This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Platforms and Software Systems for an Autonomic Internet

J. Rubio-Loyola[1], A. Astorga[2], J. Serrat[2], W. K. Chai[3], L. Mamatas[3], A. Galis[3], S. Clayman[3], A. Cheniour[4], L. Lefevre[4], O. Mornard[4], A. Fischer[5], A. Paler[5], H. de Meer[5]

[1]CINVESTAV Tamaulipas – Mexico, [2]Universitat Politècnica de Catalunya – Spain, [3]University College London – U.K., [4]INRIA, University of Lyon – France, [5]University of Passau – Germany

The current Internet does not enable easy introduction and deployment of new network technologies and services. This paper aims to progress the Future Internet (FI) by introduction of a service composition and execution environment that re-use existing components of access and core networks. This paper presents essential service-centric platforms and software systems that have been developed with the aim to create a flexible environment for an Autonomic Internet.

***Keywords*—"Future Internet", "Autonomic", "Network Management", "Virtualisation"**

I. INTRODUCTION

THE current Internet has been founded on a basic architectural premise: a simple network service is used as a universal means to interconnect intelligent end systems. The end-to-end argument has served to maintain this simplicity by pushing complexity, into the endpoints, allowing the Internet to reach an impressive scale in terms of inter-connected devices. However, the very success of the Internet is now creating obstacles to future innovation in the networking technology that lies at its cores and the services that use them. The ossification of the Internet makes the introduction and deployment of new network technologies and services very difficult and very costly. Now we are faced with the relative inflexibility of the Internet and with the lack of built-in:

- Service support, provisioning, discovery and management.
- Network and device mobility.
- Network management functionality.
- Quality of Service (QoS) and security facilities.
- Programmatic addition of new functionality - capability for activating on-demand new services, network functionalities or protocols.
- Orchestration of security, reliability, robustness, mobility, context, service support and management of the communication and services.

Several research initiatives and projects around the world are currently addressing key challenges of today's Internet and Future Internet. In Asia for example, the AKARI project [2] aims to implement a new generation network by 2015, through the development and design of a new network from a clean slate approach. The migration to the Future Internet from the current one is also considered, once the new architecture design and principles are stable. In America, FIND (Future Internet Design) [3] is a major initiative that includes research efforts working on the development of network architecture, security, advanced wireless and optical properties, economical principles, and in general, mechanisms to build a global network 15 years from now, and its realisation without the concerns of the current Internet. The GENI (Global Environment for Network Innovations) [4] Program supports fundamental challenges of the current Internet like inadequate security, reliability, manageability and evolvability.

In Europe, a number of IST projects are currently addressing Future Internet research problems that include, the design and validation of Autonomic Network Architectures [5], Biologically-inspired Autonomic Networks [6], and others like the EU TRILOGY Project (Architecting the Future Internet) [7], EU 4WARD [8], EU PSIRP Project (Publish-Subscribe Internet Routing Paradigm) [9], EU FIRE Projects (Future Internet Research & Experimentation) [10] and EU-OneLab2 (Future Internet Test Beds) [11], dealing with both FI architectural and management issues.

This paper presents relevant implementation results of the EU Autonomic Internet AUTOI project [1], which suggests a transition from a service agnostic Internet to service- and self-aware Internet managing resources by applying Autonomic principles. In order to achieve the objective of service- and self-aware networking resources and to overcome the ossification of the current Internet, AUTOI aims to develop a self-managing virtual resources overlay that can span across heterogeneous networks and that supports service mobility, security, quality of service and reliability.

After this Introduction, Section II briefly describes the AUTOI architectural framework. Section III describes the relevant platforms and software systems that enable an Autonomic Internet. Section IV provides a Future Internet use case pivotal for the experimental executions provided in the same Section IV. Finally, Section V concludes this paper.

II. AUTONOMIC INTERNET PLANES

AUTOI [1][16] advocates for an autonomic management architectural model consisting of a number of distributed management systems within the network, which are described with the help of five abstractions and distributed systems - the

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

OSKMV (Orchestration, Service Enablers, Knowledge, Management and Virtualisation) planes depicted in Figure 1.

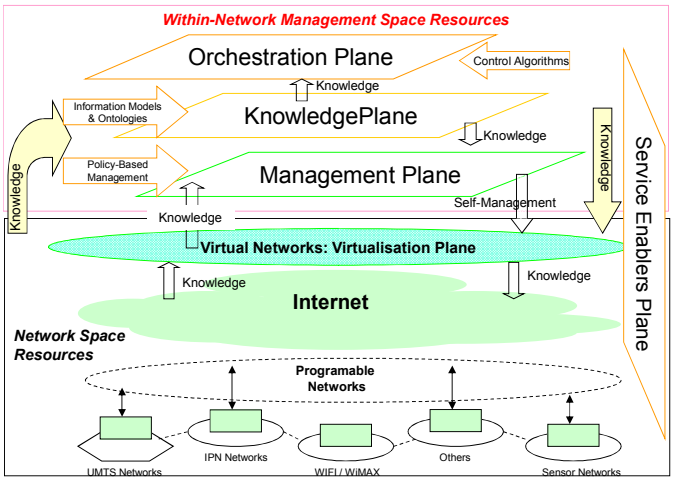

Figure 1. The Autonomic Internet (AUTOI) Planes

The ***Orchestration Plane*** **(OP)** [1] is a conceptual definition for the instruments that govern and integrate the behaviours of the management systems distributed across the network, in response to changing context and in accordance with applicable business goals and policies insuring integrity of the Future Internet management operations. The OP can be thought of as 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 and interworking with other management functions. It hosts one or more Autonomic Management Systems (AMSs) and it is made up of one or more Distributed Orchestration Components (DOCs), and a dynamic knowledge base consisting of a set of models and ontologies and appropriate mapping logic. Each AMS represents an administrative and/or organisational boundary that is responsible for managing a set of devices, (sub) networks.

The ***Service Enablers Plane*** (SP) [1] consists of functions for the automatic (re)deployment of new management services, protocols as well as resource-facing (i.e. QoS functions) and end-user facing services. It includes the enablers to allow code to be executed on the network entities. The safe and controlled deployment of new code enables new services to be activated on demand. This approach has the following advantages: i. Automatic service (re)deployment allowing a significant number of new services to be offered on demand; ii. Special management functions and services can be easily enabled locally for testing purposes before they are automatically deployed network-wide; iii. Eases the deployment of network-wide protocol stacks and management services; iv. Enables secure and controlled execution environments; v. An automatic decision making infrastructure guides the deployment of new tested network services; vi. Optimised resource utilization of the new services and the system.

The ***Knowledge Plane*** (KP) [1] consists of models and ontologies and the means to provide increased analysis and inference capabilities. It provides knowledge and expertise to enable the network to be self-monitoring, self-analyzing, self-diagnosing, and self- maintaining or –improving [12]. It brings together widely distributed data collection, wide availability of that data, and sophisticated and adaptive processing or KP functions, within a unifying structure that brings order, meets the policy, scaling and functional requirements of a global network. The main KP components are an Information and Context Service (ICS) plus models and ontologies, which enable the analysis and inferencing capabilities. The ICS provides: i. information-life cycle management (storage, aggregation, transformations, updates, distribution) all information and context in the network and addresses the size and scope of the Internet; ii. responsiveness to requests made by the AMSs; iii. triggers for the purpose of contextualisation of AMSs (supported by the context model of the information model defined by the KP); iv. support for robustness enabling the KP to continue to function as best possible, even under incorrect or incomplete behavior of the network itself; v. 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.

The ***Management Plane*** (MP) [1] consists of Autonomic Management Systems (AMSs), which are designed to follow autonomic control loops [13][14]. In our framework AMSs are designed to meet the following design objectives and functionality: i. Embedded (Inside) Network functions as the majority of management functionality should be embedded in the network. As such the AMSs run on execution environments on top of virtual networks and systems, which run on top of all current network and service physical infrastructures; ii. 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 behavior serve its service purposes; iii. Adaptive and Self-adaptive functions: It triggers changes in network operations (state, configurations, functions) function as a result of the changes in network and service context; iv. Automatic self-functions: It enables self-control of its internal network operations, functions and state. It also bootstraps itself and it operates without manual external intervention. Only manual/external input is provided in the setting-up of the business goals; v. Extensibility functions: It adds new functions without disturbing the rest of the system ((Un)Plug_and_Play/ Dynamic programmability of management functions & services); vi. Simple cost functions: Minimize life-cycle network operations' costs and minimize energy footprint.

The ***Virtualisation Plane*** (VP) is one of the key requirements that differentiate AUTOI from other efforts is its emphasis on virtualisation [15]. The Virtualisation Plane 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 VP is used by the Orchestration plane to govern virtual resources, and to construct virtual services and networks that meet stated business goals having specified service requirements. The AMSs manage through the VP, the physical resources, and the construction of virtual resources from physical resources. Virtualised resources are manageable via virtualisation interfaces. This separation enables a system-wide management of virtual resources by other planes, while the management of physical resources is done by the virtualisation plane.

III. Platforms and Supporting Systems

This Section presents a set of service-centric platforms and supporting systems that have been developed in Autonomic Internet (AUTOI) project [1][16] that combine to create a highly open and flexible environment for Future Internet to manage and wrap over the heterogeneity of multiple types of access and core networks.

A. Virtual Component Programming Interface System

Virtualising heterogeneous network resources serves two purposes: managing the heterogeneity via introduction of homogeneous virtual resources and enabling programmability of network elements. The flexibility gained helps to adapt the network dynamically to unforeseen and predictable changes. A vital component of the Virtualisation Plane (VP) is a common management and monitoring interface. For this purpose we developed an open source of this interface – the virtual Component Programming Interface (vCPI) [1]. The power gained from its monitoring and management functions gives the vCPI the ability to deploy context-aware autonomic networks. The vCPI is a modular and scalable system for monitoring and managing virtual resources. It operates locally; for each node of a physical network there is an embedded vCPI. The physical nodes are hosting virtual resources. Interacting with the virtual resources and their environment starts from the assumption that the vCPI is aware of the structure of the virtual resources. The vCPI functions as an information processor such that multiple information threads are simultaneously run (multi-threading).

The vCPI follows the client-server architecture. Clients send commands for reading or setting resource parameters. The Communication Interface acts as a gateway and does not enforce specific communication protocol. Received requests are dispatched to the Command Dispatcher where these are forwarded to request handlers.

The vCPI needs to be aware of the structural information of the relations among virtual resources. A discovery mechanism was included to inspect the contents of the physical component and map it to a data structure.

The vCPI stores the information as a tree. An example is the setup of a virtual link between two network interfaces. To monitor the activity of this virtual link, the respective network interfaces have to be monitored. This suggests modeling the interfaces as children of the virtual links. Our current implementation considers virtual routers constructed from virtual machines running under XEN [17] where each virtual machine has virtual interfaces connected to virtual bridges. All virtual machines use the same image file.

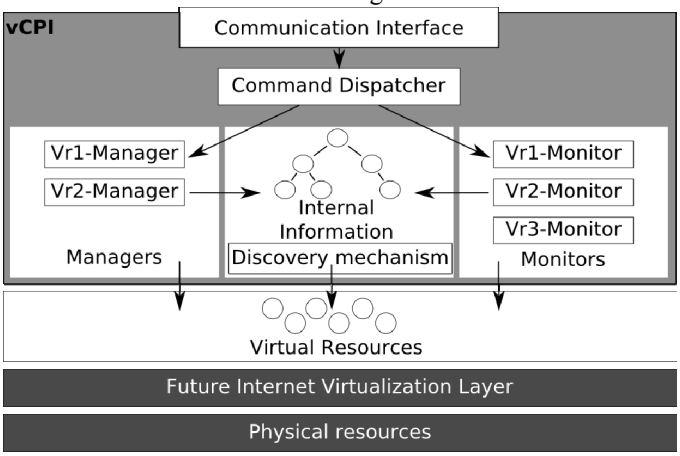

Figure 2. The conceptual view of vCPI within a physical component

B. Context Information Service Platform

The Future Internet (FI) will have many of its core features and functions based on virtualized resources [18]. Handling context information in virtual environments is an essential requirement in realizing FI. We present the Context Information Service Platform (CISP), which is part of the AUTOI's Knowledge Plane [16] as a new infrastructure for collection, processing and dissemination of context information in virtual networks, as a support for the deployment, evolution and autonomy of communication services and applications. Detailed CISP design and usage is presented in [1][19].

Context information covers at least the following: 1. Network context, including: i) Network description (e.g., network identity, location, access-types, coverage); ii) Network resources in general (e.g., bandwidth, supported network services, availability in terms of QoS); iii) Network configuration (e.g., network protocol parameters); iv) Network behavior (e.g., network monitoring information, level of congestion etc); 2. Services context, including: i) Service directory (e.g., available media ports for media conversion etc); ii) Security levels provisioned; iii) Service qualities (e.g., QoS); iv) Service execution environment characteristics; v) Service configuration characteristics; vi) Service life cycle characteristics; 3. Policies, including: i) Policies/rules representation; ii) Enforced Policies/rules, iii) History of policy conflicts / conflict predictions etc.

Context Information Service Platform (CISP) manages context information, including its distribution to clients/consumers: context-aware services, either user-facing applications/services or network management services, which make use of or adapt themselves to context information. It makes interactions between context sources and context clients simpler and efficient. It is a mediating unit and reduces the number of interactions and overhead control traffic.

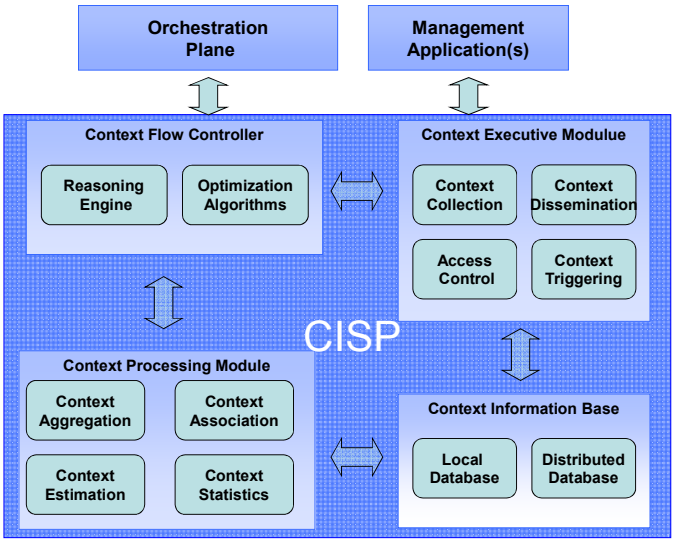

Figure 3. AUTOI context information service platform

CISP is realised by four basic functional entities: (i) the Context Executive (CE) Module, (ii) the Context Processing (CP) Module, (iii) the Context Information Base (CIB), and (iv) the Context Flow Controller (CFC) (see Figure 3). The CE deals with indexing, registering, authorizing and resolving context names into context or location addresses. It meets the requirements of context collection, context dissemination, interfaces with the CIB and supports for access control. The Context Processing (CP) is responsible for the context optimization, including context processing, estimation and creation of appropriate context associations between clients and sources. The context association allows the CISP to decide where a specific context should be stored and how. Furthermore, the CP collects statistics about context usage. The Context Information Base (CIB) provides flexible storage capabilities, in support of the CE and CP modules. The Flow Controller (CFC) configures the CP and CE Modules based on the requirements of the Management Application and the general guidelines from the Orchestration Plane (OP).

C. *Autonomic Network Programming Interface Platform*

Fast deployment of new services is a much sought-after feature in Future Internet. We propose a programmable infrastructure that partially supports the functions of the Service Enablers Plane, namely the Autonomic Network Programming Interface (ANPI).

ANPI is composed of a Service Deployment Daemon (SDD) that receives the requests from the upper layers (management and orchestration planes). It maintains a local database containing all the services it is responsible for. It signals/migrates the services it has instantiated. This insures the lifecycle management of the services. ANPI manages service deployment based on decision-making mechanism that handles the service deployment autonomously. Furthermore, ANPI maintains a database with available tested service code. The latter can be associated with information required for the autonomic decision-making. For example, the resource usage information can be seen as context information that can be collected from the autonomic managed entities. Note that the behavior and resource usage of the deployed services are monitored locally.

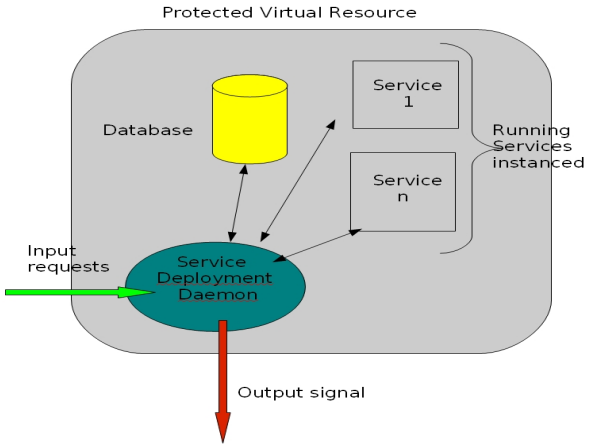

Figure 4. Conceptual Illustration of the ANPI

A set of functions that are exposed to the management and orchestration planes to enable handling of services is implemented. This programming interface is used to deal with the deployment and administration of service and management components. In our implementation, the service's deployment is composed of four main steps: (1) discovery, (2) download, (3) deployment and (4) signaling (e.g. start, stop, resume etc.). During the lifecycle of the service, ANPI updates regularly the Knowledge Plane (KP) with all information concerning the service's status using the information model.

D. *Other Supporting Systems*

Other service-centric supporting systems that similarly to all systems presented earlier have been developed in the AUTOI project under an open source software license [1].

A Model-Based Translator (MBT) takes configuration files compliant with the AUTOI Information Model XML Schema and translates them to device specific commands. This eliminates the need to update software when managing heterogeneous networked entities whose data models may change regularly. This supports the Knowledge Plane.

An Autonomic Policy-based System supports context-aware policy-driven decisions for management and orchestration. It is able to subscribe to policy-driven context changes on the Context Information Service Platform (CISP) and to manage these changes in favor of sensible policy-driven decisions for coordinated service deployment and maintenance. This is a supporting component for the Management and Orchestration Planes.

Future Internet will have many of its functions based on programmability of network resources [18]. Handling programmability [20][21] in virtual environments is an essential requirement in realizing Future Internet. For such purposes AUTOI introduces the XINA platform, which is a modular scalable platform that enables the deployment, control and management of programmable or active sessions over virtual entities, such as servers. This is platform that assesses the Service Enablers Plane functions.

IV. Use Case and Execution Results

Consider an application service that provides large amounts of diverse-nature information, such as multimedia files which are stored on geographically distributed servers. Application services are provisioned to home and corporation users with certain contracted quality levels, each with concrete requirements. For timely-effective service provision, the user downloads content from the closest server, which in turn would get the information from the server that stores the information. Servers provide a kind of P2P overlay network [22] with channels of communication with large capacity to download information directly from the server that stores it. The users are not associated to any permanent server and they can use different types and terminal manufacturers. Users can be fixed or mobile. The latter ones may pass through locations with various access systems and technologies, namely areas with 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).

Virtual infrastructures supporting the use case are created through the virtual Component Programming Interfaces (vCPIs). The scalability of the vCPI was empirically evaluated by running different tests with reproducible outputs. The testing scenarios were targeted at showing how the vCPI reacts when having to monitor/manage multiple virtual resources. The test bed consisted of two physical machines, both using the same configuration (2 Quad Core AMD Opteron 2347H CPUs and 32 GB RAM); one machine was used for running the vCPI and the other one for sending commands to the first one. We considered virtual routers as being constructed from virtual machines (Ubuntu server 8.04 with 128 MB RAM) running under XEN [17], and each virtual machine with one virtual interface connected to a virtual bridge. The process of creating a virtual router consists of the following steps: 1. vCPI receives the *registerVM* command (timestamp *Trecv*); 2. vCPI communicates with the virtualisation software (in our case XEN), initiates the creation of the virtual router and waits for the router to be created (timestamp *Tstart*); 3. the virtual router is created and is performing the boot-up sequence; 4. after it has successfully finished the boot-up sequence, the router signals back to the vCPI that it is now ready to be used (timestamp *Talive*).

A first scenario was the sequential creation, where a specified number of virtual routers are created one after the other. This means that after *Trecv* has been calculated for one virtual router the creation of the next one is initiated. The target of this scenario was to observe how the number of existing virtual routers is affecting the *Talive – Tstarted* duration. We tested the vCPI by creating 100 virtual routers. The results are depicted in Figure 5. A reason behind the sudden decrease of *Talive* could be that a single image file has been used for all routers. After the first 15 routers have started, the needed data for the initialization of the next ones is already fully cached in memory.

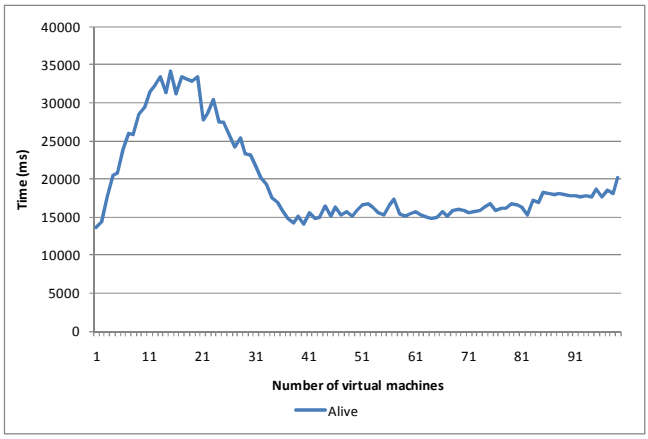

Figure 5. Sequential creation test of 100 virtual routers

The second scenario was the parallel creation of multiple virtual routers. The vCPI is multithreaded software, hence the need to observe how multiple registerVM commands are affecting the overall systems performance. The result of creating 25 virtual routers at the same time is shown in Figure 6. It takes between 40 and 60 seconds until the virtual routers are started. The machines are alive after more than 85 seconds. It can be observed, that the durations are on average less for the first 12 routers. A plausible motivation would be that the first 12 routers are faster at getting to the computational resources of the physical component, while the other routers have to wait.

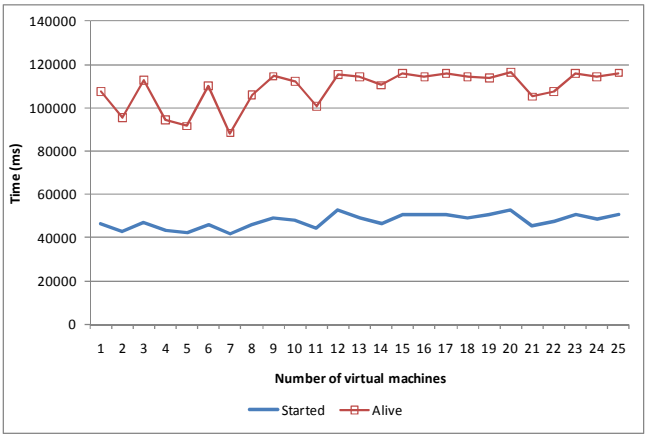

Figure 6. Simultaneous creation test of 25 virtual routers

The third scenario repeats the first scenario for a certain number of rounds separated by a specified number of seconds. Figure 7 shows a selection of these tests, namely shows the outputs of starting 100 virtual routers, starting in blocks of 10 separated by intervals of 60 seconds. With all these tests we conclude that, the duration until the virtual resources (in our case routers) are alive is linearly growing with their number. The number of repeated simultaneously started resources seems to affect the stability of the system, the unresponsiveness of the physical component increases, and influences the scalability of the vCPI. The sudden increase of the *Treg* in Figure 7 can be interpreted as symptoms that future virtual resources will crash. These symptoms can help the vCPI to better schedule the execution of the received commands. The context information in our use case is handled by the Context Information Service Platform (CISP).

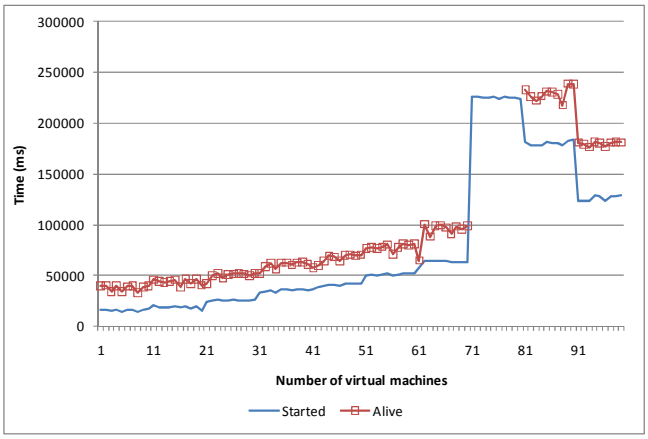

Figure 7. Repeated simultaneous creation of 10x10 virtual routers with 60 seconds intervals

We have evaluated the performance of the CISP in terms of communication and processing cost. The interested reader can consult [19] for a detailed analysis of its performance for regulating information flows based on the properties and the state of the network environment. In particular, experimental evaluation with different optimization techniques and algorithms demonstrate that CISP can adapt to different settings and optimization requirements.

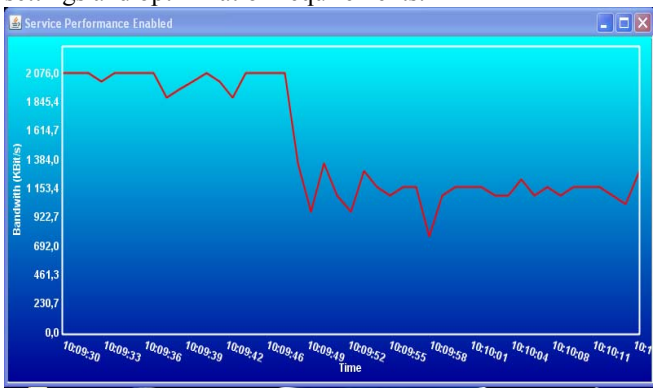

Figure 8. Experimental service Migration in the Grid 5000 test-bed

Figure 8 shows the measured bandwidth between a server and a client linked with a virtual network infrastructure (virtual routers on different physical machines) in the Grid 5000 test-bed (GRID5000 French Nationwide Experimental Platform embedding 5000 cores): We observe a service using a rate of 2 Mbits. At time 10:09:47, the ANPI intentionally generates a migration of customers to another physical machine where the network was limited to 1 Mbps. The phenomenon of migration is reflected by the drop in bandwidth induced by the capacity of the new network between server and client. This validation is currently extrapolated to large scale experiments to verify scalability of this approach.

V. Conclusion

This paper presents an essential service-centric framework and software systems that have been developed with the aim to create a flexible environment for an Autonomic Internet. The software systems are developed within the AutoI [1] project and are all available as open source. Though the detailed design of FI is still open for debate, it will undoubtedly be highly dynamic and based on virtualization technology. Architectural integration of such systems is presented in this paper through a number of scenario-based experiments. Our open source software systems represent a step in creating an environment conducive to the research and deployment of FI and fostering a total open Internet that is future-proofed.

Acknowledgment

This work is partially supported by the EU through the Autonomic Internet STREP project [1] and the NoE 216366 "EuroNF" http://euronf.enst.fr/. Some tests were performed on the Grid'5000 platform http://www.grid5000.fr, an initiative of the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS, RENATER and other partners. The first author is partially supported by CONACYT grant No. 121323. This work is partially supported by the Spanish Project TEC2009-14598-C02-02.

References

[1] EU-Autonomic Internet (AUTOI) Project http://ist-autoi.eu/autoi/
[2] AKARI "Architecture Design Project for New Generation Network" http://akari-project.nict.go.jp/eng/index2.htm
[3] FIND "Future Internet Design" http://www.nets-find.net/
[4] GENI "Global Environment for Network Innovations" http://www.geni.net.
[5] EU ANA Project "Autonomic Network Architectures" http://www.ana-project.org/
[6] EU BIONETS http://www.bionets.eu/ project "Biologically-inspired autonomic Networks and Services" http://www.bionets.eu/
[7] EU TRILOGY Project – Architecting the Future Internet http://www.trilogy-project.org/
[8] EU 4WARD http://www.4ward-project.eu/
[9] PSIRP Project - Publish-Subscribe Internet Routing Paradigm Project http://psirp.org/
[10] EU FIRE Project – Future Internet Research & Experimentation http://cordis.europa.eu/fp7/ict/fire/
[11] EU-OneLab2 – Future Internet Test Beds http://www.one-lab-2.org/
[12] Cheng, L. et al.- "Self-organising Management Overlays for Future Internet Services"- IEEE Manweek'08/MACE'08; 22-26 Sept. 2008, Greece
[13] Kephart, J.O., Chess, D.M."The Vision of Autonomic Computing", IEEE Computer", January 2003
[14] B. Jennings et al. Towards Autonomic Management of Communications Networks. IEEE Communications Magazine, 45(10):pp. 112–121, 2007.
[15] Berl, A., et al. -" Management of Virtual Networks"- IEEE Manweek/ EVGM; 22-26 Sept. 08, Samos
[16] A. Galis, et. al., "Management Architecture and Systems for Future Internet Networks" in "Towards the Future Internet – A European Research Perspective" ISBN 978-1-60750-007-0, Apr. 2009, IOS Press.
[17] P. Barham, et. al., "Xen and the art of virtualization", SIGOPS Oper. Syst. Rev. 37(5):164–177, 2003.
[18] A. Galis, et. al., "Management and Service-aware Networking Architectures (MANA) for Future Internet - System Functions, Capabilities and Requirements" Invited paper IEEE 2009 4th Int'l Conf. on Communications and Networking in China (ChinaCom) 26-28 Aug. 2009, Xi'an, China.
[19] L. Mamatas et al. "Towards an Information Management Overlay for Emerging Networks" 12th IEEE/IFIP Network Operations and Management Symposium NOMS 2010; 19-23 April 2010 Osaka, Japan.
[20] A. Galis et al(ed) "Programmable Networks for IP Service Deployment" ISBN 1-58053-745-6; pp450, June 2004; Artech House Books;
[21] D. Raz, et al. "Fast and Efficient Context-Aware Services" ISBN 0-470-01668-X; pp250, April 2006; John Wiley & Sons, Ltd.
[22] E.K.Lua et al., "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes", IEEE Communications Survey and Tutorial, March 2004