Scalable Service Deployment on Software Defined Networks

J. Rubio-Loyola[1], A. Galis[2], A. Astorga[3], J. Serrat[3], L. Lefevre[4], A. Fischer[5], A. Paler[5], H. de Meer[5]
CINVESTAV Tamaulipas[1], University College London[2], Universitat Politècnica de Catalunya[3], INRIA[4], University of Passau[5]

Abstract— **The network of the future will require a greater degree of service-awareness, and an optimal use of network resources. This paper presents the architectural design developed in the AutoI project for an open software-defined network infrastructure that enables the composition of fast and guaranteed services in an efficient manner and the execution of these services in an adaptive way taking into account better shared network resources provided by network virtualisation. Validation results are provided with special emphasis on service deployment scalability over virtualized network infrastructures.**

Index Terms—**Autonomic Internet, network virtualisation, service enablers, self-management.**

I. Introduction

It is becoming accepted that Future Networks should be service and management aware [1], which includes (among others) the following aspects:

• Delivery of content and service logic with consumers' involvement and control.

• Fulfilment of business, Quality of Service (QoS) and Service Level Agreements (SLA).

• Optimisation of the network resources during service delivery.

• Composition and decomposition on demand of control mechanisms and network domains.

Conversely, deployed services [2] in the Future Networks should be network-aware. Network-awareness means that the consumer-facing and the resource-facing services are aware of the properties, the requirements, and the state of the network environment. This enables services to self-adapt according to changes in the network context.

In the recent years, network virtualisation techniques have gained a lot of attention due to their flexibility for creating computing clouds and for creating separate and independent virtual networks on top of physical network infrastructures.

Virtual networks abstract away the complexity of the underlying infrastructure. They are characterized in the literature either as a main means to test new network architectures or as a crucial component of future networks [3], [4], [5]. Multiple logical networks can co-exist above the same physical substrate. They can take the form of virtual private networks [6], active and programmable networks [7] overlay networks [8] or virtual networks [9]. The virtual nodes and links form a virtual topology over the underlying physical network.

Virtual Networks are a collection of virtual nodes connected together by a set of virtual links to form a virtual topology, which is essentially a subset or an aggregation of the underlying physical topology. In such networks, links and nodes may be reconfigured quickly and may be, for example, powered down to save energy or the node may be redeployed to a different logical area of the network. Virtual networks aim at better utilization of the underlying infrastructure in terms of (i) reusing a single physical or logical resource for multiple other network instances, or (ii) to aggregate multiples of these resources, in order to obtain more functionality, such as providing a pool of resources that can be utilized on demand. As an example, virtual networks can be aggregated (or federated) together. Such an approach requires aggregation and dissolution of control, data, and information planes, which is a challenging problem.

Virtualized network environments are highly dynamic [4] as links and nodes may be reconfigured quickly. Virtual routers may migrate on-demand, as in [3], based on resource availability, in order to save energy or to follow the physical location of the users. Nodes may also move logically (i.e., not physically). Virtual network aggregation or dissolution triggers changes in the virtual topology, i.e., virtual network embedding [9]. It is obvious that management of virtual networks is challenging, since it is necessary to manage this complex functionality. Manageability and service deployment is considered to be the biggest concern for network virtualization [4].

This paper describes the architectural model and validation results of the EU Autonomic Internet AutoI project [10], which proposes open-source software defined network (i.e. Network Cloud). It is a self-managing overlay of virtual resources that can span across heterogeneous physical networks. All the components of the AutoI architecture were developed as open source components available from [10]. Validation results were performed by exercising the open source components on 3 physical networks ranging from 4 - 5,000 nodes. Two types of validation results based on large-scale experiments are the focus of this paper as follows:

• Built-in network management, specifically self-management functionality for service- and networking-awareness.

• Large scale provisioning and deployment of both,

application services and management services over virtual infrastructures.

This paper is structured as follows. Section II presents the AutoI architectural framework and its relevant systems. Section III presents validation results. Section IV provides some technical discussion. Section V describes related work. Section VI concludes the paper.

II. AutoI Architectural Framework and Systems

The AutoI framework consists of a software defined network described with the help of five abstractions – the OSKMV planes: Orchestration Plane (OP). Service Enablers Plane (SP), Knowledge Plane (KP), Management Plane (MP) and Virtualisation Plane (VP). At the physical level, they are embedded on network hosts, devices, and servers within the network. The main purpose of the OSKMV planes is to make the Future Networks capable of self-knowledge, and ultimately fully self-managing. The AutoI architectural model is shown in Figure 1 and a description of the major elements follows. Open source platform components developed in support of the AutoI architecture, which are available from [10], are indicated as (*) in the following section.

A. Orchestration Plane Overview

The purpose of the Orchestration Plane is to govern the behaviour of the system in response to changing context and in accordance with applicable business goals and policies. It supervises and integrates all other planes' behaviour ensuring integrity of the future Internet management operations.

The Orchestration Plane is a control framework into which any number of components can be plugged into in order to achieve the required functionality. These components could have direct interworking with control algorithms, situated in the control plane of the Internet (i.e. to provide real time reaction), and interworking with other management functions (i.e. to provide near real time reaction).

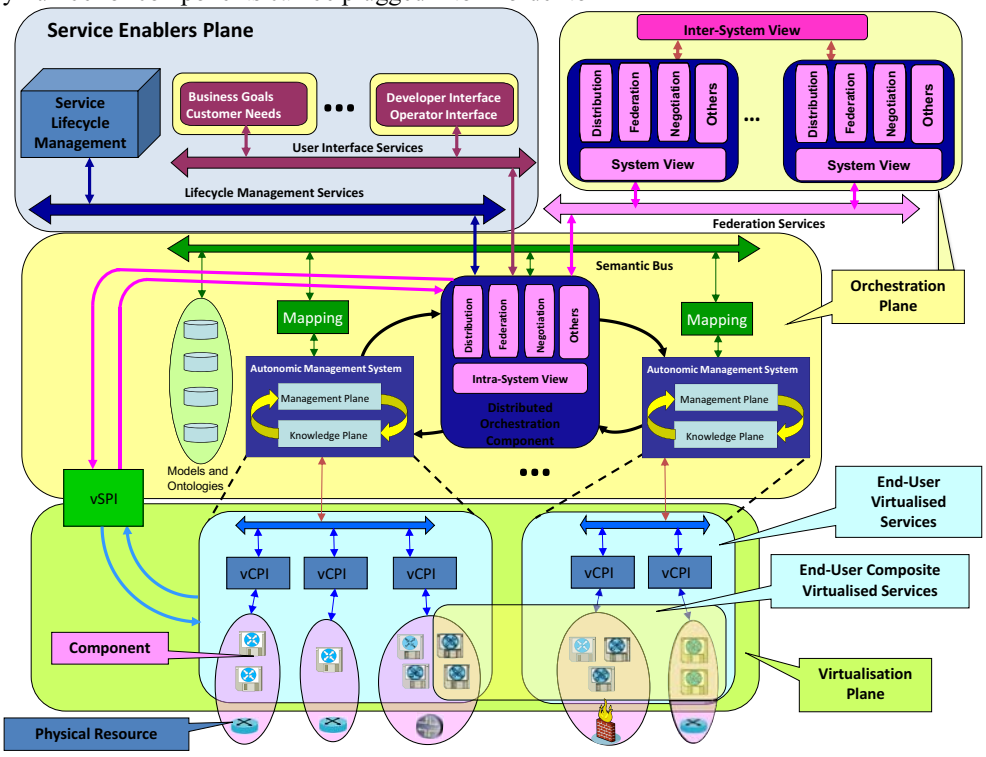

Fig. 1. AUTOI Architectural Model- software defined network

In practical terms, the Orchestration Plane controls one or more Autonomic Management Systems (AMS – described later). It acts as control workflow for AMSs ensuring their bootstrapping, initialisation, dynamic reconfiguration, adaptation and contextualisation, optimisation, organisation, closing down. It is functionally integrated by one or more Distributed Orchestration Components (DOC) and a dynamic knowledge base consisting of a set of models and ontologies and appropriate mapping logic and buses. DOCs can federate via buses. The internal design details of the Orchestration Plane can be found in [11].

B. Autonomic Management System (AMS)

A key advantage of the AutoI architecture is that it can provide a programmable mix of isolation and sharing of network resources. A key advantage of separating the control and data planes is to provide increased isolation for an application or set of applications.

Each Autonomic Management System (AMS) is part of the Management Plane (described later) and it includes interfaces to a dedicated set of models and ontologies and interfaces to one or more Distributed Orchestration Components. 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. Another set of interfaces enables framework services, such as directory services, naming, federation, and others, to be used by the AMS.

C. Distributed Orchestration Component (DOC)

The Distributed Orchestration Component (DOC) (*) provides a set of framework network services. Framework services provide a common infrastructure that enables all AMSs controlled by the Orchestration Plane to have (un)plug-and-play behaviour. Applications compliant with these framework services share common security, metadata, administration, and management services. The

DOC enables the following framework network services to the AMSs under its control:

• Federation: each AMS is responsible for its own set of virtual and non-virtual resources and management services that it governs. Federation enables a set of domains to be combined into a larger domain, where selected functionality of each constituent domain contributes to the overall functionality of the larger domain.

• Negotiation: each AMS advertises a set of capabilities (i.e., services and/or resources) that it offers for use by other components in the Orchestration Plane.

• Distribution: this service enables tasks to be split into parts that run concurrently on multiple AMS controlled by a DOC, or even across multiple DOCs. This function ensures that AMSs with different implementations and functionality can collaborate.

• Governance: this service enables each AMS to be able to operate in an individual, distributed, or collaborative mode. Business goals, service requirements, context, capabilities and constraints are all considered as part of the decision making process.

• Intra-System View: this service provides an overall, composite view of the system as seen by the components within a DOC.

• Inter-System View: this service provides an overall, composite view of collaborating DOCs, as in a multiple domain system.

D. Service Enablers Plane Overview

The Service Enablers Plane (SP) consists of functions for the automatic (re)deployment of new management services, protocols as well as resource-facing and end-user facing services. It includes enablers to allow code to be executed on the network entities. This functionality is implemented by the ANPI (Autonomic Network Programming Interface) (*)[10], which supports large scale network programmability in deployed virtual networks. The safe and controlled deployment of new code enables new services to be activated on demand, and to be made available to both, management and orchestration planes for the benefit of service-awareness. This approach has the following advantages:

- Automatic service deployment allowing a significant number of new services to be offered on demand.
- Flexible network configuration capabilities.
- Special management functions and services can be easily enabled locally for testing purposes before they are automatically deployed network-wide.
- Flexible support for service migration, both for consumer-facing and the resource-facing services.

E. Knowledge Plane Overview

The Knowledge Plane was proposed by Clark et al. [13] as a new dimension to a network architecture, contrasting with the data and control planes; its purpose is to provide knowledge and expertise to enable the network to be self-monitoring, self-analysing, self-diagnosing and self-maintaining.

AutoI introduces a narrow functionality Knowledge Plane (KP), consisting of models and ontologies, to provide increased analysis and inference capabilities. AutoI's KP brings together widely distributed data collection, wide availability of that data, and sophisticated and adaptive processing or KP functions, within a unifying structure. This brings order and meets the policy, scaling and functional requirements of a global network. The main KP components are a Context and Service Information Platform (CISP) (*) [10] and ontologies, which enable the analysis and inferencing capabilities. The CISP provides:

• information life cycle management (storage, aggregation, transformations, updates, distribution) of all information and context in the network addressing the Internet's size/scope.

• responsiveness to requests made by the AMS;

• triggers for the purpose of contextualisation of AMS (supported by the context model of the information model);

• support for robustness enabling the KP to continue to function as best possible, even under incorrect or incomplete behaviour of the network itself;

• support of virtual networks and virtual system resources in their needs for privacy and other forms of local control, while enabling them to cooperate for mutual benefit in more effective network management.

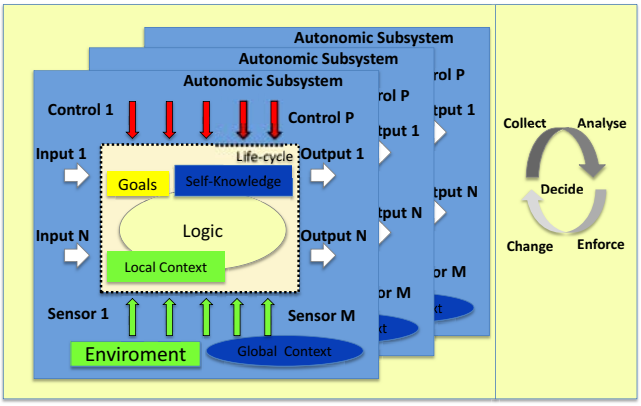

Fig. 2. Autonomic Control Loops

F. Management Plane Overview

The Management Plane (MP) governs all virtual resources, performing decisions on their optimal placement, function and continuous migration. The functionality of the MP is implemented by the AMS (*). The MP functionality is subject to constraints determined by the Orchestration Plane. The MP is designed to meet the following objectives and functionality:

• Embedded (Inside) Network functions: The majority of management functionality is embedded in the network and it is abstracted from the human activities. The AMSs run on execution environments on top of virtual networks and systems, which run on top of all current network (i.e. fixed, wireless and mobile networks) and service physical infrastructures.

• Aware and Self-aware functions: it monitors the network and operational context as well as internal operational network state in order to assess if the network current behaviour serve

its purposes.

• Adaptive and Self-adaptive functions: It triggers changes in network operation (state, configurations, etc.) as a result of changes in network and service context.

• Automatic self-functions: It enables self-control (i.e. self-FCAPS, self-*) of its internal network operations, functions and state. Manual/external input is provided in the setting-up of the business goals and other unavoidable functions.

• Extensibility functions: It adds new functions without disturbing the rest of the system (Plug and Play dynamic programmability of management functions and services). The AMSs that implement the functionality of the MP and the KP are designed to follow the autonomic control loops (collect, analyse, decide, enforce, change [14]) depicted in Figure 2.

G. Virtualisation Plane Overview

One of the key requirements that differentiate AutoI from other systems is its emphasis on virtualisation of resources and services. AutoI uses platform virtualisation to provide virtual services and resources. Platform virtualisation separates an operating system from its underlying platform resources; resource virtualisation abstracts physical resources into manageable units of functionality. For example, a single physical resource can appear as multiple virtual resources (e.g., the concept of a virtual router, where a single physical router can support multiple independent routing processes by assigning different internal resources to each routing process); alternatively, multiple physical resources can appear as a single physical resource (e.g., when multiple switches are "stacked" so that the number of switch ports increases, but the set of stacked switches appears as a single virtual switch).

AutoI extends contemporary virtualisation approaches and aims at building an infrastructure in which virtual machines can be dynamically relocated to any physical node or server regardless of location, network and storage configurations and administrative domain.

The Virtualisation Plane (VP) consists of software mechanisms to treat selected physical resources as a programmable pool of virtual resources that can be organised by the Orchestration and Management Planes into appropriate sets of virtual resources to form components (e.g., increased storage or memory), devices (e.g., a switch with more ports), or even networks. The organisation is done in order to realise a certain business goal or service requirement. Two special interfaces, called the vSPI and the vCPI (Virtualisation System Programming Interface) and Virtualisation Component Programming Interface, respectively assess the basic functionality of the Virtualisation Plane, for which a brief description is given hereafter.

H. vSPI (Virtualisation System Programmability Interface)

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. The low-level configuration (i.e., the "micro-view") of a virtual resource is provided by the vCPI, as explained in the next section.

The vSPI (*) is responsible for determining what portion of a component (i.e., set of virtual resources) is allocated to a given task. This means that all or part of a virtual resource can be used for each task, providing an optimised partitioning of virtual resources according to business need, priority and other requirements. Composite virtual services can thus be constructed using all or part of the virtual resources provided by each physical resource.

The vSPI monitors the "macro-level" status of the virtual resources that it governs. This is different from the vCPI, which monitors "micro-level" status of the virtual resources that it configures. For example, the vSPI collects global information about available physical resources. When an AMS receives requests to instantiate a new virtual resource it contacts the vSPI to determine which physical resources can be used. The vSPI then instructs the vCPI to instantiate the virtual resources on the correct physical resource for such purpose. The vSPI informs the AMS when the virtual resource is ready for use, and the vCPI informs the AMS when each virtual resource has been successfully reconfigured.

I. vCPI (Virtualisation Component Programming Interface)

The vCPI (virtual Component Virtual Interface) (*) is a modular, scalable and communication protocol-agnostic, system for monitoring and managing virtual resources. It operates locally; for each component of a physical network there is an embedded vCPI that is operating with third-party software by using a request/response mechanism. It is used for constructing, modifying and managing virtual networks (VN) consisting of virtual links (VL), virtual routers (VR) and routing services (RS). This enables the Autonomic Management System (AMS) to manage the physical resource, and to request virtual resources to be constructed from that physical resource via the vCPI. The AMS sends device-independent commands to the vCPI, which are translated into device- and vendor-specific commands that reconfigure the physical resource and manage the virtual resources provided by that physical resource. The vCPI also provides monitoring information from the virtual resources back to the AMS that controls that physical resource. Note that the AMS is responsible for obtaining management data describing the physical resource.

The vCPI is responsible for providing dynamic management data to its governing AMS that states how many virtual resources are currently instantiated, and how many additional virtual resources of what type can be supported. The vCPI needs to be aware of the structural information of the relations among virtual resources, and therefore, a discovery mechanism is included to inspect the contents of a physical component and map it to a data structure.

More details of the AutoI open source software defined network components that integrate the five-plane AUTOI approach introduced earlier are provided in [10] [12].

III. Practical Approach and Validation Results

This section tests the AutoI framework and the open source components, which were installed and run on 3 physical testbed networks ranging from 4 to 5,000 nodes. It provides relevant results emphasising in the creation of virtual networks, service deployment and scalability aspects. The section concentrates on wired physical networks. The interested reader will find a description of the AutoI support for mobile users in [10].

A. Creation of Virtual Networks

In AutoI, virtual networks can be set up by means of administrative decisions, programmed at a given time, triggered by events like threshold crossings, or they can be created on demand.

This section demonstrates the AutoI context-aware, on-demand, scalable creation of functional virtual networks. The term functional implies that virtual networks are ready to support application-services deployment, having all networking-services (e.g. routing services) deployed and configured properly. For this purpose we have used a physical infrastructure consisting of four physical components (2 Quad Core AMD Opteron 2347H CPUs and 32 GB RAM, running Linux, XEN or Qemu and Open SSH).

In this test setting the autonomic loop's *collect* part (see figure 2 for details of this loop) is exercised by the Context Information Services Platform (CISP) that manages context information and triggers notifications about changes corresponding to demand requests for virtual networks. Demand request correspond to end-points of connectivity with specific QoS constraints (e.g. throughput). CISP nodes are deployed in each physical node and also in each virtual router as soon as it is created. The collected data is made available to all AutoI components along the life cycle of the virtual infrastructure.

The autonomic loop's *analyse* and *decide* parts are implemented by the AMSs and the DOCs, which analyse context changes (e.g. end point network requests) and evaluate the conditions under which context changes occur, and eventually decide on appropriate actions that would fulfil their business goals. Business goals in this test case correspond to specific characteristics of the to-be-created virtual network topology. In this test scene a network topology that emulates the current (2010) topology of the German X-WiN network [15] was used. Such network consists of 60 nodes interconnected by 80 links, representing sites all over Germany as graphically depicted in Figure 3. The virtual routers are spread among the four physical components, 15 VRs are created on each component. Decisions are taken and result in concrete configurations that will need to be enforced via the vSPIs, vCPIs and ANPI platforms.

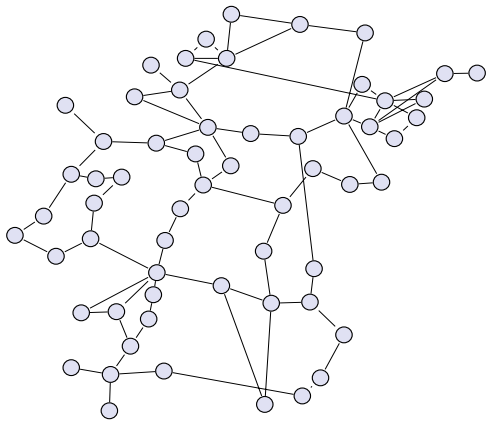

Fig. 3. 60-Virtual Router and 80-Virtual Link Virtual Network Created in Four-Component-physical Test-bed

The autonomic loop's *enforce* parts are assessed by the vSPIs, vCPIs and ANPIs. They are aimed at executing the appropriate commands to configure the Virtual Network (VN). In practical terms, creating a VN is a two-step process:

1) Creation/start-up of Virtual Routers (VRs) and creation of Virtual Links (VLs) attached to the former (*enforced* by the vSPIs and vCPIs.

2) Deployment of the networking-facing services (e.g. routing services) that would support the virtual network (*enforced* by the ANPIs.

A VL exists between two VRs and consists of three segments: one is connecting the physical hosts, and the other two are connecting each VR with its host. All three segments are aggregated to a VL by two software bridges (driven by two vCPIs). Therefore, for each VL (between VR1 and VR2 for example) the following operations are conducted:

a) Create the central segment (tunnel) between the physical components (optional if the linked routers are hosted on the same physical component).

b) Add a virtual network interface to both VR1 and VR2.

c) Bridge each virtual interface to the corresponding physical component to create the first link-segments.

d) Deploy the networking facing-services in VR1 and VR2.

To start up a VR, the vCPI uses hypervisor commands to create a virtual router from a template image. This template is instantiated with individual configuration options, like the initial network address or the amount of virtual hardware to be assigned. The tunnels to instantiate VLs are created using OpenSSH. Tunnels that are using the same physical Network Interface Card (NIC) share the available bandwidth among themselves. Statically assigning a guaranteed amount of bandwidth for a VL is possible by using traffic control mechanisms as demonstrated in [16].

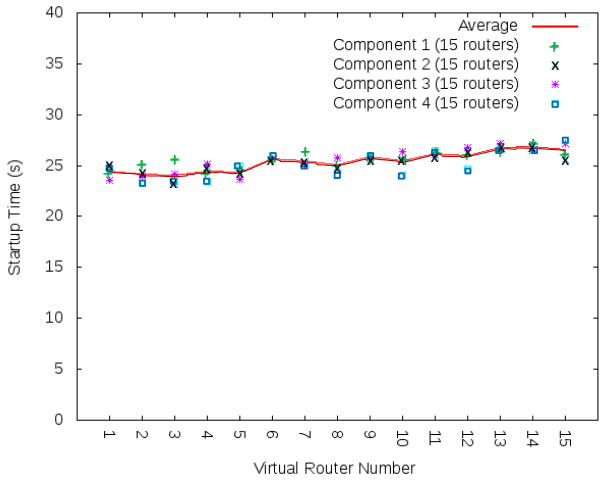

Fig. 4. Start-up Times of 60 Virtual Routers Created in Four-Component physical Test-bed

The AMSs and DOCs in this test scenario issue commands to the vCPIs sequentially, which in turn have processed them (in parallel when possible). The start-up times for the Virtual Routers (VRs) are depicted in Figure 4, where the most relevant behaviour is that the start-up times are stable for an increasing number of VRs in each physical component. In this execution run each VR needs in average 25 seconds to be started. After the routers are started, the AutoI solution reacts to this context-change (new virtual resources available) and issue appropriate commands to the vCPIs to instantiate the Virtual Links (VLs). The vCPIs enforce the commands in the four components of our test-bed in parallel when possible, using the three-segment approach described earlier. Again, the most relevant behaviour is that the start-up times of each VL are stable for an increasing number of VLs in the four physical components. The average time to construct a VL is about 5 seconds with a little variation between the minimum and maximum values as graphically depicted in Figure 5.

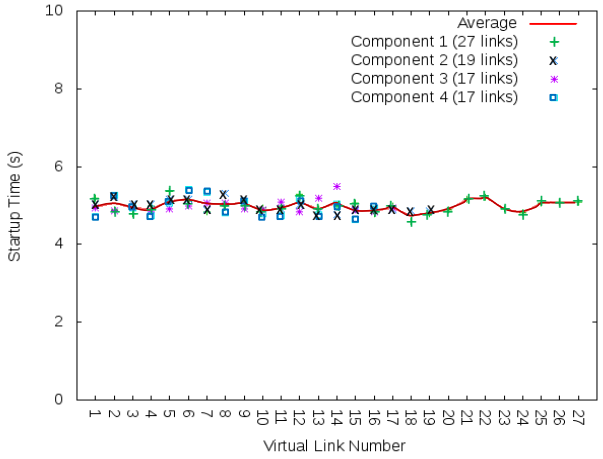

Fig. 5. Virtual Link Creation Times in 4-Component-physical Test-bed

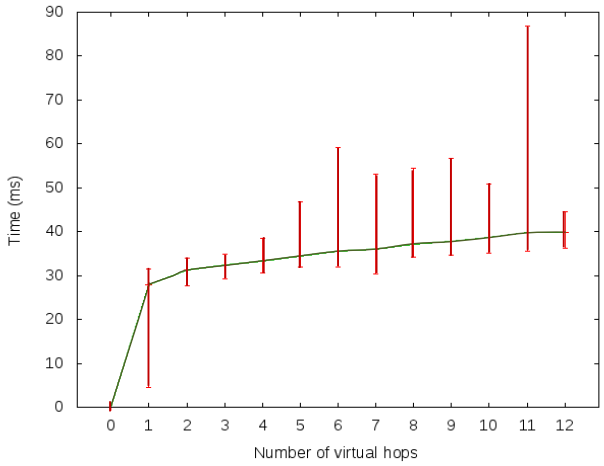

Fig. 6. Round-Trip Times in 60 Router- 80 Link Virtual Network

As mentioned earlier, the autonomic loop's enforce parts assessed by the ANPIs are devoted to discover and deploy the networking-facing services (i.e. routing) that will make such network operational. The round-trip times between each Virtual Router were measured to test that the network is operational. The minimum, maximum and mean values of 5 measurements are presented in Figure 6. The most important behaviour here is that the average round-trip time appears to be linear with the number of virtual hops, with a moderate slope and values below 40 ms in our test-bed.

B. Service Deployment Results

This section demonstrates the AutoI context-aware, on-demand, service deployment capabilities over virtual networks. Similarly as in our last test scene, the autonomic loop's *collect* part is exercised by the Context Information Services Platform (CISP) that manages context information and triggers notifications about changes corresponding to service requests for virtual networks, availability of services, resource usage information, etc.

The autonomic loop's *analyse* and *decide* parts are implemented by the AMSs and the DOCs, which analyse context changes (e.g. service requests) and evaluate the conditions under which context changes occur, and eventually decide on appropriate actions that would fulfil their business goals. Business goals in this test case would have an impact on the level of service deployment that would be eventually enforced. For example, Local Service support implies that a service would be deployed in a single virtual element, whereas Domain or Global support would imply that a set of services be deployed by network operators to a specific area of the virtual network.

The autonomic loop's *enforce* part is assessed by the Autonomic Network Programming Interface (ANPI), which executes a number of atomized autonomic processes in each node of the network when necessary. The main AutoI components enforcing this service deployment test are graphically depicted in Figure 7.

The ANPI maintains service repositories with available tested service code. Services are discovered by the ANPIs and

they are available for download from repositories located in the virtual network.

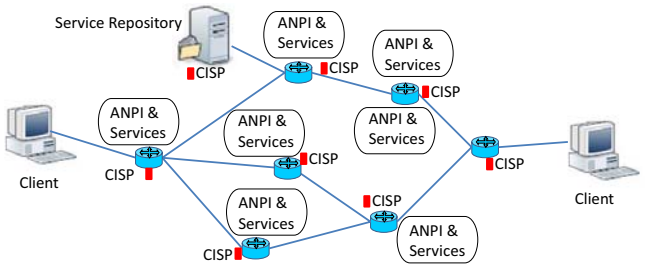

Fig. 7. Basic Virtual Network with AutoI Service Deployment Support

ANPI daemons deploy and manage the lifecycle of services. There is one ANPI running in each virtual component of the deployed network. When the ANPI receives a command from an AMS and/or DOC component to deploy or to migrate a specific consumer-facing and/or the resource-facing service, it analyses such information, and this triggers a new decision making process in the ANPI nodes. The type of service, characteristics, availability of services in the network, etc., is information that is taken into account to decide the best deployment steps.

The ANPI communicates through the Context and Information Service Platform (CISP) to expose and notify service deployment operations and services states.

Following on the described test case, the action of a client requesting a streaming service in the network of Figure 7 is emulated. AutoI creates a new virtual router on demand and attaches it to the new client. The ANPI discovers the location of the appropriate streaming services to drive the configuration of the required networking services (e.g. routing services in this case) and supporting services (e.g. context, monitoring) required. In each router of the network (now with 8 virtual routers), the ANPI deploys a basic routing service. The Figure 8 shows the time taken (bottom part) for the deployment of 8 ANPIs (continuous line), and the time taken (dotted line) to deploy 12 services (networking-facing, and application-facing) required to provision an end-to-end streaming service over the deployed virtual infrastructure. It is worth mentioning that once all ANPIs are deployed there is a small gap of time taken by the AutoI systems to find and download the required services and correlate appropriate context-changes. After this, the services are deployed, configured and started in about 3 minutes as depicted at the right part of Fig. 8.

Physical resources have limited capacities in terms of bandwidth, CPU, memory, etc., which in turn are shared among virtual resources. Lack of physical resources is eventually manifested as service degradation. AutoI reacts effectively to service degradation with coordinated service migration actions, in which, all virtual networking services and application services are actually re-configured in appropriate physical resources. The interested reader will find extensive results of the Autoi migration support in [10]. The remaining of the paper will focus on the scalability support.

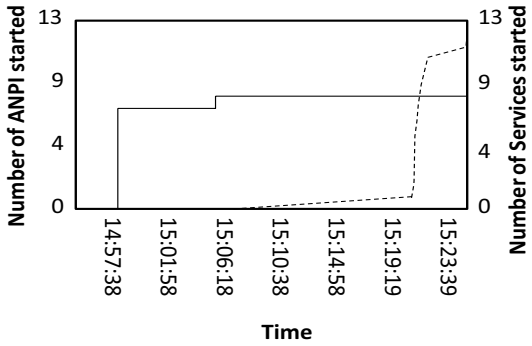

Fig. 8. Deployment of ANPIs (continuous) and Deployment of Services (dotted)

C. Scalability Results on Network Virtualisation and Service Deployment

Large scale validations were performed in an experimental test-bed (Grid5000 test-bed [17]) composed by a cluster of 10 separate sites located in France, where all AutoI components were installed. The test-bed supports 5000 cores located on various clusters connected with 10G links.

This section analyses the scalability of the AutoI solution for network virtualization and service deployments at large scale, in similar set ups as presented for mid-scale validations in sections III.A and III.B.

TABLE I
VIRTUAL INFRASTRUCTURE DEPLOYMENT FROM 10 TO 150 VR

Deployment of VR	10PM * 1VR = 10 VR	10PM * 3VR = 30 VR	50PM * 1VR = 50 VR	50PM * 3VR = 150 VR
Lyon	55s	3m18s	1m38s	3m42s
Bordeaux	57s	3m7s	1m14s	4m41s

Table I shows the result of the deployment of 10 to 150 Virtual Routers (VR) on 10 to 50 Physical Machines (PM) on two sites of the test-bed. It is worth mentioning that the deployment of virtual machines depends on the number of Virtual Routers to deploy more than depending on the number of Physical Machines. Observed results are closely similar between Grid5000 sites. Table I shows the results obtained from two sites (Bordeaux and Lyon).

For service deployment tests, target topologies like chains or trees were generated with the means to allow analysis of observed results. Figure 9 presents the service deployment results of a virtual network with 110 virtual routers in a chain topology in the Grid5000 test-bed. The deployment of 110 ANPI daemons with its activation occurred in 30 seconds (continuous line in Fig. 9), while the deployment of 110 small sized services (4.4 Kbytes) required only 8 seconds (dotted line in Fig. 9).

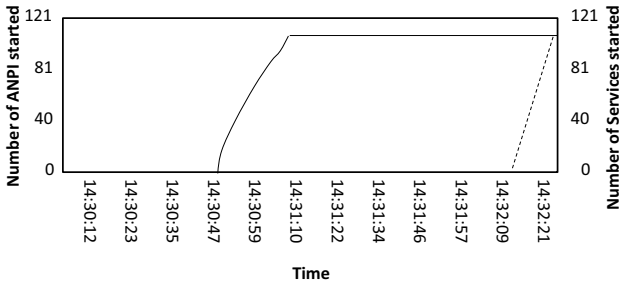

Fig. 9. Deployment of 110 ANPI (continuous) and 110 services (dotted)

Figure 10 presents the results of 220 services deployment on 110 virtual routers located in one virtual network with tree topology. This scenario shows that service deployment can occur at any time during the life of components provided that the ANPIs are already deployed in each virtual router. The deployment of the first 110 services occurred in 8 seconds (first part of the dotted line in Fig. 10). After a small gap the remaining 110 services were progressively deployed in about 9 minutes.

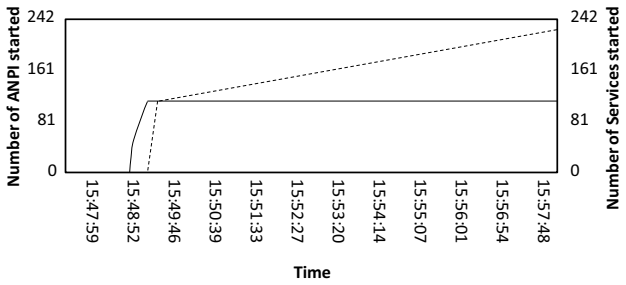

Fig. 10. Deployment of 110 ANPI with 220 services

IV. Technical Discussion and Lessons Learnt

One of the primary arguments for setting up multiple virtual infrastructures is the possibility to support networks with different network protocols on the same hardware. However, it is necessary to develop and provide the means to also manage the virtual network elements, in particular the configuration of the virtual network layer. In order for network virtualisation techniques to be a key component for next generation Internet, network virtualisation interfaces (like the vCPI and vSPI in our architectural model) need to be able to configure virtual network interfaces, while at the same time need to remain protocol-agnostic and open to future Internet protocol stacks.

Stability of virtual networks is an important issue that deserves special attention. Several instability problems were encountered during our experimental research. For example, when trying to create a number of coexisting virtual networks, each with 6 virtual routers distributed over four physical components, instability problems occurred when going beyond 15 virtual networks. Beyond this point, the creation of both virtual links and virtual routers became unpredictable. From our numerous experiments the conclusion drawn is that current virtualisation technologies are not built to cope with the dynamicity and load expected in future virtualised network environments. Additional implementation effort is needed to bring both hypervisor and virtual link technologies to a level where arbitrary creation of virtual networks becomes possible.

Programmability in network and services encompasses the study of decentralized enablers for dynamic (de)activation and reconfiguration of new/existing services, including management services and network components. AutoI has taken the challenge to enable trusted parties (users, operators, and service providers) to activate management-specific service and network components into a specific platform. Dynamic programming enablers will be created as executable service code, which can be injected/activated into the system's elements to create the new functionality at runtime. Network and service enablers for programmability can therefore realise the capabilities for flexible management support.

Large scale validation experiments were performed to validate the efficient support of AutoI service deployment support. Due to the high quality (in terms of latency and throughput) of the test-bed, time required to deploy services was extremely short. However, deploying services in less reliable infrastructures requires more fault tolerant approaches.

V. Related Work

The last decade has seen a tremendous interest for all aspects of the future Internet (FI). As a comprehensive survey would require more than a single paper, this section describes key initiatives in USA and Europe contributing to the development of the Future Internet

From the USA perspective, the National Science Foundation (NSF) supports four big projects. Named Data Networking [18] is aiming at an approach to identify the content to be supported by the Future Networks by itself instead of the locations where it resides. Mobility First [19] is looking at the inherent challenges of mobility and in particular to the use of opportunistic networking to support communications between end points. NEBULA [20] is aiming at a cloud computing architecture as a means to guarantee always available services. Finally, eXpresive Internet Architecture [21] addresses the growing diversity of network use models, the need for trustworthy communication, and the growing set of stakeholders who coordinate their activities. The scope of all these projects is much broader and concentrated on issues different than the ones presented in this paper. None of them use an orchestration plane to coordinate autonomic systems and do not explicitly mention the autonomic networking paradigm in support of service deployment and maintenance.

From the European perspective, efforts are driven by the EC and its Future Internet Assembly (FIA). The FIA is supported by researchers working on approximately thirty projects of dealing with aspects of the Network of the Future. Closely related to our objectives we can mention TRILOGY, SOCRATES, 4WARD and Universal projects..

The focus of TRILOGY [22] is on the development of the generic control functions of the network. These control functions deal with routing mechanisms, resource control and social and commercial control. Instead our approach concentrates on a management plane in support of service deployment.

The scope of the SOCRATES project [23] is the bottleneck problems created by the mobile access network and proposes self-* mechanisms to create a solution. In that sense our approach adopts the same conceptual solution because the autonomic Internet as we have conceived it has to make extensive use of self-* mechanisms. Nevertheless the AutoI autonomic approach goes beyond the access network and makes it an integral part of the fixed network as well. Furthermore, the mobile technologies considered in AutoI are not related to any particular technology (e.g. LTE) as in SOCRATES.

4WARD [24] makes use of paravirtualization systems like AutoI approach to virtualize routers and network links. The most important similarity is the management approach; both approaches adopt the autonomic paradigm. Nevertheless the main difference is in the architectural approach. In fact, 4WARD makes emphasis on the "in-network management" solution, i.e. embedding the management functionality in the same managed network device, which also makes use of a hierarchy from the operator management guidelines to the enforceable actions in the managed network devices. Instead, AutoI creates a multi-level hierarchy involving the service plane, the distributed orchestration elements and the autonomic management systems, which take care of one or many devices as necessary. From the architectural point of view the AutoI approach is more evolutionary from the current distributed management systems. The management functions are separated from the data forwarding and control functions and are not necessarily associated to the network device, but can be shared by different network devices. Separating the management activities in planes and not associating management functions to managed devices makes our solution scalable and easy to deploy in current network infrastructures.

UniverSelf [25] is meant to create a framework federating different self-management approaches to make the Future Internet a global autonomic management system.

VI. Concluding Remarks

This work has presented the design and validation results of an open-software defined network infrastructure (i.e. a Network Cloud) that enables a fast and scalable composition of services in an efficient manner, and the execution of these services in an adaptive way taking into account better shared network resources provided by a virtualized network substrate.

Current communication networks are composed of a set of heterogeneous resources. Virtualising these resources has served two purposes: Managing the heterogeneity through introduction of homogeneous virtual resources and enabling programmability of central network elements. The flexibility gained through this approach helps to adapt the network dynamically to both unforeseen and predictable changes in the network.

The Autonomic Internet service deployment approach has demonstrated that dynamic programming can be used to enable creating new functionality at runtime over virtual infrastructures. Executable service code can be injected and activated into the virtual systems elements in runtime to give higher degree of flexibility in the deployment of services in the future networks.

In addition, the adopted approach has revealed its scalability when large scale testbeds like Grid 5000 are used. Nevertheless, scalability in such a complex and multitier system like AutoI requires much more extensive testing than the experiments reflected in this paper. We mean for instance experiments within scenarios stressing specific planes or components of planes. Considering the AutoI architecture this would likely yield to tens of scenarios. This paper reflects the results of scalability tests in one particular scenario. Then we have to emphasize that our system scales well under the conditions of this particular scenario and that by no means these results can be generalized to different scopes or situations. Additional testing is in fact part of a challenging future work.

Acknowledgment

This work was undertaken in the context of the FP7-EU AUTOI project and the MCYT TEC2009-14598-C02-02.

References

[1] "Future Network Vision - Objectives and Design Goals" Y.3001 Recommendation ITU-T - http://www.itu.int/en/ITU-T/focusgroups/fn/Pages/Default.aspx

[2] B., Rochwerger, et al. "An Architecture for Federated Cloud Computing" in Cloud Computing: Principles and Paradigms (eds R. Buyya, et al), John Wiley & Sons, Inc., Hoboken, NJ, USA. 2011

[3] Y. Wang, et al. "Virtual routers on the move: live router migration as a network-management primitive" ACM conference on Data communication SIGCOMM 2008

[4] N. M. K. Chowdhury, R. Boutaba "A survey of network virtualization" Computer Networks. vol. 54, pp. 862–876, April 2010.

[5] T. Anderson et al. "Overcoming the internet impasse through virtualization," Computer, vol. 38, pp. 34–41, April 2005.

[6] L. Andersson, T. Madsen, "Provider provisioned virtual private network (VPN) terminology" Internet Engineering Task Force, RFC 4026, March 2005

[7] A. Galis et al. "Programmable Networks for IP Service Deployment" Artech House Books, 2004

[8] N. M. K. Chowdhury and R. Boutaba "Network virtualization: State of the art and research challenges" IEEE Communications Magazine, vol. 47, no. 7, 2009

[9] M. Yu et al. "Rethinking virtual network embedding: Substrate support for path splitting and migration" ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 17– 29, 2008

[10] EU IST Autonomic Internet Project Web Site http://ist-autoi.eu.

[11] D. F. Macedo et al. "The Autonomic Internet Approach for the Orchestration of Next-Generation Autonomic Networks" Journal Annals of Telecommunications, Ed. Springer, April 2011

[12] J. Rubio-Loyola et al. "Platforms and Software Systems for an Autonomic Internet" IEEE GLOBECOM 2010, 6-10 Miami FL, USA

[13] D.D., Clark, C., Partridge, J.C. Ramming, J.T. Wroclawski "A knowledge plane for the interne" IEEE SIGCOMM 2003
[14] AG Ganek, TA Corbi "The dawning of the autonomic computing era" IBM Systems Journal, 2003, 42(1):5–18
[15] J. Pattloch et al. "X-WiN: The New German National Research and Education Network" Praxis der Informationsverarbeitung und Kommunikation. Volume 29, Issue 1, Pages 50–53, ISSN 0930-5157
[16] A. Berl et al. "Using System Virtualization to Create Virtualized Networks" Journal Electronic Communications of the EASST, 17:1--12 2009, ISSN: 1863-2122
[17] Grid5000 TestBed www.grid5000.fr/
[18] Named Data Networking, http://www.named-data.net/index.html
[19] Mobility First, http://mobilityfirst.winlab.rutgers.edu/
[20] Nebula, http://nebula.cis.upenn.edu/
[21] eXpressive Internet Architecture, http://www.cs.cmu.edu/~xia/
[22] 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/
[23] EU IST SOCRATES project "Self-Optimisation and self-ConfiguRATion in wirelEss networkS" http://www.fp7-socrates.org
[24] EU IST FP7 4WARD Project, http://www.4ward-project.eu/
[25] EU IST FP7 UNIVERSELF Project "Realizing Autonomics for Future Networks" http://www.univerself-project.eu