descriptions are submitted to the local UAFAReS repositories and in turn disseminated by the IDE to R2 and to the end system in order to facilitate resilient behaviors on these devices. After the FM_DE on R2 has been fed with the information regarding the detected symptoms, its FDLI functions component will perform Fault-Isolation based on the Causality Model stored in the local CMR repository and will convey the results of the Fault-Isolation process to the FRF module in order to execute a fault removal action on R2. We expect that the Autonomic Fault-Management control loop will be completed by an action reconfiguring the firewall in question such that ICMPv6 messages (according to the IPv6 security model described in [RFC4942]) can get through, and that the RS_DE (more specifically the FMF) on the end system (on the left) would readjust the PMTU towards the FTP server while trying to prolong the lifetime of the connection – this corresponds to the behavior specified in Figure 5. In that sense, the actions undertaken by UAFAReS will be of benefit for all transport layer communications, including connectionless UDP.

6 Concluding Remarks

In this paper, we presented the developments within the EFIPSANS [10] project towards the definition of a standardizable Reference Model for Autonomic Networking and Self-Management dubbed GANA Model, and the investigation of the interplay between Autonomic Fault-Management, Resilience and Survivability concepts and mechanisms towards the implementation of self-healing in autonomic networks. We presented our framework for achieving self-healing, called UAFAReS (Unified Architecture for Autonomic Fault-Management, Resilience and Survivability), which is based on GANA. UAFAReS was initially designed to implement resilience within a limited network scope, e.g. subnet, LAN, OSPF routing area. In this paper, we extend it, propose and investigate aspects and mechanisms on how collaboration between end

systems, access, edge and core network components can be facilitated and exploited in such a way that collaborative self-healing across the different domains is realized through the UAFAReS framework. Moreover, we presented some types of faulty conditions that can be resolved by the collaboration of UAFAReS instances inside the ISPs network, the edge, the access network and the subscribed end systems (user terminals). Finally, we looked into the details of a particular case study, showing how the collaboration of UAFAReS instances across different domains can enable overcoming potential problems with PMTU changes in an IPv6 network. Our future research efforts will be mainly concentrated on the implementation of the UAFAReS framework and the investigation of an extensive number of issues which can be addressed by UAFAReS mechanisms in enabling self-healing across different domains.

Acknowledgement. This work has been partially supported by EC FP7 EFIPSANS project (INFSO-ICT-215549).

References

1. Autonomic Computing: An architectural blueprint for autonomic computing, IBM White Paper (2006), http://www-01.ibm.com/software/tivoli/autonomic/
2. Chaparadza, R.: Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments. IEC Annual Review of Communications 61 (December 2008)
3. The FCAPS management framework: ITU-T Rec. M. 3400
4. Tcholtchev, N., et al.: Towards a Unified Architecture for Resilience, Survivability and Autonomic Fault-Management for Self-Managing Networks. To appear in the Proceedings of the 2nd Workshop on Monitoring Adaptation and Beyond MONA+
5. Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.N., Ganjali, Y., Diot, C.: Characterization of Failures in an Operational IP Backbone Network. IEEE/ACM Transactions on Networking 16(4), 749–762 (2008)
6. Touvet, F., Harle, D.: Network Resilience in Multilayer Networks: A Critical Review and Open Issues. In: The Proceedings of the First International Conference on Networking-Part 1, July 09-13, pp. 829–838 (2001)
7. Chaparadza, R.: UniFAFF: A Unified Framework for Implementing Autonomic Fault-Management and Failure-Detection for Self-Managing Networks. John Wiley & Sons, Chichester (2008)
8. Steinder, M., Sethi, A.S.: A survey of fault localization techniques in computer networks. Journal – Science of Computer Programming 53, 165–194 (2004)
9. Li, N., Chen, G., Zhao, M.: Autonomic Fault Management for Wireless Mesh Networks. Electronic Journal for E-Commence Tools and Applicatoins, eJETA (January 2009)
10. EFIPSANS project: http://www.efipsans.org/ (as of date September 17, 2010)
11. Shalunov, S., Carlson, R.: Detecting Duplex Mismatch on Ethernet. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 135–148. Springer, Heidelberg (2005)
12. Kompella, R.R., Yates, J., Greenberg, A., Snoeren, A.C.: Detection and Localization of Network Blackholes. In: The Proceedings of IEEE Infocom, Alaska, USA (May 2007)
13. Hubble: Monitoring Internet Reachability in Real-Time, http://hubble.cs.washington.edu/ (as of date July 12, 2010)
14. CIM, http://www.dmtf.org/standards/cim/ (as of date September 17, 2010)
15. ITU-X.733: Information Technology – Open Systems Interconnection – Systems Management: Alarm Reporting Function