IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, p-ISSN: 2278-8719, www.iosrjen.org
Volume 2, Issue 10 (October 2012), PP 63-70

Management of Future Internet with Autonomic Network Architecture and Model

Prof.P.P.Rewagad[1], Dipali P.Sapkal[2]
[1,2](Computer Science and Engineering G.H.R.E.M, Jalgoan)

Abstract : *Research efforts a strong foundation towards standardization of architectural principals of the Managing Future Internet. Autonomic network engineering for managing Future Internet (AFI) has been established under the European Telecommunications Standards Institute (ETSI).AFI has been focusing on Scenarios, use cases, and requirements for autonomic self managing Future internet, as well as on architectural reference model for autonomic networking and self management.AFI has continued with a new work Item on requirements analysis and specification of implementation-oriented solutions for autonomics and self-management.AFI is establishing strategic liaisons with the standards developing organizations and research community. In this paper, we also describe the current internet techniques and future internet requirements and managing the future internet.*

Key words *— ANA, node, IDP, IC, FB, Future Internet.*

I. INTRODUCTION

The goal of the ANA project is to explore novel ways of organizing and using networks beyond legacy Internet technology and to demonstrate the feasibility of autonomic networking. To reach this goal, we have designed and implemented a framework which allows to dynamically composing protocol stacks at runtime, according to the current needs of the network. For example, security mechanisms are added on the fly if a device is in a hostile environment. In the following, we will give a brief overview of the relevant concepts and paradigms of the ANA framework [5, 1]. Basically, a new network abstraction used to describe the interactions in the ANA framework. In an ANA three basic elements: Functional Blocks - elements that implement protocol functionalities;

Information Dispatch Points (IDP) - used to address the functional blocks; and Information Channels (ICs) – used to illustrate the communication with a distant node 2. Fig. 1 depicts the node internal organization of a node in the ANA world

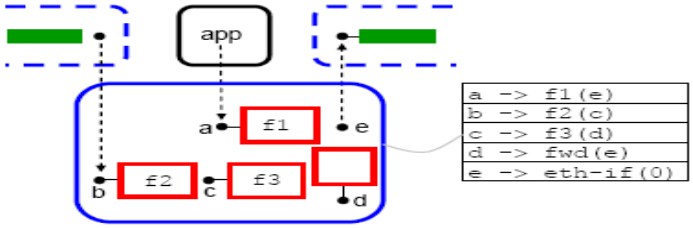

Figure 1. Internals of an ANA node

The boxes, labeled f1 - f3, represent *FBs* and are the dynamically compostable units of functionality. Data encapsulation, checksum computation or encryption is examples of such Functional Blocks. Each FB has one or multiple *IDPs* attached to it, labeled a - e. conceptually, an IDP is similar to a file descriptor in a file system. In order to dispatch the data within a node, the node maintains an *Information Dispatch Table* which keeps track of the IDPs allocated to specific instances of a FB. Modifying this dispatch table at runtime allows dynamically composing and recomposing the protocol stack within the node. This flexible and IP independent architecture of the ANA framework allows the integration of new communication paradigms and protocols for which legacy Internet technology is not well suited. As an example of such a communication paradigm, In a Pub/Sub messaging system, messages are not sent to a specific receiver address, but instead are categorized to classes, to which a receiver can subscribe [3][4]. In order to route messages in the Pub/Sub system, we use a field based approach as described in [6]. In field based routing, for each service that gets published, a potential field is

established in the network. This field is analogous to a physical potential field, for instance an electrostatic field. A service (e.g., a printer) is modeled as a positive point charge and a service request message (e.g., a print request) is modeled as a negative charge which is attracted by the service instances. Node n calculates its potential in the field according to where N is the number of service instances (e.g., the number of printers), Qj is the service capacity of a specific instance and |n − nj | is the distance (in number of hops) of node n to this service instance. Each node then forwards the requests it gets for a certain service according to the steepest potential ascent, that is, to the neighbor with the highest potential for the given service.

Designing the building blocks of the Future Internet is today one of the most important challenge for the networking Community. Visions are flourishing, encouraged by many large initiatives like GENI1 in the USA and the Future Internet initiative2 part of the European 7th framework program. Two main approaches have emerged as the leading paths to the future Internet: one advocating a clean slate, revolutionary approach to design the future Internet and a second one advocating an evolutionary approach extending current protocols with the required capabilities to address all issues of the future Internet

Self-managing Future Internet involves the interaction of five functional blocks:
• Self-configuration at device level and Self- Organization at a network level to adapt configuration to context.
• Self-protection to recognize and avoid threats (DDOS, heat, power failure).
• Self-healing to diagnose abnormal operation and take actions to normalize the behavior.
• Self-optimization to continuously improve the system performance.
• Self-monitoring to continuously collect state and context information.

Additional strata, often transversal to the above listed functions are commonly agreed-upon in the autonomic networking communities nowadays like network awareness [11], which extended with self-discovery and analysis features leads to autognostics.

A growing fraction of researchers and network operators are actually seriously concerned about the ability of the Internet to sustain more disruptive evolution steps. For many researchers including some of the early Internet designers the salient design principles (i.e. end-to-end principle and transparency, and global addressing) that contributed to the success of the Internet are also preventing any radical update of its core functionalities. As a result, the networking community has witnessed a renewed interest in the design of *clean-slate* network architectures that could better support disruptive evolution and emerging network paradigms such as content-based, ad hoc, sensor, mobile, and delay-tolerant networking. A key motivation is to foster innovation by freeing researchers from providing backwards compatibility with the current Internet. To support these objectives, the *Autonomic Network Architecture* (ANA) project [8] is exploring novel ways of organizing and using networks beyond legacy Internet technology. We are designing and developing a network architecture that can demonstrate the feasibility and properties of autonomic networking. The main guiding principle behind the architectural work in ANA is to strive for a maximum degree of flexibility in order to support *functional scaling* by design at all levels of the architecture. Functional scaling means that a network is able to extend both horizontally (*adding* more functionality) as well as vertically (different ways of *integrating* abundant functionality).

II. WORKING PRINCIPAL

Network architecture is a set of design principles describing the scope, the objectives, and the abstract (i.e. high level) operation of a communication system. It defines the atomic functions and entities that compose the network and specifies the interactions which occur between these various building blocks. In ANA we have defined the networking abstractions that we consider key in order to let autonomic Functionality "play" with the various ways networks can be implemented and be operated. Our view is that ANA has to provide a minimal yet universal set of networking concepts which permits different networking styles to be expressed and allows users to access communication services in a generic manner. The core abstractions of ANA are illustrated by Figure 2, which shows the modeling of a simple unicast communication. In the next subsections, we introduce these abstractions and describe their basic usage and purpose

2.1 Network compartment and information channel (IC)

At the coarsest level, the ANA world is organized in *network compartments*. The concept of network compartment is similar to the notion of *context* as proposed in Plutarch [9] where "a context describes a region of the network that is homogeneous in some regard with respect to addresses, packet formats, transport protocols, naming services, etc". In ANA, each compartment is free to *internally* use whatever addressing, naming, routing, networking mechanisms, protocols, packet formats, etc. However in order to offer a standard access to the communication services it provides, each compartment must support a generic "compartment API"

that *wraps* its internal operation with generic constructs. In other words, each compartment operates as a black-box supporting a generic communication API (which is described in details in Section III-B). In practice, an Ethernet segment, the public IPv4 Internet, a private IPv4 subnet, peer-to-peer systems like Skype, and distributed web caching networks like Akamai, can each be modeled in ANA as a network compartment.

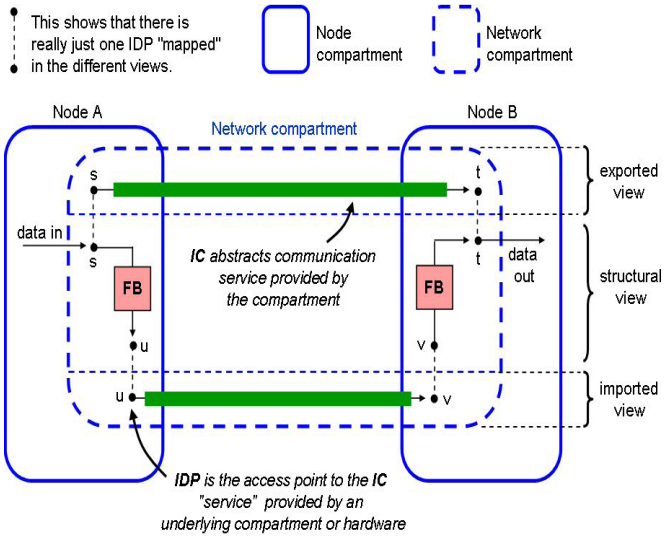

Fig.2 Fundamental abstractions of ANA

Typically in each network compartment, a distributed set of protocol entities collaborate in order to provide communication services to other compartments and applications. Whatever the nature of the communication service (i.e. unicast, multicast, datagram, reliable stream, etc) it is abstracted in ANA by information *channel* (IC). This notion is illustrated by Figure 2 where the network compartment exports a communication service abstracted by the information channel (IC). How this channel is instantiated is typically not exposed to entities not being part of the compartment which merely use the provided communication service and often do not care how this is implemented. Of course a compartment is an abstract construct which is really a collection of distributed systems. When interacting with a compartment with the generic API, an entity really interacts with a (node-local) software component implementing the operation of the compartment: the functional block (FB).

2.2. Functional Block (FB)

In ANA, any protocol entity generating, consuming, processing and forwarding information is abstracted as a *functional block* (FB). For example, an IP stack, a TCP module, but also an encryption function or a network monitoring module can be abstracted as a functional block. In contrast to OSI's protocol entities [10], functional blocks are not restricted to being abstractions of network protocols and the provided functionality can range from a full monolithic network stack down to e.g., a small entity computing checksums (like *elements* in Click [11]). In addition, a functional block (e.g. a passive monitoring module) does not necessarily provide access to some network compartment. On Figure 2, the two functional blocks providing the information channel are shown in the *structural view* of the compartment which shows how the compartment algorithms and protocols are implemented. Hence while the *export* view shows the information channel provided by the network compartment, the structural view shows the functional blocks implementing this communication service. Beside ICs and FBs, a fundamental feature of ANA is that an IC or a FB is always accessed via an indirection system that forms the core of ANA: the *information dispatch point (IDP).*

2.3. Information Dispatch Point (IDP)

IDPs are inspired by network pointers and by network indirection mechanisms and they are somehow similar to *file descriptors* in UNIX systems. Basically inside an ANA node, a functional block (FB) is always accessed via one or multiple IDPs attached to it: in other words, all interactions are carried out via an indirection level *built-in* in the network architecture. Like UNIX systems enforce access to all system components (e.g. devices, sockets, files) via file descriptors, ANA mandates that all functional blocks are accessed via IDPs. However unlike file descriptors and sockets, the binding of an IDP (i.e. the FB to which it is attached) is dynamic and can change over time as the "network stack" is reconfigured. The bindings between IDPs and FBs are stored in the core forwarding table of the ANA Node where each IDP is identified by a node-local label1. ,

Each IDP maintains some dynamic information such as the FB to which it is attached, a lifetime counter, some status information, etc. The advantage of IDPs is two-fold: first, they act as generic

communication *pivots* between the various functional blocks running inside an ANA node and, second, they provide the required flexibility allowing for the re-organization of communication paths. For packet forwarding, IDPs permit to implement forwarding tables which are fully decoupled from Addresses and names: i.e., in ANA the next hop "entity" (local or remote) is always identified by an IDP. This permits to easily add and use new networking technology and protocols as long as they *export* their communication services as IDPs. *D. Node compartment* The ANA architecture has the additional particularity of pushing networking abstractions inside the network hosts. We indeed consider a networking node to be itself a network composed by the functional blocks running on the host. As a result, every ANA node is organized as a *node compartment* which operates like any other (network) compartment except that it does not provide information channels. This permits functional blocks to discover each other and interact inside the node compartment in the same manner as with any other network compartment. Node compartments are illustrated on Figure 2 by solid-line rectangles with rounded corners.

2.4. IDPs, FBs, and ICs by example

As stated earlier, Figure 2 illustrates how the main abstractions of ANA are used to model two nodes communicating via a network compartment. On this figure, two functional blocks (shown as red squares) in two ANA nodes provide an information channel (shown as a green thick line) from node A to node B. This IC is abstracted by the IDP's' (shown as a thick dot) in node A and leads to the IDP't' in node B. The functional blocks involved are shown in the *structural view* of the compartment which shows how the compartment algorithms and protocols are implemented. The *import* and *export* views show the information channels that are respectively being used and provided by the network compartment shown. Note that the IC (used by the two FBs to communicate) is typically exported by another network compartment (not shown here). For example, the IC might be provided by an Ethernet compartment which is then used by an IP compartment to provide the IC.

All together, the core abstractions of ANA permit to model communication systems with a simple but structured framework. In particular and with the help of the communication API described in the next section, this permits to *wrap* all network compartments with a generic interface such that it becomes possible to access communication services in a "compartment-agnostic" way. This is critical in order to achieve autonomous behavior and adaptability without having to hard-code (and hence freeze) the interactions between the various (protocol) entities of ANA.

III. AFI Methodology

Figure 3 depicts the AFI Top-Down & Bottom-up Methodology. This methodology is composed of two horizontal layers, high level "operators' requirements, use cases and scenarios" at the bottom and "Generic Autonomic Network Architecture" (GANA), interconnected at the top, with each other to show that some input flows from one process to the other. A vertical layer provides inputs and enablers in terms of "Use Cases" for the bottom layer and "GANA" for the top layer. In this vertical layer we gather outcome provided by the AFI's stakeholders (research community, operators' experience, vendors' experience, customer quality of experience, etc. The outcome will be the definition of an "Implementable Autonomic Network Architecture" (IANA) composed of Autonomic functional blocks, reference points, Info Models/Data Models and associated implementation neutral specifications.

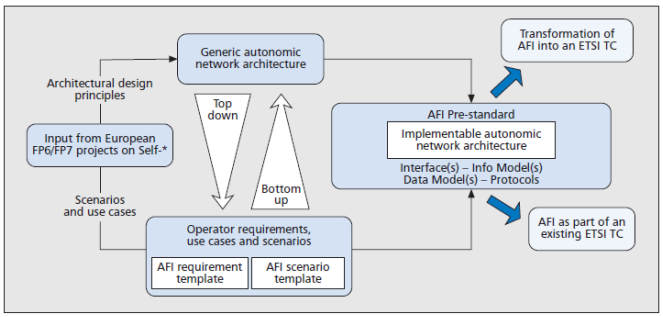

Figure 3: AFI Top-Down & Bottom-up Methodology

In order to meet the operators requirements will be defined, we need to instantiate this "Generic Autonomic Network Architecture" by applying its prescribing "generic automaticity and self-management concepts and principles" into an existing network architecture such as the NGN architecture. The result of this instantiation of GANA is an "Implementable Autonomic Network Architecture" (IANA) as a production network in the short-term, and evolves into the mid-term and then long-term. To achieve this goal, the inclusion of operators' operation requirements in this "GANA" is required. That means which Reference point should or

shall be translated into interface. And which Information Models and Policy Frameworks should be translated into related rules, data models and protocols (syntax, semantic) from implementation point of view. This is linked to the choice of the right and cost effective ones from integration point of view in the OA&M/OSS systems. This result in the "AFI pre-standard" based in what we named "Implementable Autonomic Network Architecture" as the major AFI deliverable. Besides, this instantiation will map the "GANA" on to the hierarchical management architecture at network level, service level and policy management level of an existing architecture; will result in an "Autonomic architecture" when the "generic automaticity and self-management concepts and principles" prescribed by the GANA are applied by fusion with an existing architecture in use by the industry. These instantiations or mappings must be driven by operators' use cases. The industry (Equipment suppliers, the OSS vendors) will play a major role as well. In this context, end to end inter-domain issues are also treated when it comes to deliver end to end service through numerous networks or sub-networks (in the case of portioning). Besides, building trust and confidence on the "Implementable Autonomic Network Architecture" by designing test and validation framework is a major topic of this AFI Methodology.

IV. Principles of AFI Architectural Reference Model

The goal of the AFI architectural reference model is to provide guidelines for introducing autonomic network engineering, as well as to incorporate the capabilities of cognition and self-management. The model is a set of fundamental design

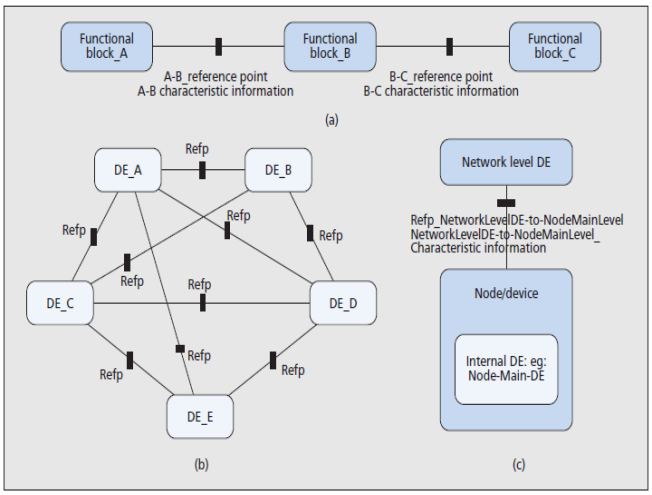

Fig.4. Example illustration of reference points in the reference model

and operational principles, describing the functions, processes, and interfaces, and so it provides Decision-making Elements (DEs), responsible for autonomic management and control of network resources. This reference model can be instantiated onto concrete network architecture to form automaticity- enabled reference architecture. The starting point for the development of such a generic model is based on the analysis of the existing research initiatives, autonomic architectures, and standards for telecommunication network operations and management, as described in Fig.5. This analysis allows for identification of the gaps between the current (non-autonomic) approaches and the requirements and expectations expressed toward autonomic networks by the actors of the telecommunications environment. The AFI Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management is based on the GANA Reference Model [5, 9]. The GANA Model has been evolved by AFI to address different, previously not considered concepts, such as the knowledge plane and cognition, impact on evolution of management paradigms, characterization of reference points, and enablers for autonomic features. The purpose of the GANA Reference Model is to serve as a blueprint prescribing the design and operational principles of autonomic decision-making managing entities, responsible for autonomic and cognitive management, as well as control of network resources such as protocol stacks and mechanisms. The aspect of controlling is understood as a notion of encompassing the features of observing, supervising, regulating, and dynamically adapting the behavior of managed resources within a control-loop [5]. The aspect of managing is, in turn, related to the traditional management plane, and some aspects of both node/element- and network-intrinsic management, without involving the Network Management System. The Reference Model is defined and refined iteratively to allow the inclusion of enhancements and revisions, as the technologies mature and contributions from the community arrive. The first release of the AFI specification will include the primary architecture, its properties, principles, building blocks, and their respective interactions.

V. Managing Future Internet using NetServ

NetServ is a programmable node architecture designed for deploying in-network services. It is suited for any types of nodes, such as routers, set-top boxes, and user equipment. NetServ includes an in-network virtualized service container and a common execution environment for both network services and traditional addressable services (e.g. a Web server). NetServ is thus able to fill the gap between these two types of services that have traditionally been kept separated in the Internet architecture. In this way administrators can be provided with a suitable flexibility to optimize resource exploitation. The NetServ prototype architecture is shown in Figure 1. It is currently based on the Linux operating system. It includes an NSIS-based signaling protocol [2], used for dynamic NetServ node discovery and service modules deployment therein. The NetServ controller coordinates NSIS signaling daemons, service containers, and the node transport layer. It receives control commands from the NSIS signaling daemons, which may trigger installation or removal of both application Modules within service containers and filtering rules in the data plane. Each deployed module has a lifetime associated with it and it needs to be refreshed by a specific signaling exchange before its lifetime expiration, otherwise it is automatically removed. The NetServ controller is also in charge of setting up and tearing down service containers, authenticating users, fetching and isolating modules, and managing service policies. Service containers are user space processes. Each container includes a Java Virtual Machine (JVM), executing the OSGi framework [3] for hosting service modules. Each container may handle different service modules, which are OSGi compliant Java archive files, referred to as *bundles*. The OSGi framework allows for bundles hot-deployment. Hence, the NetServ controller may install modules in service containers, or remove them, at runtime, without requiring JVM reboot Service modules, represented by circles in Figure 1, are OSGi bundles deployed in a service container. Figure 1 shows two types of modules:

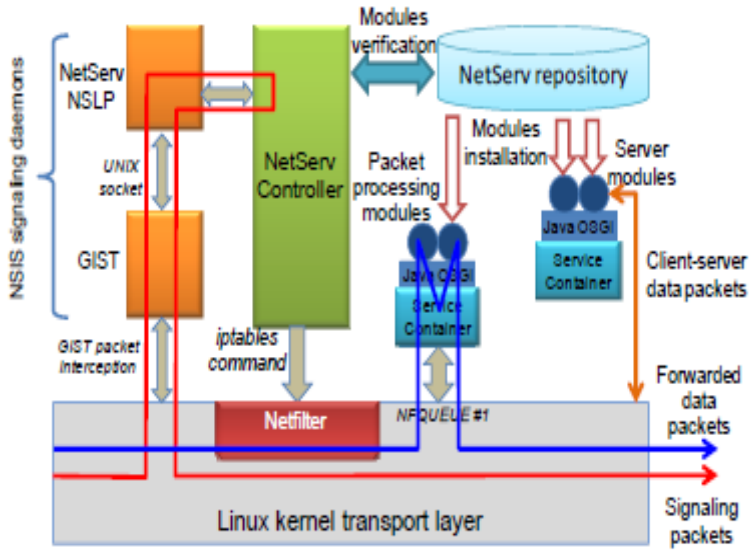

Fig.5.NetServ node internal architecture

Server modules, circles located within the upper-right service container. They act as standard network servers, communicating with the external through a TCP or UDP port.

Packet processing modules, circles located within the lower-left container. They are deployed in routers along packet path and can both inspect and modify packets in transit. The blue arrow in Figure 1 labeled "forwarded data packets" shows how an incoming packet is routed from the network interface, through the kernel, to a service container process being executed in user space. The packet is handled by two different modules before being sent back to the kernel and routed towards its final destination. The module classification as *server module* or *packet processing module* is only logical, since each NetServ module may act in both ways. This is actually an important NetServ feature since it overcomes the traditional distinction between router and server by sharing each other's capabilities. The *NetServ repository*, introduced in the NetServ architecture for management purposes, includes a pool of management programs deployable through NetServ signaling in the NetServ nodes present in the managed network. Currently, the Linux kernel is used to implement the NetServ transport layer. Packet filters, used to intercept packets in the NetServ node, and rules, used to route them to the proper service container, are installed in the node forwarding plane by using the *net filter* library through the *ip table's* tool.

VI. ANA and Future Internet

ANA as a meta-architecture for networks
1. Extracts and refractors core networking concepts,
2. Supports heterogeneous addressing and naming styles:
3. ANA is not address or protocol centric
ANA as a driver to promote disruptive NW research.
1. No IP-backward-compatibility constraints.
2. Allows new ideas and experiments to emerge and interact

VII. Current Internet and Future Internet

One of the key aspects fundamentally missing from the current Internet infrastructure is an advanced service networking platform and facilities, which take advantage of flexible sharing of available connectivity, computation, and storage resources. The current Internet has been founded on a basic architectural premise, that is: a simple network service can be used as a universal means to interconnect both dumb and intelligent end systems. The simplicity of the current Internet has pushed complexity into the endpoints, and has allowed impressive scale in terms of inter-connected devices. However, while the scale has not yet reached its limits, Internet use is expected to grow massively over the next few years with an order of magnitude more Internet services, the interconnection of smart objects from the Internet of Things, and the integration of increasingly demanding enterprise and societal applications. The Future Internet research and development trends are covering the main focus of the current Internet, which is connectivity, routing, and naming as well as defining and design of all levels of interfaces for Services and for networks' and services' resources. As such, the Future Internet covers the complete management and full lifecycle of applications, services, networks and infrastructures that are primarily constructed by recombining existing elements in new and creative ways. The aspects which are fundamentally missing from the current Internet infrastructure, include the advanced service networking platforms and facilities, which take advantage of flexible sharing of available resources(e.g. connectivity, computation, and storage resources).Due to the existence of multiple stakeholders with conflicting goals and policies, modifications to the existing Internet are now limited to simple incremental updates and deployment of new technology is next to impossible and very costly. In-Network clouds have been proposed to bypass this ossification as a diversifying attribute of the future inter-networking and inter-servicing paradigm .By allowing multiple heterogeneous network and service architectures to cohabit on a shared physical substrate, In- Network virtualisation provides flexibility, promotes diversity, and promises security and increased manageability.

In-Network clouds can be defined as an integral part of the differentiated Future Internet architecture, which supports multiple computing clouds from different service providers operating on coexisting heterogeneous virtual networks and sharing a common physical substrate of communication nodes and servers managed by multiple infrastructure providers.

By decoupling service providers from infrastructure providers and by integrating computing clouds with virtual networks the In-Network clouds introduce flexibility for change. In-Network Network and Service Clouds can be represented by a number of distributed management systems described with the help of five abstractions: Virtualisation Plane (VP), Management Plane (MP), Knowledge Plane (KP), Service Plane (SP), and Orchestration Plane (OP). These planes are new higher-level artefacts, used to make the Future Internet of Services more intelligent, with embedded management functionality. At a logical level, the VMKSO planes gather observations, constraints and assertions, and apply rules to these in order to initiate proper reactions and responses. At the physical level, they are embedded and execute on network hosts, devices, attachments, and servers within the network. Together these distributed systems form a software-driven network control infrastructure that will run on top of all current networks (i.e. fixed, wireless, and mobile networks) and service physical infrastructures in order to provide an autonomic virtual resource overlay**.**

VIII. CONCLUSION

The Internet has to change to satisfy new requirements and go beyond its limitations**.** It is an implementation oriented solution for autonomics and self management. Their is need to improve our internet so can our physical infrastructure will build, it will not only save our time but also increase the speed of the nation. The only way you can predict the future is to build it as said by Alan Kay

REFERENCES

Report

[1] D. Clark, et al., "newarch: future generation internet architecture", new arch final technical report,

White paper

[2] Autonomic communication, white paper, fraunhofer focus November 2004.

Ieee paper

[3] IEEE 802.11: "ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: wireless lan medium access control (mac) and physical layer (phy) specifications".

[4] Etsi gs afi 002: "autonomic network engineering for the self-managing future internet (afi); generic autonomic network architecture".

[5] J.w. lee et al., "NetServ: active networking 2.0", future net iv 2011vkyoto, japan, june 2011.

IEEE magazine

[6] X.. Fu et al., "nsis: a new extensible ip signaling protocol suite", ieee communications magazine, 43(10), 2005, pp. 133- 141.

Journal paper

[9] P.. Gouvas et al., "integrating overlay protocols for providing autonomic services in mobile ad-hoc networks," ieice communications, special issue on implementation, experiments, and practice for ad hoc and mesh networks,Vol. 93-b, aug. 2010, pp. 2022–34.

[10] R. Chaparadza et al., "towards the future internet — a european research perspective," ios press, 2009, chapter: "creating a viable evolution path towards self-managing future internet via a standardizable reference model for autonomic network engineering," pp. 313–24.

[11] N. M. K. Chowdhury and r. Boutaba, "a survey of network virtualization," computer networks, vol. 54, no. 5, 2010, pp. 862–76.

[12] M. Blumenthal, d. Clark, "rethinking the design of the internet: the end to end arguments vs. The brave new world", acmtransactions on internet technology, vol. 1, no. 1, pp. 70-109, aug. 2001

[13] *A. Fe ldman, "internet clean-slate design: what and why?" Acm sigcom computer communication review, vol. 37, issue 3, 2007 112-116.*