Manageability of Future Internet Virtual Networks from a Practical Viewpoint

J. Rubio-Loyola[1] – CINVESTAV Tamaulipas Mexico {jrubio@tamps.cinvestav.mx}, A. Astorga, J. Serrat – Universitat Politècnica de Catalunya Spain {aastorga, serrat}@tsc.upc.edu, L. Lefevre, A. Cheniour – INRIA France {laurent.lefevre, abderhaman.cheniour}@ens-lyon.fr, D. Muldowney, S. Davy - Waterford Institute of Technology Ireland {dmuldowney, sdavy}@tssg.org, A. Galis, L. Mamatas, S. Clayman - University College London U.K. {a.galis, l.mamatas, sclayman}@ee.ucl.ac.uk, D. Macedo, Z. Movahedi, G. Pujolle - LIP6 France {Daniel.Macedo, Zeinab.Movahedi, Guy.Pujolle}@ lip6.fr, A. Fischer, H. de Meer- University of Passau Germany {andreas.fisher, demeer}@uni-passau.de

***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 the OSKMV plane approach and that are being validated. This paper positions the distributed systems and networking services that specialise the OSKMV solution in the provision of Future Internet services for self-configuration and self-performance management scenes.**

***Index Terms*—Virtualisation, Management Plane, Knowledge Plane, Service Enablers Plane, Orchestration Plane, Self-configuration, Self-performance**

I. AUTONOMIC INTERNET APPROACH

The Future Internet (FI) will have many of its core features and functions based on virtualized resources [3]. As such management of Virtual Networks becomes a fundamental capability in Future Internet. The Autonomic Internet (AutoI) solution [1] consists of distributed management systems described with the help of the OSKMV 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 application service physical infrastructures, to provide an autonomic virtual resource overlay. The OSKMV 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. The blocks of the AUTOI approach are graphically represented in Fig. 1. AUTOI consists of orchestrated autonomic management systems. Each autonomic system contains policy, context, discovery and other services. They are federated through orchestration services of the same nature (policy, context, etc).

II. 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, which is stored on servers distributed in a given geographic area, should be provided to users with certain levels of service. The users are not associated to any permanent server; use different types and terminal manufacturers and their position changes continuously. Indeed, these are mobile users, although the service can be provisioned to fixed users. 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.

[1]On the leave from Universitat Politècnica de Catalunya, Spain

Users access the network by means of various wireless access technologies. In particular, we assume an environment of mobile Internet offered by the family of IEEE standards. Specifically, it is assumed that there are points of access 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). Users can access through any of these technologies as long as their terminals can support them. Moreover, when the users enter the coverage area of another access point, there must be a change between access technologies (Vertical Handover) automatic and transparent to the user, depending on various factors including: specific service characteristics being offered; 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 various sub-networks; cost and grade of service.

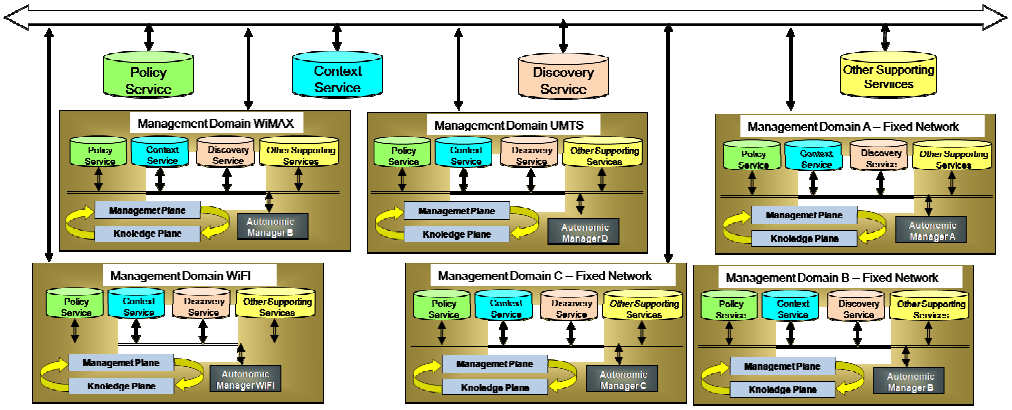

Figure 1 - Functional Blocks Architecture of the Autonomic Internet Approach

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 making, among others, the following management tasks: Deploy an appropriate management system that can support the provision of the services within the appropriate resources; Setting up VPNs on demand, depending on the context of the user and the network; Support mechanisms for automatic vertical handover to ensure the best possible access to the network at any moment; Support for the management communication overlays' setting up with uniform distribution of the traffic load injected to network resources; Reaction to Quality of Service degradation, identifying the causes and taking corrective and preventive actions to solve it; Reaction to failures in the network infrastructures, identifying their causes and restoring the services concerned in a transparent manner; Reaction to failures in the network infrastructures, identifying their causes and restoring the services concerned in a transparent manner. The Fig. 2 shows the overall picture of the management systems that would support the above application scenario.

III. FUNCTIONAL COMPONENTS OF THE AUTOI APPROACH

This section describes the functional blocks of the AUTOI solution.

A. 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-facing and/or consumer-facing services). The goal is to propose a set of primitive functions dealing with all service requests of the Autonomic Internet Project planes. This programming interface 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 in a Service Deployment Daemon, which can be run in any virtual machine/router and/or a physical machine. Once, instantiated, the ANPI takes care of commands to enable in-network programmability.

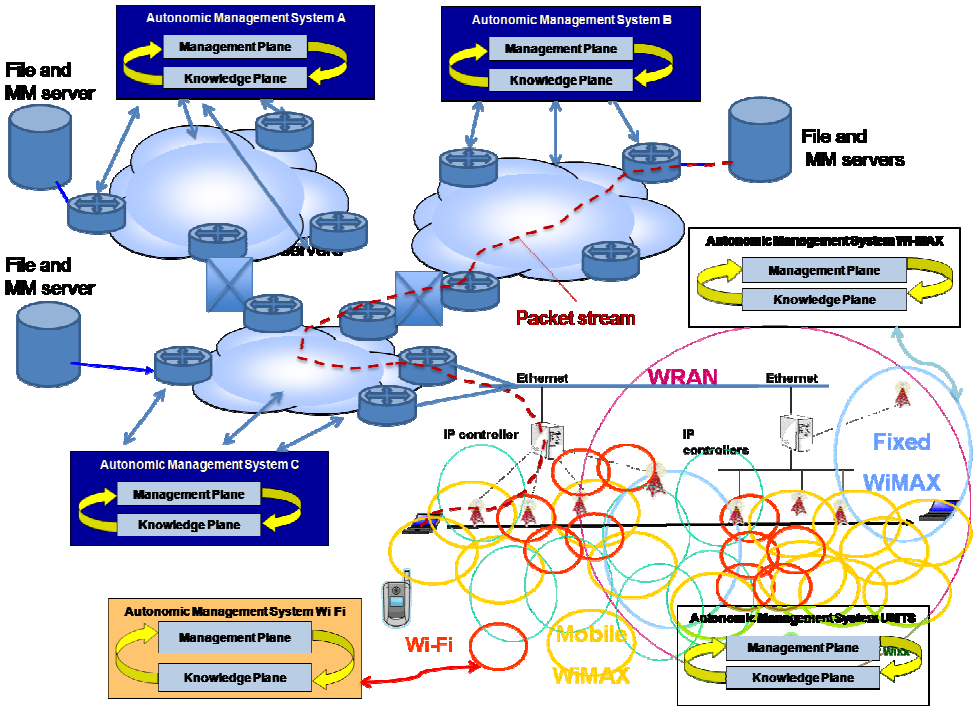

Figure 2 – Overall Picture of the Application Scenario

B. Autonomic Management System (AMS)

Autonomic Management Systems implement functionalities of the Management Plane and a 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) that 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.

C. 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.

D. 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). The goal of making the control functions of Autonomic Networks context-aware is therefore 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.

E. 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.

F. Virtualisation Component Programming Interface (vCPI)

The vCPI [2] is an interface that allows managing Virtual Resources (e.g. Virtual Routers) from within a Physical Component in the context of the AutoI architecture. Currently the vCPI supports 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 like *getVMList* and *getVLList* are used to get a list of identifiers associated with Virtual Routers and Virtual Links on a component respectively. The vCPI provides several 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.

G. 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.

IV. SELF-CONFIGURATION MANAGEMENT SUPPORT

This section describes in practical terms the support that the above software components provide in self-configuration scenes. This description is presented with the help of illustrative interacting methods between components.

A. 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. 3.

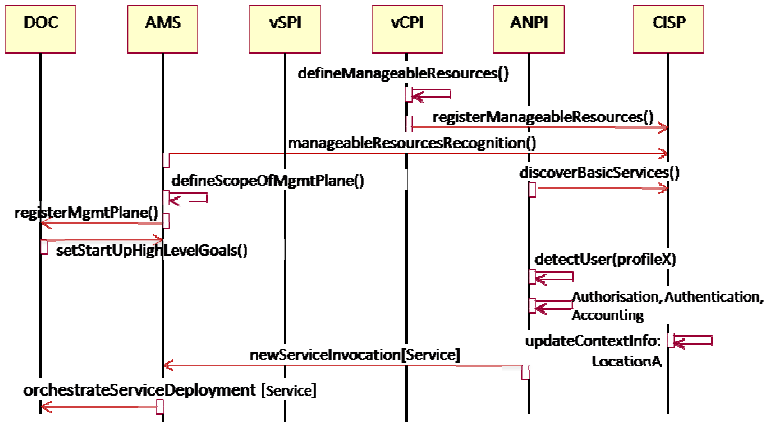

Figure 3 – Relevant Interactions for Setting-Up of Virtual Infrastructures

Initially, the availability of resources is defined by the vCPI interfaces that control (interaction *defineManageableResources()*) the physical resources. Once the manageable resources are defined, the management entities and components that will take actions over them need to have references to these available resources (interaction *registerManageableResources()*). This information is registered in the CISP platform. The management plane via the AMS in turn recognizes the above resources (interaction

manageableResurcesRecognition()), so that it can define the management scope of the resources it is able to control (*defineScopeOfMgmtPlane()*). The 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 DOC (*registerMgmtPlane()*), with which the DOC takes control for orchestrating the underlying AMS. At this stage the AMSs are aware of the resources and basic services that are subject of management tasks. In the case of our application scenario (see Fig. 2), three AMSs control the fixed network, namely AMS_A, AMS_B and AMS_C. For simplicity, consider the wireless network is controlled by a generic Wireless AMS. Based on the scope of the AMSs the DOC pushes initial high-level start up goals (*setStartUpHighLevelGoals()* in Fig. 3), 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 that requests a content service through the wireless AMS. The content is not physically located close to the wireless AMS and hence it cannot be provisioned by the wireless AMS on its own. The setting up of virtual infrastructures aimed at bringing the content from the resources controlled by other AMSs is the next step towards deploying the end-user content service. The following elaborates on this part.

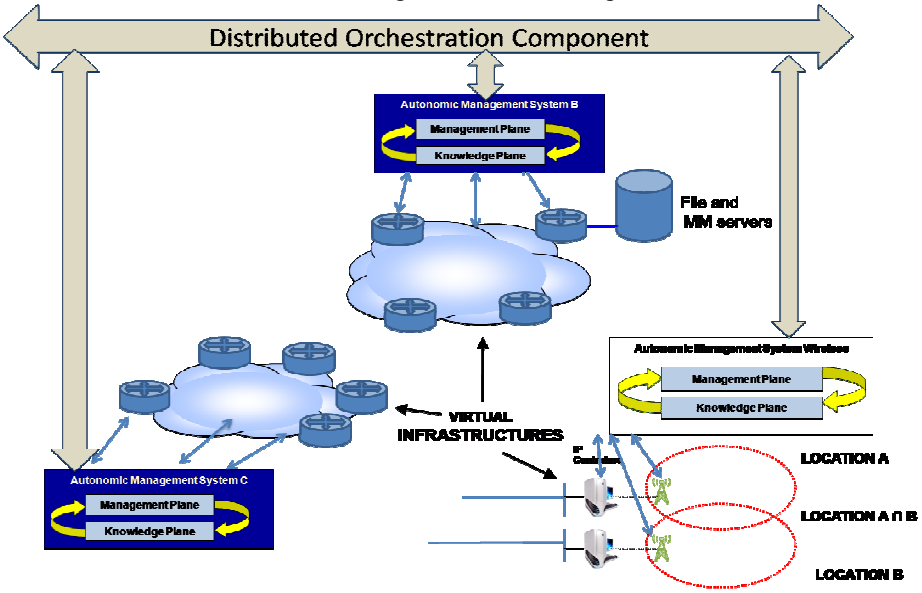

Figure 4 –Virtual Infrastructures and Management Systems

Consider an end-user approaching the wireless environment in Location A (Fig. 4), namely a location in which 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. 3) and makes sure that the end-user is subscribed to an AUTOI service (*Authorization, Authentication, Accounting (AAA)* in Fig. 3) in which he can express his needs. After a successful invocation, the context of the user is updated in the CISP (*updateContextInfo: LocationA* in Fig. 3), in this case, updating his current position, namely to Location A. Following on with this, 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. 3). 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. 3).
The DOC is aware of the end-user service requirements and the availability of the Wireless and Fixed AMS. It has been delegated to coordinate the deployment of the end-user service taking into account that it has a number of AMSs under its control. AMSs are registered to further participate in the negotiation and federation processes.

End-user service deployments can be started on demand by the end user contacting an AMS domain, or contacting the user interface of the DOC. In any case, a number of negotiation rounds among the 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 provided by the AMSs at this stage 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 is responsible to support a service marketplace where the AMSs offer and publish their services (*svceOffer&Publish: [Services]* in Fig. 5). The DOC provides the means to facilitate negotiations as AMSs can belong to several different operators, and they could send and receive information in different formats, mechanisms or protocols.

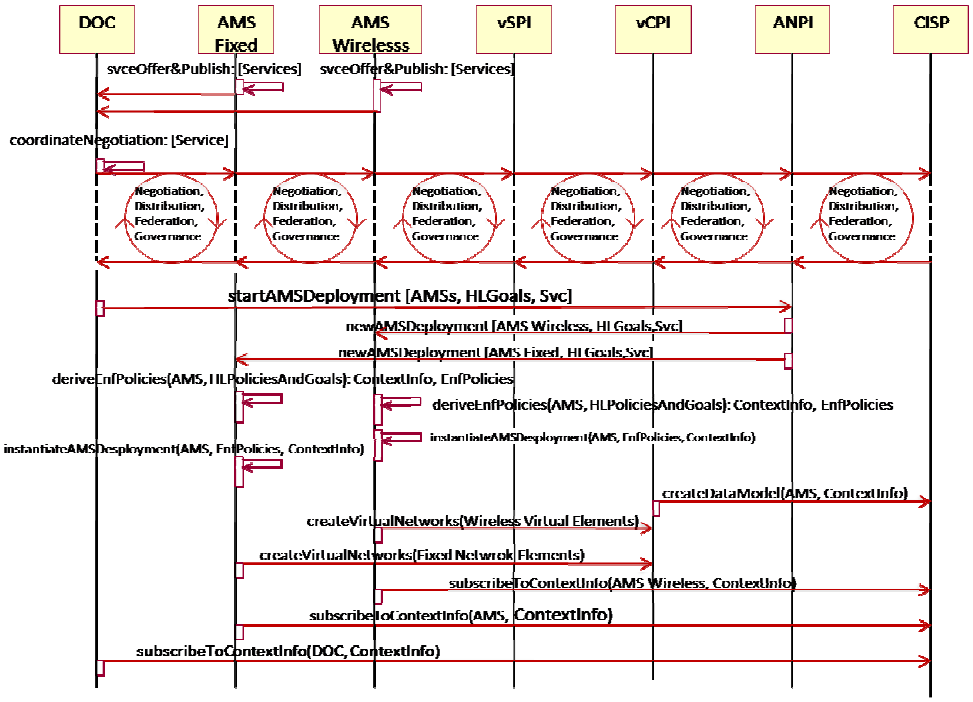

Figure 5 –Interactions for the negotiation and setting up of virtual infrastructures

The negotiation phase is supported by the core functionalities of the Distributed Orchestration Components. The negotiation is represented as a loop between the functional entities is triggered by DOC (*coordinateNegotiation:[Service]* in Fig. 5) and they are carried out taking into account the self- governance, -distribution and -federation behaviours of the AMSs, which are coordinated by the DOC. The aim of these interactions 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]* in Fig. 5) supported by the ANPI to deploy the corresponding management components into the virtual resources (*newAMSDeployement[AMS, HLGoals, Svc]* in Fig. 5). 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. 5). For this, the AMS administrator instantiates a policy continuum and refines the corresponding policies that will be used to govern the management entities through autonomic control loops.

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)*). It describes the characteristics of all virtual resource 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.

The AMSs now have all the information needed 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. 5, see also Virtual Infrastructures pointers in Fig. 4). 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. 5) 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.

B. 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 CISP platform (*updateContextInfo:LocationA* in Fig. 6). The AMS Wireless senses the end-user location and grants it access. This is sensed by the AMS_C (*contextSensing(user):LocationA* in Fig. 6) as this information is part of the context information to which it subscribed earlier. AMS_C executes the management tasks to fulfil the end-user requirements according to its profile and the high-level goals compromised with the 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. 6). 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)* in Fig. 6). Finally, the DOC interconnects the domains (due to security concerns) with the help of the vSPI and the ANPI. The end-user service is finally deployed and this information is updated in the CISP (*deploymentUpdate()* in Fig. 6) 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. 6). The DOC subscribes to inter-AMS-domain contextual information for coordination and maintenance purposes (*subscribeContextInfo(OP_ContextInfo)* in Fig. 6). The configuration of the service is graphically depicted in Figure 7.

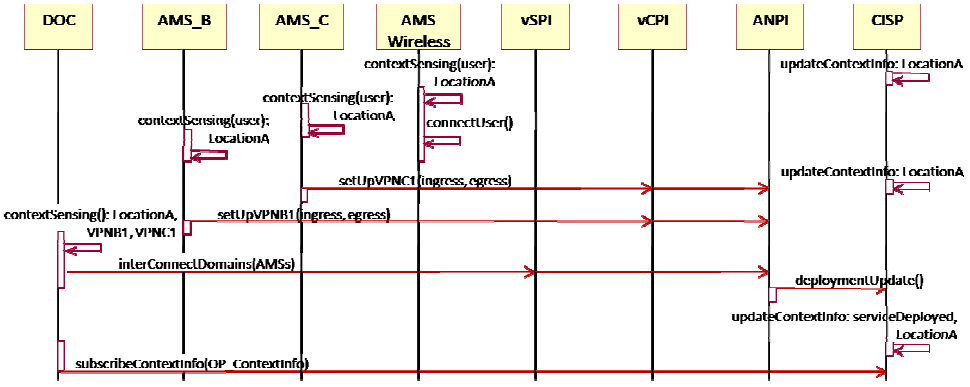

Figure 6 –Interactions for the Deployment and Activation of Service

V. SELF-PERFORMANCE MANAGEMENT SUPPORT

A. Intra-AMS Self-performance Management

The AMSs exhibit self-governance properties in enforcing management decisions aligned to their internal policies, all in all, aligned to fulfil previously agreed goals. Each AMS takes care of the network and resource services compromised in the federated services.

Consider that AMS_B has subscribed to track packet losses in two virtual routers VRa1 and VRb1 that support the end-user content service of our use case. Packet losses are retrieved periodically from the vCPI to the CISP. Specifically for VRb1 these are represented with interaction *getPerformanceInfo(VRb1):packetLoss* in Fig. 8. When VRb1 has packet losses higher than a given threshold (*updateContextInfo:VRb1 PktLssUP_THR* in Fig. 8) 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. 8) which eventually triggers a recovery process in component B (*reassignResources(C): ComponentB* in Fig. 8). 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. 8).

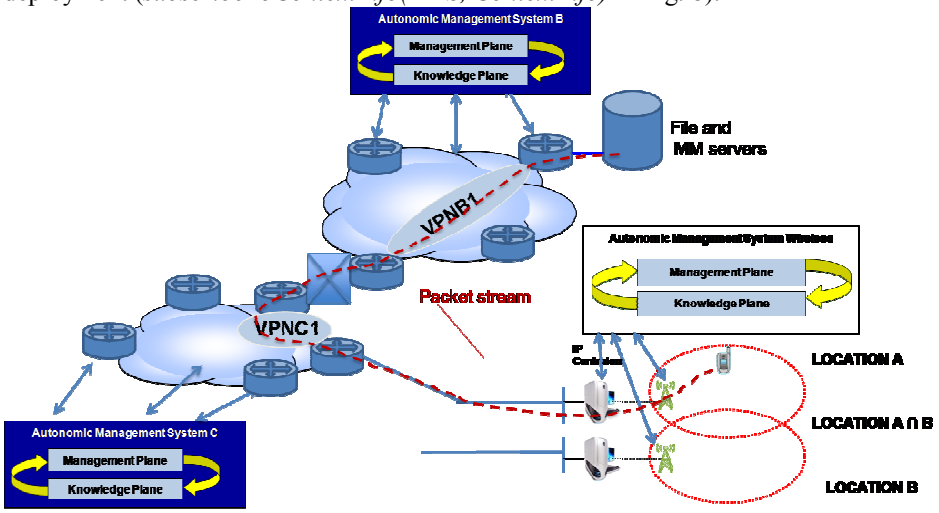

Figure 7 –End-user Service Configuration

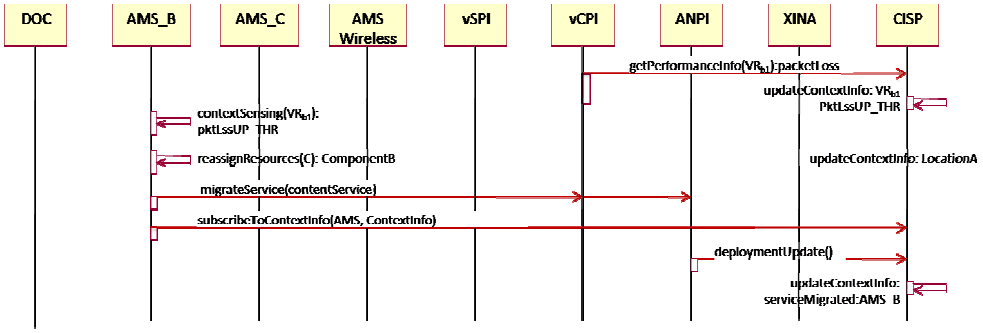

Figure 8 –Interactions of Intra-AMS Self-performance Management

B. Inter-AMS Self-performance Management

In the above scene AMS decisions are transparent to the DOC. The DOC has a broader scope of the service deployment and its maintenance. It is in charge of critical issues like activation and maintenance of inter-AMS domain connections between the AMS_B, AMS_C and AMS Wireless in our running use case, and other inter-domain issues like the coordination of actions to prevent service degradation or interruption that lay out of the scope of the AMSs. This section summarises relevant self-performance management interactions in our running example. These interactions are graphically depicted in Figure 9.

In the deployment and operation of a service, the DOC is responsible for the orchestration of the AMSs. Meanwhile, the AMSs manage the network infrastructures that make up the service and ensure that they operate properly. Consider that AMS_B, AMS_C and

AMS_Wireless in our use case are operated by different operators. In this case it is not possible for the DOC to monitor and/or configure any of the equipment on the orchestrated AMS domains. Instead, it should rely on the AMSs to do so. Following on with the intra-AMS self-performance management scene described earlier, consider that at some point of the service management life cycle, AMS_B enforces management actions that go against the high-level goals defined for the services deployed. Namely, consider that AMS_B has run out of resources and that it autonomously reassigns resources in component A (*reassignResources(C): ComponentA* in Fig. 9), followed by a migration of our application content service towards another virtual infrastructure in such component (*migrateService(contentService)* in Fig. 9). Under these conditions the QoS of the content service of our use case is violated. Further management actions in AMS_B are unable to avoid service degradation as it has run out of resources to commit to the QoS with the agreed levels of quality. This situation is updated by the AMS_B in the CISP (*contextUpdate: QoSDeviation(serviceID)* in Fig. 9) since it is in its interests that the clients obtain the stipulated QoS levels.

The DOC senses this deviation (*contextSensing(AMS)QoSDeviation* in Fig. 9) and takes actions to solve the deviation. Upon the perception of QoS degradation, the Orchestration Plane contacts each of the AMSs involved in the service provision in order to identify the source(s) of the QoS degradation. If the involved entities reach a consensus on the corrective action to be performed through a new negotiation process, then the process is completed. However, in case a consensus cannot be reached, the DOC may request a change in the deployment of the service (i.e. migrate the service to another AMS domain), replace one or more AMSs, re-negotiate the QoS of the service, re-negotiate the high-level goals of the AMS, or even close down the service. These (re)negotiation rounds are depicted with the Negotiation, Distribution, Federation, Governance interaction rounds in Figure 9. The result of these interactions in our use case is the deployment of a new AMS_A which will be in charge of providing the content service with QoS guarantees, substituting AMS_B as graphically shown in the bottom part of Figure 9.

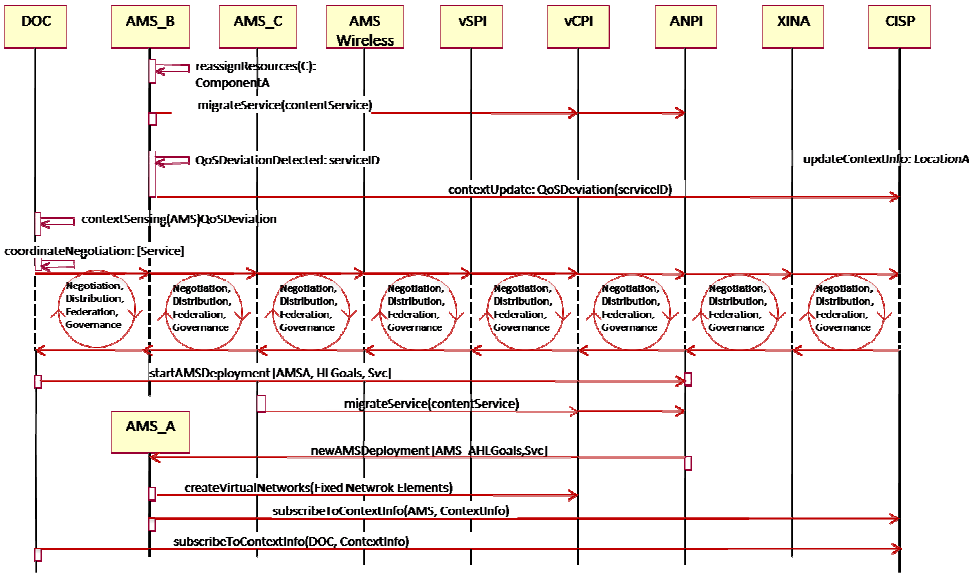

Figure 9 –Interactions of Inter-AMS Self-performance Management

The ultimate goal of the AUTOI components is to maintain services taking the best proactive actions in case the service performance is degraded. Following on with our running use case consider that AMS_B in Fig. 9 is able to support the service with guaranteed lower service rates. Figure 10 shows the result of a real migration of a service this kind in terms of served service rates. The picture depicts that around time 10:09.46, the service experiences 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 in this case with lower guaranteed service rates. This way, the self-performance management capabilities have reacted to a sensible self-performance scene with the best option available.

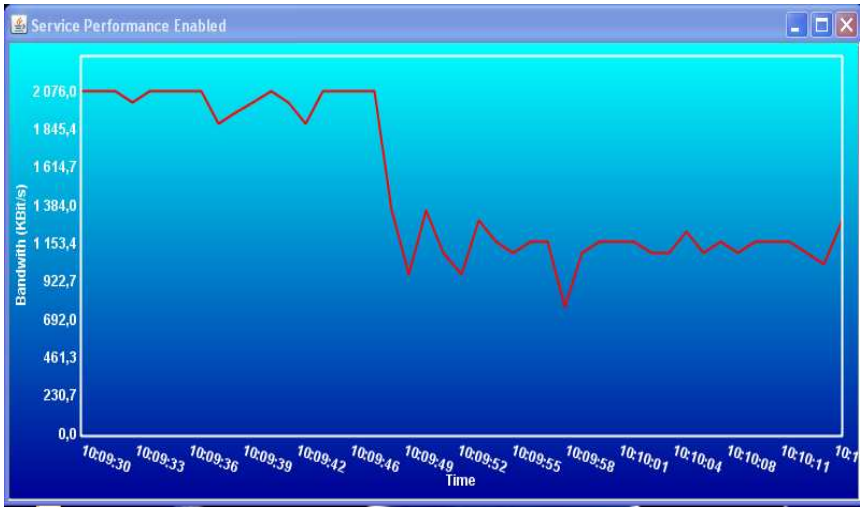

Figure 10 –Service Rate of a Deployed Managed Service

VI. 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 [1]. 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] Autonomic Internet Project http://ist-autoi.eu/autoi/

[2] EU Autonomic Internet Project Deliverables http://ist-autoi.eu

[3] A. Galis, H. Abramowicz, M. Brunner, D. Raz, P.r Chemouil, J. Butler, C. Polychronopoulos, S. Clayman, H. de Meer, T. Coupaye, A. Pras, K. Sabnani, P. Massonet, S. Naqvi "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

[4] 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/