Towards the Future Internet
G. Tselentis et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-539-6-105

Manageability of Future Internet Virtual Networks from a Practical Viewpoint

J. Rubio-Loyola[a,1], A. Astorga[b], J. Serrat[b], L. Lefevre[c], A. Cheniour[c], D. Muldowney[d], S. Davy[d], A. Galis[e], L. Mamatas[e], S. Clayman[e], D. Macedo[f], Z. Movahedi[f], G. Pujolle[f], A. Fischer[g] and H. de Meer[g]
[a] *CINVESTAV Tamaulipas,* [b] *Universitat Politècnica de Catalunya,* [c] *INRIA,* [d] *Waterford Institute of Technology,* [e] *University College London,* [f] *LIP6,* [g] *University of Passau*

Abstract. The Autonomic Internet project [1] approach relies on abstractions and distributed systems of a five plane solution for the provision of Future Internet Services (OSKMV): Orchestration, Service Enablers, Knowledge, Management and Virtualisation Planes. This paper presents a practical viewpoint of the manageability of virtual networks, exercising the components and systems that integrate this approach and that are being validated. This paper positions the distributed systems and networking services that integrate this solution, focusing on the provision of Future Internet services for self-configuration and self-performance management scenes.

Keywords. Virtualisation, Management Plane, Knowledge Plane, Service Enablers Plane, Orchestration Plane, Self-configuration, Self-performance

Introduction

Networks are becoming service-aware. The network's design is moving towards a different level of automation, autonomicity and self-management. The Future Internet (FI) will have many of its core features and functions based on virtualized resources [1]. As such management of Virtual Networks becomes a fundamental capability in Future Internet. The Autonomic Internet (AUTOI) solution [2] consists of distributed management systems described with the help of five planes: Orchestration, Service Enablers, Knowledge, Management and Virtualisation Planes. These distributed systems form a software-driven network control infrastructure that will run on top of all current networks and physical infrastructures, to provide an autonomic virtual resource overlay. The five planes gather observations, constraints and assertions, and apply rules to these to generate service-aware observations and responses in order to converge to optimal service deployments and optimal service maintenance.

A number of European projects are currently addressing Future Internet research problems that include, the design and validation of Autonomic Network Architectures [4], Biologically-inspired Autonomic Networks [5], Component-ware for Autonomic Situation-aware Communications and Services [6], Opportunistic Communications [7],

[1] On the leave from Universitat Politècnica de Catalunya, now at CINVESTAV Tamaulipas – Km. 5.5 Cd. Victoria-Soto La Marina, 87130 Tamaulipas Mexico. jrubio@tamps.cinvestav.mx

new Future Internet architectures [8] and self-* properties in wireless networks [9]. The Autonomic Internet (AUTOI) approach consists of orchestrated autonomic management systems [3]. The blocks of the AUTOI approach are graphically represented in Fig. 1. Each autonomic system contains policy, context, discovery and other supporting services. Autonomic management systems are federated through orchestration services of the same nature i.e. policy, context, discovery, etc.

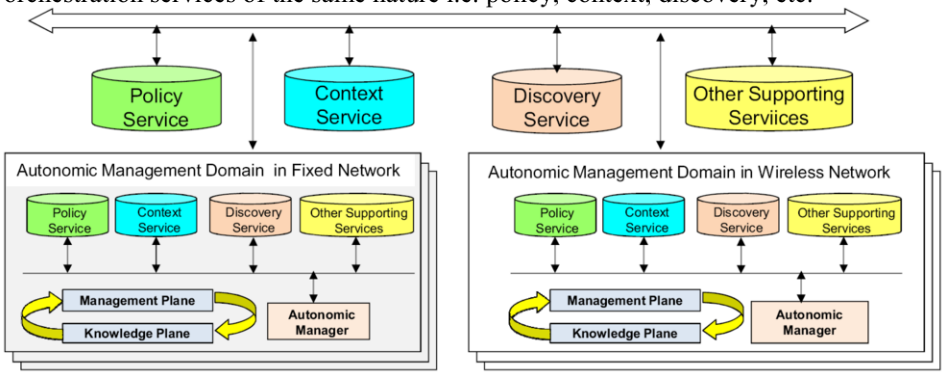

Figure 1. Functional Blocks Architecture of the Autonomic Internet (AUTOI) Approach

1. Future Internet Application Scenario

Consider an application service that provides large amounts of diverse-nature information, such as multimedia files or information to users with different profiles. Such information stored on servers distributed in a given geographic area. Users use different types and terminal manufacturers and their position changes continuously. In order to cope with different types of users, especially users with security requirements, the system will establish virtual private networks (VPN) probably with encryption.

Users access the network by means of various wireless access technologies. We assume a mobile Internet environment offered by the family of IEEE standards. Specifically, it is assumed that there are access points for local area network IEEE 802.11 (Wi-Fi), wide area network fixed and mobile (IEEE 802.16 and IEEE 802.16e respectively) and regional area network IEEE 802.22 (WRAN). Change between access technologies (Vertical Handover) should be automatic and transparent to the user, depending on factors like: offered service characteristics; user's profile; signal intensity; time response of the switching process; position, speed and direction of movement of the users; traffic, load and applications supported by the sub-networks; cost and grade of service.

The above scenario will need a management system that ensures a transparent and fast delivery of the service that the clients have contracted. This will require among others, the following tasks: Deploy appropriate management systems that can support the provision of the services within the appropriate resources; Setting up VPNs on demand, depending on the user and network's context; Support for automatic vertical handover to ensure the best possible access to the network; Support for the management communication overlays' setting up with uniform distribution of traffic load; Reaction to Quality of Service degradations identifying their causes and restoring the services concerned in a transparent manner.

2. Functional Components of the AUTOI Approach

This Section describes the functional blocks of the AUTOI solution.

2.1. Autonomic Network Programming Interface (ANPI)

The Autonomic Network Programming Interface (ANPI) implements the functionality of the Service Enabler Plane (SEP) [2]: new programmatic enablers to allow code to be deployed and then executed or activated on the Network entities, and a safe and controlled deployment of any kind of services (resource- and consumer-facing services). The ANPI provides a programming interface that is used by the AMSs and DOCs (both described below), in order to deal with the deployment and administration of services and management components. The ANPI consists of a Service Deployment Daemon, which can be run in any virtual machine/router and/or a physical machine. Once it is instantiated, the ANPI takes care of commands to enable in-network programmability.

2.2. Autonomic Management System (AMS)

Autonomic Management Systems implement functionalities of the Management Plane and Knowledge Plane [2]. It uses a dedicated set of models and ontologies. Mapping logic enables the data stored in models to be transformed into knowledge and combined with knowledge stored in ontologies, to provide a context-sensitive assessment of the operation of one or more virtual resources. The AMS interface with one or more Distributed Orchestration Components (DOC, described later), which enable the former to federate with other AMS as to provide end-to-end context-aware services. This component reacts to context-change events that can be for example, on-demand service requests, statistical fluctuations of QoS, faults taking place in the network, and so forth. This component is currently being validated with a Context-aware Policy-Based System with ontology-based reasoning capabilities.

2.3. Distributed Orchestration Component (DOC)

The Distributed Orchestration Component (DOC) provides a set of framework network services and partially implements the functionality of the Orchestration Plane [2]. Framework services provide a common infrastructure that enables all components in the system managed by the Orchestration Plane to have plug-and-play and unplug-and-play behaviour. Applications compliant with these framework services share common security, metadata, administration, and management services.

2.4. Context & Information Service Platform (CISP)

The Context & Information Service Platform (CISP) is a narrow functionality Knowledge Plane [2]. It is an infrastructure for collection, processing and dissemination of context information, as a support for the deployment, evolution and autonomy of communication services and applications (context-aware applications and services). Context-aware control functions is essential in guaranteeing both a degree of self-management and adaptation as well as supporting context-aware communications that efficiently exploit the available network resources.

2.5. Virtualisation System Programmability Interfaces (vSPI)

The vSPI [2] is used to enable the Orchestration Plane (and implicitly the AMSs under control of the Orchestration Plane) to govern virtual resources, and to construct virtual services and networks that meet stated business goals having specified service requirements. The vSPI contains the "macro-view" of the virtual resources that a particular Orchestration Plane governs, and is responsible for orchestrating groups of virtual resources in response to changing user needs, business requirements, and environmental conditions.

2.6. Virtualisation Component Programmability Interfaces (vCPI)

The vCPI [2] is an interface that allows managing Virtual Resources (e.g. Virtual Routers) from within a Physical Component. It provides methods that support the self-configuration for Virtual Links and Virtual Routers Management. Links Management's methods provide support to instantiate, remove, and modify virtual links. Virtual Routers Management's methods are used for example to register, start and shutdown a Virtual Router. Other supporting methods are used for example to get associative identifiers with Virtual Routers and Virtual Links on a component respectively. It also implements methods that allow monitoring information and that are the key instruments to enforce the self-performance and -fault management functions. A selection of these methods are: cpuUsage, availableCPUs, availableRAM, totalRAM, assignedRAM, stateCurrent (e.g. state of a Virtual Router), usedBW, pktsLost, etc. Methods are invoked by the AMSs, the DOCs or even the ANPIs.

2.7. Model-based Translator (MBT)

The Model-based Translator (MBT) [2] is a software package that translates commands and messages from a technology independent "network" language to device specific commands and configurations, and also can convert device specific monitoring messages into a technology independent language. The MBT uses a detailed information model of communications networks in combination with model-to-text translations templates to carry out the translations. It is intended for use by network management software designed to manage heterogeneous networking equipment and services without understanding the associated data models. This component is used by the AMSs to decouple their management decisions from the network and resource technology in which the services are deployed.

3. Self-configuration Management Support

This section describes with the help of illustrative interacting methods the support that the above software components provide in self-configuration scenes.

3.1. On-demand Setting-up of Virtual Infrastructures

Virtual infrastructures are set up prior service deployment. The components' interactions for this initial step are shown in Fig. 2.

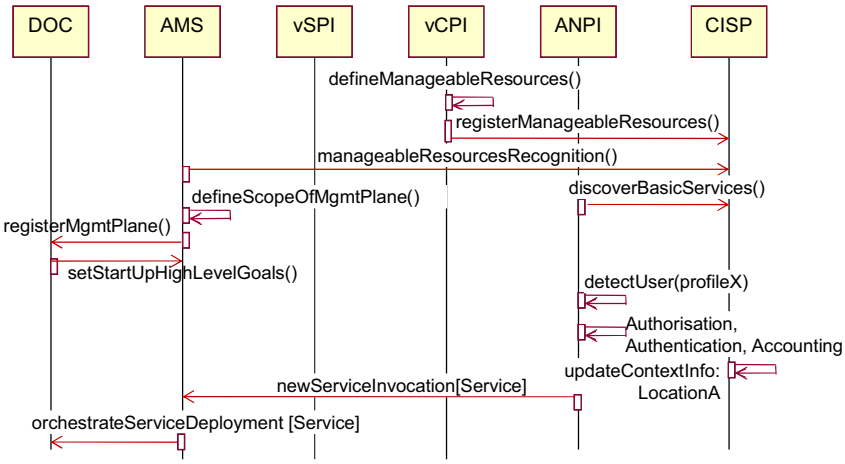

Figure 2. Relevant Interactions for Setting-Up of Virtual Infrastructures

The availability of resources is defined by the Virtualisation Component Programmability Interfaces (vCPI) that control the physical resources (*defineManageableResources()* in Fig. 2). Once these are defined, the management entities and components need to have references to these available resources (*registerManageableResources()*). This information is registered in the Context & Information Service Platform (CISP). The Autonomic Management Systems (AMSs) in turn recognize these resources (*manageableResurcesRecognition()*), so that they can define the scope of the resources they control (*defineScopeOfMgmtPlane()*). The Autonomic Network Programming Interface (ANPI) performs the discovery of basic services that reside in the manageable resources (*discoverBasicServices()*) and registers them in the CISP. At this stage the CISP is updated with resources and basic services that are subject to management tasks for further usage.

The next step is the registration of AMSs in the Distributed Orchestration Component (DOC) (*registerMgmtPlane()*)for orchestration purposes. Based on the scope of the AMSs the DOC pushes initial high-level start up goals (*setStartUpHighLevelGoals() in Fig. 2*), so that the AMSs can define the internal mechanisms and policies with which they would control their resources and basic services during the start up of their network infrastructures. High-level start-up goals are intended to create a cooperative environment in which the AMSs could share information with other domains, all in all, orchestrated by the DOC.

Consider an end-user approaching the wireless environment in Location A as graphically shown in Figure 3, where a unique wireless access controller is able to connect the end-user. The ANPI (Autonomic Network Programming Interface) detects the user profile (*detectUser(profileX) in Fig. 2*) and makes sure that the end-user is subscribed to an AUTOI service (*Authorization, Authentication, Accounting (AAA) in Fig. 2*). Successful invocations are updated in the CISP (*updateContextInfo: LocationA in Fig. 2*), namely user's Location A. The ANPI communicates with the AMSs that a new service is being invoked and that it needs to be provisioned with specific guarantees (*newServiceInvocation[Service] in Fig. 2*). As the AMS Wireless is not capable of providing the content to the end-user on its own, the DOC is required to orchestrate the deployment of the content service with the help of other AMSs that it has subscribed (*orchestrateServiceDeployment[Service] in Fig. 2*).

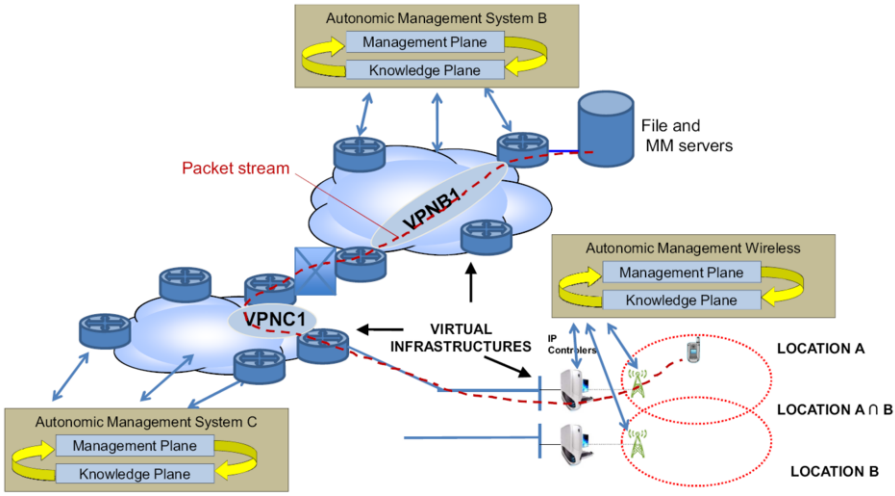

Figure 3. End-user Service Configuration

End-user service deployments can be started on demand by the end user contacting an Autonomic Management System (AMS) domain, or contacting the user interface of the Distributed Orchestration Component (DOC). In any case, a number of negotiation rounds among Fixed and Wireless AMS systems are coordinated by the DOC using its behaviours' tools with the aim to define what services are provisioned by which AMS. The services are anything needed to provision the application services to the user: content services, connectivity services, management services, storage services, access services, and so forth. Negotiation-wise [2] the DOC supports a service marketplace where the AMSs offer and publish their services (*svceOffer&Publish:[Services]* Fig. 4).

The negotiation phase is supported by the core functionalities of the Distributed Orchestration Components, taking into account the self- governance, -distribution and -federation behaviours of the AMSs. Negotiations are triggered by the DOC and are represented as a loop between all functional entities (*coordinateNegotiation:[Service]* in Fig. 4). The aim is to determine the specific AMS that will provision the end-user application service and the management services and resources that will be compromised for such a purpose. The DOC starts the deployment of the AMSs (*startAMSDeployment [AMSs, HLGoals,Svc]*) supported by the ANPI to deploy the corresponding management components into the virtual resources (*newAMSDeployement[AMS, HLGoals, Svc]* in Fig. 4). It is worth mentioning that the configuration and maintenance of application services passes through an effective and systematic refinement process of the high-level directives that the DOC provides to the AMSs (*deriveEnfPolicies(AMS, HLPoliciesAndGoals):ContextInfo, EnfPolicies*) for each AMS in Fig. 4). The AMS administrator instantiates a policy continuum and refines the corresponding policies that will be used to govern the management entities.

A data model is created with the help of the vCPI interfaces for each AMS and it is registered in the CISP platform (*createDataModel(AMS, ContextInfo)* in Fig. 4). It describes the characteristics of all virtual resources and virtual links/topologies that are used for the provision of the application services. It also describes the associations between Components and virtual resources to minimize impact on the to-be-deployed and recently instantiated AMSs. To create this data model the local context environment of the AMSs is used.

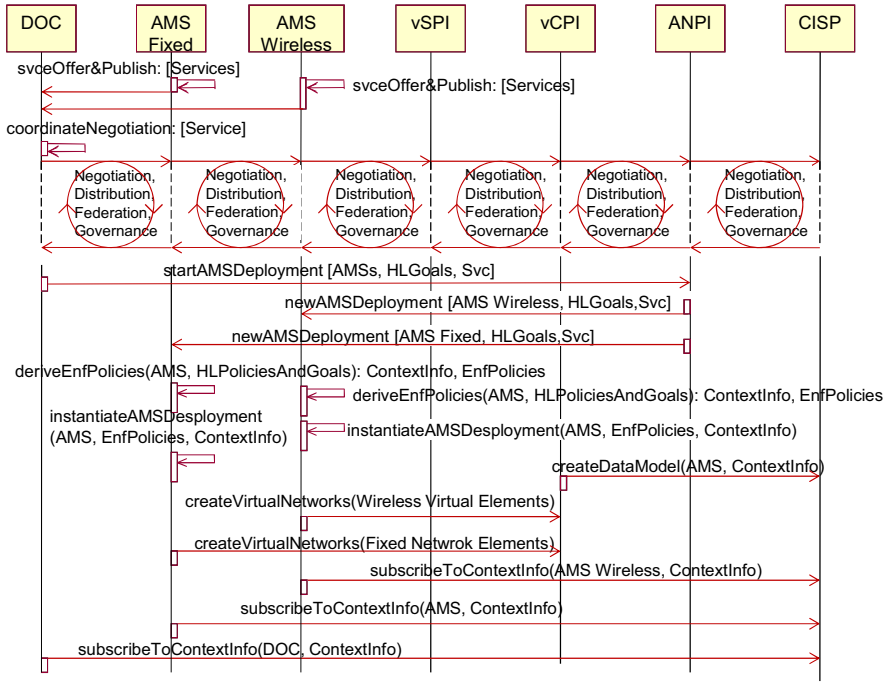

Figure 4. Interactions for the negotiation and setting up of virtual infrastructures

The AMSs now have all the information they need to create Virtual Infrastructures that will support the services they compromised during the negotiation. The AMSs enforce configuration policies to create the Virtual Routers and Links and to activate the network and resource facing services as well as content services (*createVirtualNetworks(Virtual Elements)* in Fig. 4, see also *Virtual Infrastructures* pointers in Fig. 3). Each AMS exhibits self-governance properties. AMSs have internal policies for the creation of virtual infrastructures which control virtual and non-virtual resources, e.g. CPU processing, memory assignment, and distribution of traffic load for the services, etc. Having the AMSs created the virtual infrastructures, AMSs subscribe to relevant context information identified earlier (*subscribeToContextInfo (AMS, ContextInfo)* in Fig. 4) with the aim to react to statistical changes of the to-be-configured service. Similarly, the DOC subscribes to relevant context information that will drive the inter-AMS domain management functions, e.g. setting up, activation and release of virtual links between AMS domains.

3.2. On-demand Deployment of End-user Services

This Section describes the actual deployment of the end-user service.

In our running scenario the setting up of the virtual infrastructure is preceded by an end-user approaching the wireless environment in Location A. This is updated in the Context & Information Service (CISP) Platform (*updateContextInfo:LocationA* in Fig. 5). The Autonomic Management System (AMS) Wireless senses the end-user location and grants it access. This is sensed by the AMS_C (*contextSensing(user):LocationA* in Fig. 5) as this information is part of the context information to which it subscribed earlier. AMS_C executes the management tasks to fulfill the end-user requirements

according to its profile and the high-level goals compromised with the Distributed Orchestration Component (DOC). The result of these management tasks is the setting up of a VPN between an edge of its management domain (defined by the DOC in the High-Level goals) and the edge router with which the wireless access controller A will hand in the content to the end user (*setUpVPNC1(ingress, egress)* in Fig. 5). The setting up of VPNC1 is achieved with the help of the vCPI and the ANPI components. Similarly, AMS_B sets up a VPN for the same purposes (*setUPVPNB1(ingress, egress)*). Finally, the DOC interconnects the domains (due to security concerns) with the help of the Virtualisation System Programmability Interfaces (vSPI) and the Autonomic Network Programming Interface (ANPI). The end-user service is finally deployed and this information is updated in the CISP (*deploymentUpdate()* in Fig. 5) by the ANPI. Eventually, the availability and the context information of the deployment is made available by the CISP to all other planes (*updateContextInfo:serviceDeployed, LocationA* in Fig. 5). The DOC subscribes to inter-AMS-domain contextual information for coordination and maintenance purposes (*subscribeContextInfo(OP_ContextInfo)* in Fig. 5). The actual End-to-End configuration of the service is graphically depicted in Figure 3.

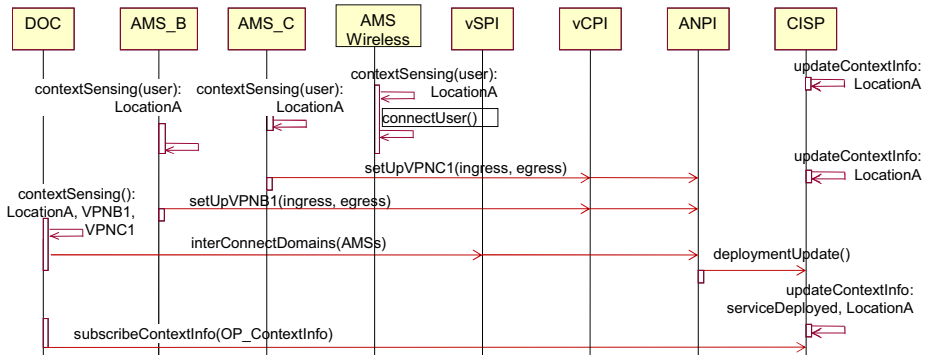

Figure 5. Interactions for the Deployment and Activation of Service

4. Self-performance Management Support

Autonomic Management Systems (AMSs) exhibit self-governance properties while enforcing their management decisions, aligned to their internal policies, and also to the previously agreed goals.

Consider that AMS_B in our scenario has subscribed to track packet losses in two Virtual Routers (VR) VRa1 and VRb1 that support the end-user content service of our use case. Packet losses are retrieved periodically from the Virtualisation Component Programmability Interfaces (vCPI) and updated periodically in the Context & Information Service (CISP) Platform. Specifically for VRb1 these are represented with interaction *getPerformanceInfo(VRb1):packetLoss* in Fig. 6. When VRb1 has packet losses higher than a given threshold (*updateContextInfo:VRb1 PktLssUP_THR* Fig. 6) the CISP issues an alarm that is basically a context change event indicating that such a threshold has been crossed upwards. This is sensed by AMS_B (*contextSensing(VRb1): pktLssUP_THR* in Fig. 6) which eventually triggers a recovery process in component B (*reassignResources(C): ComponentB* in Fig. 6). This process entails a reassignment of

resources that results in the migration of the end-user content service traffic from VRb1 to a less used VR to reduce congestion of the former (*migrateService(contentService)*). AMS_B reserves some spare capacity for eventual QoS degradation recovery. The migration is enforced by the AMS_B in the vCPI with the programmability support of the ANPI. Other resources reassignment options include increasing the CPU for the affected router or a migration of part of the supported services to spare resources. Finally, AMS_B updates its subscription to context variables monitoring aligned to the updated assignment and service deployment (*subscribeToContextInfo(AMS, ContextInfo)* in Fig. 6).

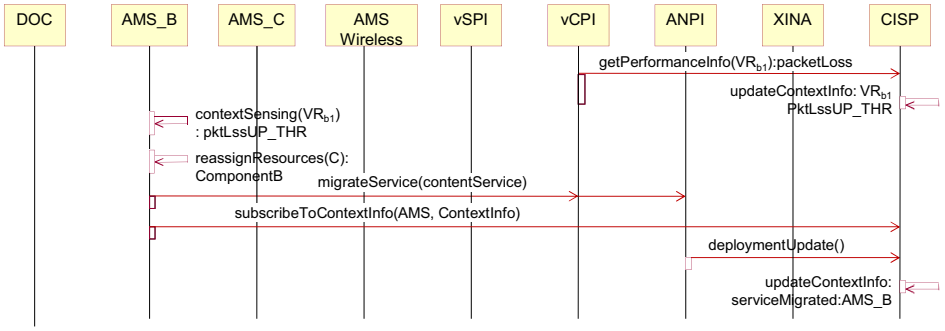

Figure 6. Selected Intra-AMS Self-performance Management Interactions

The ultimate goal of the AUTOI components is to maintain services taking the best proactive actions in case the service performance is degraded. Figure 7 shows the result of a self-performance management scene like the one described above. The picture depicts that around time 10:38, a deployed service has experienced a sensible reduction of service rate, namely experiencing service degradation. The proactive actions taken by the AUTOI components result in the partial migration of the service to a new virtual infrastructure maintaining the service rates as when the service was deployed originally. This way, the self-performance management capabilities have reacted to a sensible self-performance scene with the best option available.

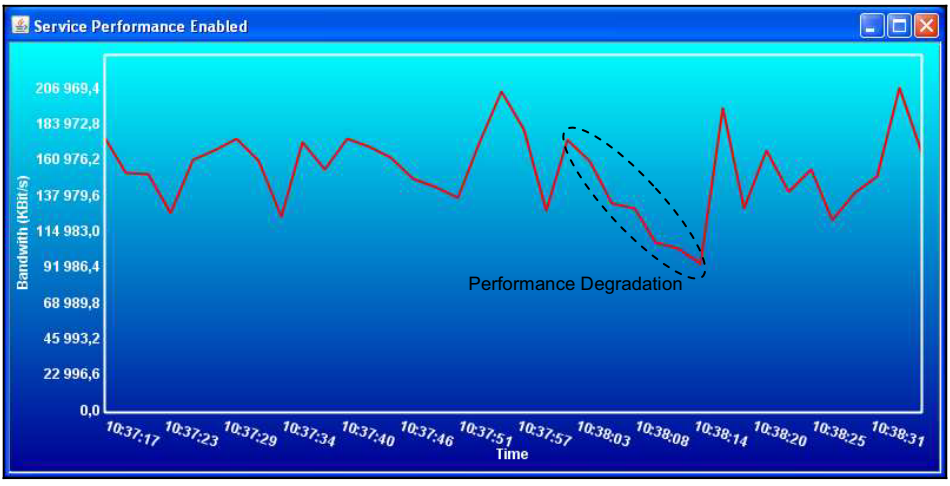

Figure 7. Rate of Deployed Service in Self-performance management execution

5. Conclusions

This paper has presented the manageability of virtual networks from a practical viewpoint according to the AUTOI approach [1]. We have described an application scenario and the functional AUTOI components that support the provision of Future Internet Services. We have also provided the description of the components interactions, for which illustrative descriptions of self-configuration and self-performance management scenarios have been described. In addition, we have provided the result of a self-performance management scene of a real service deployment as part of the validation efforts being carried out in our project. The software components described in this paper are released as Open Source components for experimental purposes [2]. Future work will be mainly devoted to validate our approach in large-scale deployments. Large scale deployments of the AUTOI concept will be executed in the GRID5000 (French Nationwide Experimental Platform embedding 5000 cores) test-bed and relevant results of this effort will be further disseminated to the Future Internet research community.

We strongly believe that Future Internet service provisioning relies on embedded in-network management functionalities and orchestrated management systems that can take advantage of virtualisation and programmability principles that can support dynamic and statistical changes of the network conditions, insuring stability and convergence to optimal service provisioning results. We expect that the components' descriptions, use case scenarios and partial results described in this paper encourage Future Internet research community to target these and other research challenges towards conceiving a Future Internet with guaranteed service provisioning capabilities.

References

[1] A. Galis et al. "Management and Service-aware Networking Architectures (MANA) for Future Internet Position Paper: System Functions, Capabilities and Requirements" - Invited paper IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09) 26-28 August 2009, Xi'an, China; http://www.chinacom.org/2009/index.html.

[2] European Union IST 7th Framework Programme AUTOI Project (Autonomic Internet) Deliverables http://ist-autoi.eu

[3] A. Galis, S. Denazis, A. Bassi, A. Berl, A. Fischer, H. de Meer, J. Strassner, S. Davy, D. Macedo, G. Pujolle, J. R. Loyola, J. Serrat, L. Lefevre, A. Cheniour – "Management Architecture and Systems for Future Internet Networks" – in "Towards the Future Internet – A European Research Perspective" – ISBN 978-1-60750-007-0, pp350, April 2009, IOS Press, http://www.iospress.nl/

[4] EU IST FP6 ANA Project "Autonomic Network Architectures" http://www.ana-project.org/

[5] The EU IST BIONETS http://www.bionets.eu/ project "Biologically-inspired autonomic Networks and Services" http://www.bionets.eu/

[6] IST FET CASCADAS Project "Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services" http://www.cascadas-project.org

[7] The EU IST FP6 HAGGLE project (An innovative Paradigm for Autonomic Opportunistic Communication) http://www.haggleproject.org/index.php/Main_Page

[8] EU IST FP7 TRILOGY Project "Re-Architecting the Internet: An Hourglass control Architecture for the Internet, Supporting Extremes of Commercial, and social and Technical Control" http://www.trilogy-project.org

[9] EU IST SOCRATES project "Self-Optimisation and self-ConfiguRATion in wirelEss networkS" http://www.fp7-socrates.org