On analyzing problems of distributed systems and current internet in front of the future internet architectures

Antonio Marcos Alberti [1]
Marco Aurelio Favoreto Casaroli [1]
Rodrigo da Rosa Righi [2]
Ivam Guilherme Wendt [2]
Dhananjay Singh [3]

Abstract: Hoje em dia, existem centenas de projetos em curso em todo o mundo para redesenhar ambos os protocolos de comunicação e arquitetura da Internet. Iniciativas são coletivamente chamadas de pesquisa sobre a " Internet do futuro ". A maioria dessas iniciativas dependem de sistemas distribuídos existentes, que, muitas vezes, limitam ou mesmo impedem o desenvolvimento de soluções totalmente inovadoras. A razão principal é que a grande maioria dos sistemas distribuídos são firmemente ligados com a pilha de protocolos TCP/IP. Neste artigo, será fornecido um debate de primeiro olhar sobre as relações entre as pesquisas sobre a Internet do futuro e sobre sistemas distribuídos, com foco em dependências e requisitos semelhantes entre essas áreas. A partir desta análise, torna-se evidente que muitos dos requisitos da Internet do futuro (e desafios abertos) são repetidos no panorama de sistemas distribuídos. Embora existam muitos estudos em ambas as frentes de pesquisa individualmente, o estudo dos principais desafios da Internet futuro, a tratar os requisitos de sistemas distribuídos é um tema ainda pouco explorado na nossa pesquisa contemporânea. Este trabalho tem como objetivo determinar as lacunas e requisitos que a Internet do futuro deve cumprir a fim de apoiar os sistemas distribuídos do futuro. Para apoiar este objetivo, um conjunto de métricas de projeto são identificados e um espaço de design convergente é proposto.

***Palavras-chave*:** Redes de Computadores. Sistemas Distribuídos. Arquitetura da Internet do Futuro. NovaGenesis.

Abstract: Nowadays, there are hundreds of underway worldwide projects to redesign both communication protocols and architecture of the Internet. These initiatives are collectively called "future Internet" research. Most of these initiatives rely on existing distributed systems, which often limit or even prevent the development of "clean slate" solutions. The main reason is that the great majority of distributed systems are tightly-linked with the TCP/IP protocol stack. In this article, we provide a first glance discussion on the relationships between future Internet and distributed systems research, focusing on dependencies and similar requirements among these areas. From this analysis, it becomes evident that many of the future Internet requirements (and open challenges) are repeated in the distributed systems landscape. Although there are many studies on both research fronts individually, the study of the key challenges of future Internet when addressing distributed systems requirements is a topic yet not explored in our contemporary research. This paper aims at determining the gaps and requirements future Internet must fulfill in order to support future distributed systems. To support this objective, a set of design metrics are identified and a convergent design space is proposed.

***Keywords*:** Commputer Networks. Distributed Systems. Future Internet Architecture. NovaGenesis.

[1]Instituto Nacional de Telecomunicações (INATEL) - Av. João de Camargo 510 - Santa Rita do Sapucaí (MG) - Brazil
{alberti,favoreto@inatel.br}
[2]Universidade do Vale do Rio dos Sinos - Av. Unisinos 950 - São Leopoldo (RS) - Brazil
{rrrighi@unisinos.br,ivamwendt@yahoo.com.br}
[3]Hankuk University of Foreign Studies - 107 Imun-ro, Dongdaemun-gu - Seoul - South Korea
{dsingh@hufs.ac.kr}

http://dx.doi.org/10.5335/rbca.v8i3.5889

1 Introduction

The Internet is no longer a network of fixed computers among governmental institutions. It invaded most aspects of our life and society, changing lifestyles, business practices, work relations between employers and employees, communication channels and even social interactions. Technically, the Internet can be seen as a single communication subsystem providing communication among all of the hosts that are connected to it [1, 2]. Modern Internet's facet concerns a vast collection of interconnected computer networks of many different types, including, for example, a wide range of wireless communication technologies such as wireless LANs (Wi-Fi), worldwide interoperability for microwave access (WiMAX), bluetooth and fourth-generation (4G) of mobile phone networks. Differences among the networks are masked by the fact that all of the computers attached to them use the Internet protocols to communicate with one another. While the usage of standardized and widespread adopted protocols are useful for connecting devices, the price for accomplishing portability is high [3, 4, 5]. The protocols of the TCP/IP reference model are organized in a single, monolithic stack, in which the most of the layers are processed in software [6]. Although this model presents four layers originally, Figure 1 illustrates the five-layered common vision of both physical and virtual communication paths when using TCP/IP. TCP/IP layers and the Internet itself were designed in an era where technological development was completely different from today [7]. The computing and communications capacities were much more limited.

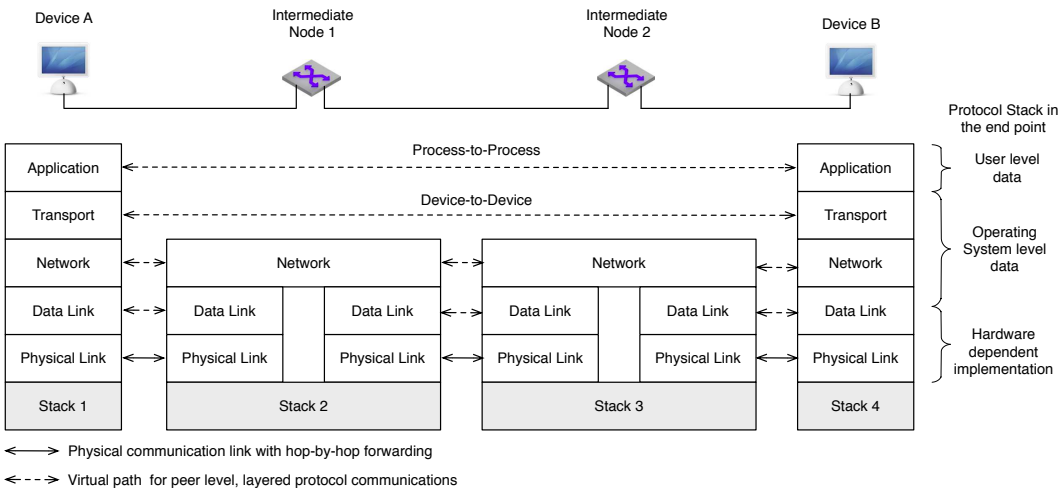

Figure 1: TCP/IP layered protocol communication between two end point devices. Functionally, intermediate nodes only require the bottom three layers of the reference model.

The current Internet is also a very large distributed system. It enables users, wherever they are, to make use of services such as the worldwide web (WWW), email, virtual conferences, and file transfer. We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages [8, 9]. They can incorporate large numbers of nodes and provide distributed system services for global organizations. Building on this foundation, large-scale distributed systems started to emerge in the 1990s in response to the dramatic growth of the Internet during this time. More recently, it is possible to observe some trends such as the extensive use of both cloud computing tools and mobile commerce environments, as well as the diffusion of big data as a mandatory science for successful business companies [10, 11, 12].

As the Internet was opened to the general public and began to be used for an increasing diversity of applications, some limitations of its design have become increasingly apparent. Internet naming is limited to hosts, ports, and web resources. Services and applications are not directly named, being identified by socket names (e.g. 192.168.10.88:8000) or world wide web resources (www.example.com). When a computer moves from one network to another, its IP address changes, disconnecting the sockets, making applications unreachable. To access contents and services users need to inform where they are. A certain content or service can be very close to some user, but since they cannot be accessed by their own names, users need to inform where they are in advance. Content provenance and integrity are not provided by the TCP/IP stack. Volatile IP addresses and lack

of integrity check on IP layer left these requirements to be implemented by web applications. Named-content [13] and named-services [14] are emerging approaches to deal with the expressiveness limitations of the current architecture.

Another limitation of the current Internet is related to network controllability and elasticity of physical resources. Network and cloud operators, as well as service and content providers are claiming for more agility on deploying new networking functionalities, such as routing, switching, payload processing, etc. The current Internet architecture has experienced significant changes on the top and bottom layers, but its is still very difficult to change the core protocols, like the IP. A proof of this innovation barrier is the IPv6 proposal. There are decades of tentative to introduce IPv6 and its new functionalities. Meanwhile, new paradigms have emerged in an attempt to facilitate the deployment of new technologies on the Internet. An example is the so called software-defined networking (SDN), which is an initiative to facilitate networking control while introducing new frame forwarding mechanisms. SDN is being adopted by industry to provide more agility on network operation, consequently increasing the role of software on Internet architecture. It enables new technologies to be deployed at software level, as applications of a network operating system (NOS) — an analogy to the computers operating systems. The distributed state at equipment control plane are replaced by a logically centralized view of the network, which is implemented in a software controller. The controller configures network equipment using a configuration protocol improving controllability and elasticity of physical resources.

Concerned with this reality and its possible implications in next decades, several initiatives emerged worldwide to discuss the Internet architecture under the banner of the so called "**future Internet architecture**" (FIA) design [15, 16, 17]. The term was adopted in the first initiatives aimed at rethinking the Internet, including the 4D project [18], the future Internet design (FIND) initiative [19], the global environment for network innovations (GENI) [20], and the European future Internet assembly (FIA) initiative [21]. By future Internet, we mean any Internet-like network that could emerge in the future. This includes evolutionary approaches, where the fundamental protocols of the current Internet are maintained and where new ideas are introduced incrementally, or revolutionary approaches, where the architecture is redesigned from scratch. The revolutionary approaches are usually called as "clean-slate" architectures [22].

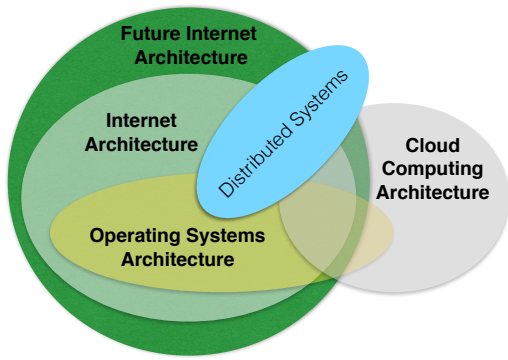

Figure 2: Information architecture and their relative scopes.

Figure 2 illustrates some of the most important information architectures we have and their relative scopes, as well as their relationship with distributed systems concept. All these architectures are becoming strongly software-based, overlapping one another, and becoming deeply related to distributed systems (DSs), since in essence they are all DSs, and most often their implementations depend on other DSs. Thus, the quest for future architectures is not of a matter of mutually exclusive efforts, but instead a matter of essentially and synergistically combining all of them. Therefore, considering the aforementioned context, this article presents a first glance discussion on the relationships between FI and DS research, focusing on dependencies and similar requirements among these areas. We would like to determine the gaps and requirements FIAs must fulfill in order to support future DS. To meet this objective, a set of design metrics are identified and a convergent design space is proposed. As far we know, this is the first paper in this subject. The motivation for the work is clear: *when you think a new model for the Internet it is impossible not considering its impact over the functioning of existing DSs. Moreover, a*

new Internet may benefit from the state-of-the-art of underlying DSs.

The remainder of this article is organized as follows. Section 2 discusses about the fundamental challenges in the distributed systems and current Internet architecture, defining a common design space and metrics that FIAs should support for DS future. Section 3 presents four FIAs that address the raised gaps and requirements, comparing them according to the proposed design space. Finally, Section 5 presents the final considerations, emphasizing both the main contributions of the article and future work.

2 Limitations of Distributed Systems and the Current Internet Architecture

The overlapping between DS and Internet architecture is historic. Many aspects of distributed systems have been considered at the time the Internet protocols were designed. Examples are sockets and inter-process communication. The contrary is also true. Internet emergence facilitated the overspread of complex DSs, which take deep advantages of its application layer. As it happened in the past, we believe that the FIAs will play a decisive role in future distributed systems. In this context, this section presents a set of fundamental requirements of DSs that will challenge FIAs. Or in contrary, DS requirements FIA should support. The analysis of gaps and requirements between both research topics will be limited to a common set of metrics, defining a limited design space.

2.1 Naming and Naming Resolution: The Claim for Expressiveness

Naming is related to language, i.e. to formulate a name requires a language. According to Day [23], in 1892, Frege defined a name as:

Definition 1. *“A proper name (word, sign, sign combination, expression) expresses its sense, means or designates its meaning. By employing a sign we express its sense and designate its meaning."*

From this definition, it is clear that a name is related to meaning — the semantics of a language. In this sense, names are used to distinguish among entities, since they can have different meanings in a language. In the scope of computer systems and communication networks, the roles of names have been discussed by Saltzer [24][25], Shoch [26], Hauzer [27], Day [23], and Chun [28], among others. Based on these works, we define a name as:

Definition 2. *"A name is a set of characters in a language used to denote an entity."*

Therefore, names denote entities (or a group of entities) and can be used to define them as the target of a communication effort. In this case, to denote means to represent by signals or to attribute significance, meaning. Thus, a name is required by any entity candidate to be the target of some communication effort. In this context, what does it means expressiveness?

Definition 3. *"Expressiveness is the degree to which some entity can express its intents using names."*

For example, in the current world-wide web (WWW) users can express their communication intents using uniform resource identifier (URI), such as "www.inatel.br". Or a web browser can express its communication intents identifying a *socket* to some web server. The socket is the combination of a network address, the transport port number and a transport protocol option (TCP or UDP). In other words, service identifiers are coupled to host names, limiting the expressiveness services can have, since to point some target service for communicating requires to have its node address. Ideally, services would determine target services using service identifiers that are independent of node identifiers. This would allow a service to specify another target service regardless of where it is. Besides URIs and sockets, the current Internet also supports domain names.

The current Internet lacks on expressiveness. Users cannot express their intentions using services, contents, or device names directly, independently of network addresses or domain names. For instance, to download some desired content, an user should express its intent using a URI. Or to search in the web browser using natural language keywords (names) to obtain a ranking of candidate URIs. To express the intent to communicate with some device, its network address should be provided.

Information-centric networking (ICN) [29] claims for expressing intents directly using content names instead of host addresses. The idea is that no matters where the content is, but what is the name of the desired content. Named-data enables perennial access to entire content objects or individual data chunks. Service-centric networking (SCN) [14] proposes a similar approach for service naming. Services express their communication and coordination intents directly using service names.

Besides limited naming and expressiveness, another limitation of current Internet is related to name resolution. The current Internet employs a hierarchical naming scheme, where names are attributed to represent hosts and networks. For example, the name host.somedomain.com.br denotes a host that can be found inside a network named somedomain.com.br. These names are mapped to IP addresses by the domain name system (DNS). In this case, hierarchical names are used to denote hosts and networks. The DNS is restricted to resolve host and network natural language names to hierarchical IP addresses. There is no DNS for individual services or content names. This limits the expressiveness of services directly to other services or contents. ICN employs content name to storage location resolution, while SCN accommodates service name to host location resolution.

Many distributed systems bypass the restricted Internet support for naming and name resolution implementing their own naming approaches. Examples are peer to peer systems, which use self-verifying names (SVNs) to locate content chunks by their names, such as Chord [30] and Kademlia [31]. Distributed systems for Internet of things (IoT) [32] are also advancing by themselves towards more expressive architectures. As an example, there is the data distribution system (DDS) [33], which adopts a "global name space", where named topics are used to provide the rendezvous among data writers and readers. Topic names where not supported by DNS. The DDS names enable nodes to express their interests to the network, enabling one node to discover possible peers. For this purpose, nodes employ natural language names. The DDS allows services to express their intents directly, decoupled to network addresses.

2.2 Limited Support for Data Integrity and Provenance

Internet architecture was designed when computing and communication resources were very limited. The support for data integrity and provenance was not a priority in the 70's. Data retrieval in today's Internet typically involves web protocols, like HTTP, where content/services are accessed via uniform resource locators (URLs). Naming follows domain name system (DNS) hierarchical structure. IPsec [34] is being incrementally added to provide a set of security services at the IP layer, specially when creating virtual private networks (VPNs). IPsec provides several hashed message authentication codes (HMAC) to check message integrity. Cryptographic hash functions, like MD5 or SHA-1, are employed to calculate an HMAC. Provenance is provided by authenticating the source of the datagrams.

In content-centric networking (CCN) [35, 36] information objects are named and secured independently of network connections. According to Ghodsi et al. [36], CCN requires information objects to be directly validated, independently of location (hosts holding object copies). Van Jacobson et al. [35] contends that "trust in content is easily misplaced, relying on untrustworthy location and connection information". CCN (or ICN) argues on replacing *where* by *what*, protecting and guaranteeing provenance of content itself.

In this matter, naming and name structure is fundamental. The need to verify content integrity and provenance motivated many ICN approaches to employ self-verifying names [36]. Ghodsi et al. contended that SVNs offer better security properties than NLNs. Convergent studies on the role of SVNs on FI and DSs is nonexistent — an open research opportunity.

2.3 Lack of Mobility Support

The Internet emerged in a technological environment where there is no mobile computing. Mobile computing concerns the execution of computing tasks while the device is on the move, or visiting places other than their usual environment (roaming) [37]. The main idea is to offer both location and access transparencies by providing the access to the Internet, or Intranet, anywhere at any time. Nowadays, we can observe the exponential growth of tablets and smartphones usage, which normally use a wireless connection ($i.e.$ IEEE 802.11) or third/fourth generation of mobile telephony technology.

These technologies suffer with the poor support that the original Internet gives for mobile computing. When a node moves, its network address must be changed in order to receive datagrams on the new location. However, this change on Internet address affects the service socket. Thus, in current Internet a service name depends on node address. The root of this limitation is the double semantics of IP address. It serves as host identifier (ID) as well as a locator (LOC). Let's define both to clarify this point. From Chun et al. [28]:

Definition 4. *An identifier unambiguously identifies some existence from others in some scope.*

This means that an identifier needs to point to just one existence in a certain population. Thus, a name can be an identifier if it points to just one existence in a determined scope.

The definition of a locator requires the notion of a locator space. Also from Chun et al. [28]:

Definition 5. *"The space of a locator is the set of all possible positions an entity can inhabit and the relation defined on the positions."*

Thus, considering the Definition 5 it is possible to define a locator as:

Definition 6. *A locator points the position of some entity in a given locator space.*

Interestingly, this demonstrates that names can be used as locators. To be a locator, a name needs to point to an unambiguously position in the space, as well as they should allow determining the distance between entities using some distance metric.

Considering these definitions, mobility support requires to move entities keeping their IDs fixed and valid, while changing their LOCs consistently to new positions. Many incremental proposals have been developed to improve host mobility support on the Internet, directly affecting consistency of network connectivity to socket services. Examples are mobile IP (MIP) [38], locator/identifier separation protocol (LISP) [39], host identity protocol (HIP), mobility and multihoming supporting identifier locator split architecture (MILSA) [40], among many others. A common issue is that all these approaches are focused on host mobility. However, there are many other important entities on the Internet that deserve similar treatment: virtual machines, services, and contents. The mobility support directly affects performance and availability of mobile computing and is related to name resolution system (NRS), since IDs and LOCs are names and ID/LOC splitting is better implemented by name rebinding as in the aforementioned proposals.

The support for host mobility turns possible a mobile terminal to disconnect from one wireless network and to connect to another seamlessly, without requiring any intervention from the user. The presence of computers everywhere only becomes useful when they can communicate with one another. In this way, the network core infrastructure must deal with this growth of devices by providing multihoming, with simultaneous connectivity over heterogeneous networks.

2.4 Lack on Exposition, Flexibility, and Elasticity of Hardware Resources

Networking infrastructure is becoming more and more complex, requiring a lot of human intervention (operational costs). New protocols were created to fulfill gaps in previous designs without adequate analysis of unintended effects. Many design inconsistencies were created, increasing inefficiency and complexity of the current protocol stacks. We became masters at keeping this complex network running and little has been done to extract simplicity from the lessons learned in the last decades [41]. In this context, some networking researchers, like Scott Shenker, Nick McKeown, and Martin Casado, to name a few of them, have proposed a revision of the current networking design abstractions. They defend the idea that abstractions have played a big role on computer science for decades, shielding high-level software from the complexity existing in lower levels. Therefore, they contend that it is time to apply the “power of abstractions" for networking, since the role of software on the contemporary technologies is increasing considerably. They propose the notion of software-defined networking (SDN).

The first target of SDN proposal defended by Shenker et al. [41] is on simplifying network control. For this aim, four abstractions have been proposed: (i) forwarding abstraction, which encompasses a flexible, soft-ware-controlled, frame forwarding model to decouple data plane from control plane; (ii) a state distribution abstraction, which comprehends a centralized control program that operates over a summarized network view, aiming at reducing the in-consistencies of distributed states (as in the current Internet data plane); (iii) a configuration abstraction, where the output of the control program is transported to the controlled equipment by a control protocol, creating what can be called a net-work operating system (NOS); and (iv) a specification abstraction, that enables the generation of abstract configurations for network devices, enabling the creation of virtual topologies over the physical ones. The distributed states at equipment control plane are replaced by a logically centralized view of the network, which is implemented in a software controller. The controller configures network equipment using a configuration protocol. OpenFlow is the first standard protocol developed to this purpose.

These abstractions allow to speed up: (i) the deployment of new network functionality as NOS applications; (ii) flexible forwarding tables using rules defined in software; and (iii) facilitates on demand hardware resources allocation. However, state-of-the-art SDN does not accommodate controller-as-a-service (CaaS) [42]. Controllers are not integrate to service-oriented philosophy, like service-oriented architecture (SOA). Additionally, OpenFlow does not include mechanisms to expose hardware resources, i.e. capabilities like number of ports, bit rates, etc. Hardware exposition to software orchestration is not an abstraction in current SDN. Service-oriented controllers could expose hardware capabilities enabling dynamic composition of physical resources, maximizing flexibility and elasticity.

3 Future Internet Architectures

In this section we present a selected group of FIAs considering the distributed systems’ point of view, namely: NovaGenesis [43], scalable and adaptive Internet solutions (SAIL) [44], expressive Internet architecture (XIA) [45], and recursive Internet architecture (RINA) [46].

3.1 NovaGenesis

NovaGenesis (NG) design started with a conceptual-driven survey on FI requirements, technologies, and challenges [15]. Based on this background, a set of promising ingredients to solve the identified requirements and challenges was chosen. In 2012, there was a search for synergies among selected ingredients for future convergent information architectures [47]. The results of this study fueled NovaGenesis design, by defining a set of fundamental distributed systems where any information treatment or exchanging is seen as a service. NovaGenesis is deeply founded on naming and naming resolution. Names are representations used to denote one or more individual existences. In this case, *to denote* means to represent something by signals. By definition, names denote meaning and sense. Therefore, naming is the act of attributing names, while a naming structure is an scheme to denote existences following some planned strategy. NovaGenesis supports any kind of naming and naming structures.

Examples of naming approaches supported by NovaGenesis are: a) natural language names (NLNs) and

b) self-verifying names (SVNs). NLNs are used by humans on their languages, e.g. the word "Host" in English language. SVNs can be obtained as the output of a mathematical hash algorithm. A binary word (also called hash code) is obtained by hashing some input binary pattern. This input binary pattern can be some immutable attribute of a certain entity, like the iris pattern of a person or the complete data chunk of some coded picture, e.g. a jpeg file. They are said self-verifying since at any time they can be hashed again generating the same name. Therefore, the binary input of the hash function can be the existence itself ($e.g.$ computer program executable, source code, or information files) or other binary input related to the existence being named (e.g. existences' immutable attributes).

The NG's key design abstractions are: (i) name binding (NB); (ii) distributed name bindings graph (DNBG); (iii) protocols-implemented-as-a-service (PIaaS); (iv) publish/subscribe service (PSS); (v) generic indirection resolution service (GIRS); (vi) hast table service (HTS); (vii) name-based publish/subscribe application programming interface (NB-Pub/Sub-API); and (viii) proxy/gateway service (PGS). A NB is a mapping among two or more names or even a name to some data object. For example, the name "Einstein" can be bound to the name "Relativity". NovaGenesis accommodates natural language or self-certifying semantic operators as name bindings. Two examples of semantic operators are "is contained" and "contains". An NB can represent that the city named "Houston" *is contained* at the "Texas" state. Or that the state "Texas" *contains* the city of "Houston". Name bindings can be related to semantic operators, greatly expanding expressiveness in the network. NovaGenesis creates a distributed, gigantic, world wide web of NBs to represent all existent entities, from cars up to people, house appliances up to networking devices, contents up to services. This world wide web of names is called DNBG. By scanning the name bindings graph, NovaGenesis entities can rank "equivalence" among named semantic operators.

The exponential advances in computational capability allows more and more network functions (like forwarding, routing, error check, fire walling, etc). to be implemented as software. It is a phenomena some authors are calling "softwarization" [48]. NovaGenesis assumes that a large portion of the current networking stacks can be implemented as software. It also advocates for employing modern software-as-a-service (SaaS) paradigm to implement every information processing as a service. This includes protocol implementations in a new idea that is called PIaaS. PIaaS means to implement networking protocols as services. New protocols could be implemented by flexible software components that are dynamic linked at run time. For instance, only the necessary protocols for a particular purpose are combined, passing the idea that the protocol stack is on-the-fly rewritten as needed. The PIaaS opens a new world of possibilities in protocol design, since the communication stack can be redefined dynamically. Besides network protocols, this principle is also applied to any architectures' services or applications. Very importantly, NovaGenesis services (and protocol implementations) need to establish service contracts (agreements) before working together. This forms a trust network of services.

The name bindings can be published to other services through a PSS. In other words, a service (or a protocol implementation) can expose its NBs to other services. This is done by publishing them to a PSS. Thus, other services can subscribe a services' NBs. The PSS does the rendezvous between publisher and subscriber services, enabling them to discover how published names are related each other in a secure way. Eventually, services can successively subscribe NBs to identify and locate other existences, storing these name-bindings and routing information based on them.

The PSS forwards the NBs to the GIRS, which selects an instance of HTS were finally the NBs are stored. Thus, the PSS does not stores any published NB. It forwards to a GIRS instance which selects the proper HTS where the NB is stored. Typically, a NovaGenesis administrative domain can have several PSS, GIRS, and HTS instances. The HTS provides a mechanism to retrieve already published NBs and deliver them directly to an authorized subscriber. To improve the scalability of the design, NBs are categorized and the HTS implements a separated hash table to each NB category. Also, published NBs and content can be removed to avoid scalability issues. The PSS functions are exposed to application developers via a name-based-Pub/Sub-API. This API has six methods: (i) publish NB (with associated data, if any); (ii) publish NB/data and notify other services about a publication; (iii) subscribe a NB/data; (iv) subscribe a NB/data and notify other services about a subscription; (v) deliver a subscribed NB/data; and (vi) revoke a NB/data publication. The PSS can be seen as a distributed API, accessed via PSS identifiers.

The latest NovaGenesis abstraction is the PGS. It encapsulates NovaGenesis messages over traditional (or new) networking technologies. Therefore, it is a gateway for NG messages that leave or enter the NovaGenesis cloud. Additionally, the PGS is a proxy for other NG services inside some computer. It represents these services during bootstrapping, forwarding public NBs to other friend PGSs. The PGS has an specific component that opens

and maintains sockets in the computer operating system to enable NG messages encapsulation. It implements a raw socket that bypasses TCP/IP stack, enabling NG messages to be transported directly over Ethernet or Wi-Fi. However, for compatibility reasons it also implements TCP and UDP sockets.

3.2 SAIL

The SAIL project was funded by the European Commission's 7th Framework Program (FP7) and brings together 25 partners (operators, vendors and research institutions) aimed at designing a future network architecture in Europe. It encompasses three key technologies, namely: network of information (NetInf) [29], cloud networking (CloNe), and open connectivity services (OConS). The NetInf defines an ICN were named data objects (NDOs) can be accessed without a priori localization, i.e. the data is localized only during final transfer phase. Network caching is used to store NDOs closely to the interested peers, maximizing content distribution efficiency and robustness. The NetInf nodes provide two services: (i) the forwarding of NDOs requests to the closest caching/storage nodes and then the transfer back of the NDOs to the requester; (ii) a name resolution service (NRS) that can be used alternatively to the name-based routing (NBR). The NetInf protocol covers publishing, searching, and subscription for NDOs.

The NetInf [29] relies in a convergence layer (CL) to transport its messages among NetInf nodes. Current realizations of CL include support for HTTP over TCP, UDP, disruption/delay tolerant network (DTN), bundle protocol (BP), and Ethernet. In OpenNetInf [49] implementation, nodes communicate via HTTP or TCP. They can also use Google protocol buffers (GPBs) [50] or XML [51] for the serialization of the messages. OpenNetInf nodes are implemented using *Guice* [52], a Java framework developed by Google. Nodes perform search by using a semantic web query language called SPARQL [53]. A SDB database is used to store data on resource description framework (RDF) [54] format. The NRS is implemented in Java using FreePastry [55] and MDHT [56].

The OpenNetInf nodes have an event-driven service that can be used to publish, notify, and subscribe NDOs. The aim is to update applications regarding events related to NDOs of interest. The implementation has an event service framework (ESF) that can use one or more available underlying event-driven open source solutions, like the Java implemented scalable internet event notification architectures (SIENA) [57] or Hermes [58]. Both SIENA and Hermes were developed for the current Internet protocols. Interestingly, the ESF enables applications to use both event-driven systems for their own purposes.

The OConS covers open connectivity services for heterogeneous networking resources. It provides support for multi-path, multi-layer, multi-domain, multi-protocol routing for CloNe and NetInf. It includes services for dynamic and distributed mobility and multihoming management, as well as transport network self-management and autonomic control of data center interconnectivity. The OConS is implemented by three functional entities, namely: information management entity (IE), the decision making entity (DE), and the execution and enforcement entity (EE). The IEs manage information collection, aggregation, and preparing for decision making. The DEs make transport decisions considering rules, goals, policies, and preferences. The EEs execute the decisions, enforcing control and management. The model creates an hierarchy of control loops, where open APIs are employed at every interface, including an open API for the entire OConS component.

The CloNe integrates cloud computing and virtual networking to deal with the complex life-cycle of virtual machines and their networking requirements. It introduces the notion of a flash network slice (FNS), which can provide customized, on demand, virtual networking among different provider clouds. CloNe uses implementation specific virtualization interfaces, such as *libvirt*, cloud data management interface (CDMI), Amazon simple storage service (S3), Open Cloud Computing Interface (OCCI), Amazon elastic compute cloud (EC2), libCloud, etc. CloNe provides an application deployment toolkit (ADT), which includes a link driver to handle DHCP and SDN configurations.

From CloNe and OConS point of view, NetInf is just an application that requires specific communication support. Meanwhile, CloNe can support NetInf scalable deployment at data centers, managing NetInf components life-cycle and virtual connectivity by creating specific FNSs. From OConS point of view, both NetInf and CloNe are possible clients, since both create overlay communication topologies over the physical substrate. Both also provide discovery mechanisms, which can be seen as duplicated. Importantly, OConS makes no use of NetInf services.

3.3 XIA

The XIA is a clean slate *all in one* Internet architecture funded by the USA national science foundation (NSF). The key XIA concepts are: (i) unique identification, (ii) expressing intent, (iii) flexible addressing, (iv) iterative refinement, and (v) intrinsic security. XIA employs global unique identification of networking principals, i.e. administrative domains, hosts, services, and contents. Self-certifying identifiers (XIDs) are calculated by hashing the public cryptographic keys of domains, hosts, and services or the entire binary pattern of contents. Observe that not only XIA [45], but also NovaGenesis borrowed this idea from previous work, including [59].

To express intent means to give clues to the network — XIDs — that point clearly the desired communication targets (e.g. content, service, host, and domain). For example, when searching for some content using a content ID (CID), an interested client would provide a path that contains the domain ID (AD), host IDs (HIDs), and service IDs (SIDs) where the content can be found. Considering these information, the network can better address the task of satisfying users intents.

XIA provides a flexible addressing approach where alternative forms of accessing some communicating target (fallbacks) are provided. In other words, XIA routers can use more than one destination address to forward packets to a desired target. Therefore, if some ID is unknown at some router, alternative addressing can help to found the desired communicating target (a CID for example). A fallback allows alternative measures in the case a router cannot proceed using a preferred intent. Depending on available XIDs on a router, it can iteratively refine the intent by using other XIDs to determine the next hop. For instance, if some router does not know how to route to a certain CID, it can use the AD provided in the address. If the AD cannot route to the desired CID, the router forwards the packet to the informed HID. By using XIDs on packet headers, XIA provides intrinsic security, since address veracity can be verified anytime.

XIA proposes a principal-independent expressive Internet protocol (XIP) that covers flexible addressing, packet format, routing based on XIDs, and in-network caching of data chunks. The current XIA prototype provides a socket API - similar to the UDP socket. Also, three transport protocols are provided: X-datagram protocol (XDP), X-stream protocol (XSP), X-chunk protocol (XChunkP). The XDP is an unreliable connection less message forwarding protocol. The X-stream is a connection oriented reliable message forwarding protocol. Finally, the XChunkP provides reliable content transfer from in-network caches. XIA authors call this mechanism as *on-path interception*. XIA routers and hosts can cache content chunks that are transferred to requesters based on their CIDs. Thus, if a content is available in the path to some content server, XIA provides network stored copies of the data chunks to the interested clients. Interestingly, XIA also provides: (i) a X-address resolution protocol (XARP), which is similar to the address resolution protocol in the current Internet; (ii) a X-control message protocol (XCMP), which is similar to the Internet control message protocol (ICMP); and (iii) X-host configuration protocol (XHCP), which provides network bootstrap similar to the current Internet‘s dynamic host configuration protocol (DHCP). The current router prototype was implemented using Click modular router.

The XIA socket API uses XIDs instead of port numbers and IP addresses. Nowadays, it is implemented as a user-level library using python and C/C++ programming languages. Sockets have been implemented for XDP, XSP, and XChunkP. A service that wants to interact with others need to select one of these sockets and provide the target services SIDs. Thus, if an target service moves from one host to another, its SID remains the same, while the destination HID should be transparently changed. To enable application communication, XIA proposes a NRS to enable services to resolve XIDs from human language keywords. Associations among human readable names (e.g. “my web server") and XIDs can be published at the NRS.

3.4 RINA

RINA is a clean slate architecture designed considering that networking is IPC, and only IPC, i.e. the only thing that a network must provide is a good IPC service for applications. Nowadays, there are four RINA implementations: Alba, TRIA network systems LLC, Boston university, and investigating RINA as an alternative to TCP/IP (IRATI) funded by FP7 in Europe. The Alba and Boston university implementations are Java-based, while the TRIA network systems and IRATI are c-based. However, the IRATI RINA offers support to other programming languages via a simplified wrapper and interface generator (SWIG). The IRATI‘s key RINA abstractions are: (i) distributed IPC facility (DIF), (ii) distributed application facility (DAF), (iii) resource information base (RIB),

(iv) inter-DIF directory (IDD), (v) error and flow control protocol (EFCP), (vi) common distributed application protocol (CDAP), and (vii) common application connection establishment phase (CACEP).

The DIF is a single layer abstraction that can be recursively used to create multi layer networks. All DIFs have the same interface and components, regardless of their architectural level. DIFs are populated by IPC processes, which have a common internal structure. The EFCP is implemented as an internal component of the IPC process and is based on the Delta-*t* protocol from the 70s. It is divided in two protocols: the data transfer control protocol (DTCP) and the data transfer protocol (DTP). Another internal component is the Resource Information Base (RIB). It is a virtual object database. The IPC processes communicate by exchanging operations over the RIB objects using CDAP. The protocol supports object creation, deletion, reading, writing, starting, and stopping. There is an event driven RIB daemon implemented on Linux kernel to access the RIB either locally or remotely. Objects are referred by names or IDs, although the naming/identification approach was not standardized yet.

A DAF is a set of distributed application processes (DAPs) that exchange objects. DAF communication model requires application naming, the selection of adequate supporting DIFs, the establishment of data flows (with proper QoS) inside the DIFs, the establishment of a connection with peer DAPs (using CACEP), and carrying of objects (using CDAP). In current Internet, applications are not named. In RINA, applications are named. Therefore, DIF‘s flows connect named applications, instead of addresses or ports. Also, flows are identified inside applications by port IDs. The CACEP implements application connection allowing two applications to discover peer names, as well as to authenticate each other. Application connection occurs after flow establishment on a supporting DIF. After connection establishment, communication continues using CDAP messages, which are encoded using Google protocol buffers (GPB), XML, ASN.1, or JSON.

The IDD is a DAF that contains several DAPs supported by one or more DIFs. This distributed system provides application discovery when applications are supported by different DIFs. In other words, there is no DIF that provides communication between the two applications. The problem requires the expansion of an already existent DIF (to cover both applications) or the creation of a new DIF.

While the IRATI RINA implementation is divided between user and kernel spaces in Linux/Unix, the service provided by a DIF is exposed to upper layers or distributed applications by an IPC API. The API contains 6 methods: allocate flow, data write, data read, deallocate flow, register application, and unregister application. Upper layers can use this API to access bottom layers services. The IRATI RINA also provides two shim (a small code to intercept IPC API and redirect to traditional sockets) to expose the TCP/UDP/IP and Ethernet protocols as a regular DIF to upper layers. Thus, it is possible to forward RINA messages over legacy Internet protocols. Netlink sockets, system calls, and sysFS virtual file system are some of the external software used by IRATI RINA implementation. They are accessed by a a common library called *librina*.

4 Qualitative Comparison

Table 1 provides a comparison among the aforementioned FIAs considering the limitations and design metrics presented in Section 2.

4.1 Expressiveness

Contrary to current Internet approach, where applications have no name — and must be addressed by socket addresses — all studied FI initiatives provide named-entities APIs to client applications. These names are perennial, therefore enabling DSs to establish communication without depending on mutable sockets. Unique-named-applications is a better abstraction for current DSs requirements, that the 70's sockets.

Named-service orchestration is an emerging approach that is supported by XIA, RINA, and NovaGenesis. Ideally, FIA initiatives would support the entire service life-cycle, preferably adopting self-verifying names to identify services, as it is proposed on XIA's XIDs or NovaGenesis's SVNs. NovaGenesis considers service life-cycling as a fundamental requirement, which embraces event the core protocol implementations.

Naming has an important role on linking human and machine languages. Current DSs employ port names to establish collaboration among distributed processes. Operating systems enables users to name files, directories, and

processes. They also attribute names to identify processes (PIDs), as well as IPC resources, like shared memory (SHMIDs) or message buffer (MBIDs). Network interfaces are also named. e.g. “eth0" on Linux. Certainly, new Internet architectures will support thousands of DSs and for this reason, their naming strategies must be comprehensive, secure, and flexible. They need to support natural language names (multilingual), hierarchical names (follow some naming structure) or flat names (self-verifying or public key) for all kinds of existences. XIA and NovaGenesis can accommodate these requirements. However, XIDs are restricted to XIA principals. NovaGenesis allows any kind of name to be used as a principal and proposes a name-binding service to represent relationships among named existences. This is an unique feature of NovaGenesis. All traditional networking ingredients (like forwarding, routing, retransmission, multiple access, management) can be name-based and all existences can be a principal.

Also related to naming, there are the identification and routing requirements. To identify means to determine unequivocally the communication target. To locate means to determine unequivocally the position of the communication target. NetInf, XIA, RINA, and NovaGenesis enable name-based routing, i.e. the communication target can be identified by a name that is mapped to a locator for delivery purposes. OConS and CloNe do not employ the name-based approach inside SAIL. However, identification and path definition permeate all architectural levels. For example, a DS could require to identify a certain process inside a kernel or to define a path among process for a certain content. Therefore, to support these requirements, it is desirable that FIA approaches can generalize naming, identification, and location for all existences, e.g. content, services, shared memory, paths, hosts, networks, domains, etc. Additionally, to avoid unnecessary duplication at application level, architectures should provide generic mapping systems at the FIA core, which can be transparently accessed by distributed systems. This is exactly the NovaGenesis approach, implemented by PSS, GIRS, and HTS — a distributed pub/sub name resolution service.

4.2 Mobility

Future distributed systems require consistence and continuity of service during any existence mobility. Identification decoupling from location for all levels can provide efficient and flexible mobility support. Host mobility inside a domain is supported by local rebinding of mappings in OConS, XIA, RINA, and NovaGenesis. XIA and NovaGenesis provides similar support for service mobility, i.e. rebinding of service ID to host ID. Content mobility (e.g. in-network caching) is also supported by XIA, NetInf, and NovaGenesis. DSs can ask for the closest copy of some uniquely identified content in these architectures.

4.3 Data Integrity and Provenance

Self-verifying names are supported in all proposals, including RINA, which does not limit the naming structure implemented inside a DIF. However, in SAIL SVNs are limited to NetInf. XIA principals have SVN-based IDs that provide integrity and provenance. NovaGenesis supports the same features, but not limited to principals classes. Therefore, all studied proposals are aligned to Ghodsi et al. [36] rationale to adopt self-certifying names on ICT architectures. The advantages for DSs build over this new approaches are clear: content is accessed by validated names, with data integrity and provenance of sources. Trust in content itself can be granted in a location independent way [35]. Security goes beyond connections. Sockets can be validated by their names.

4.4 Exposition, Controllability, and Elasticity

From the distributed systems point of view, communication and computing hardware resources need to be exposed to software for adequate orchestration. For example, some network interface card could be shared by several application in a certain host. By exposing this hardware to software, new data flows can be admitted and the QoS required by each flow guaranteed. The same exposition is wanted for CPUs, disks, etc. Computing hardware (e.g. core in a multicore system) exposition is in the beginning and highly required to solve some of the current drawbacks on MPI for example. FIA approaches are far from named-exposition of computing resources. SAIL supports a top down model for transport (OConS) and virtual machine/network (CloNe) resources reservation. RINA adopts a similar top down model for DIF resources reservation. XIA does not covers such issue. NovaGenesis exposes substrate resources using a proxy/gateway model. This bottom-up model can be applied to

Table 1: Comparison of how FIA approaches address current Internet and distributed systems limitations.

FI initiative	SAIL	XIA	RINA	NovaGenesis
Naming	NetInf employs flat names (public signatures and content hashing). OConS employs 128 bits names for hosts. Services have additional 16 bits, which are added at the less significant portion of the host name.	Supports natural language and flat names to current principals, i.e. domains, hosts, services, and content. XIDs have 160 bits long.	Application natural language and flat naming is supported inside DIFs and DAFs. However, it lacks on content, host, domain, and other entities naming. Application and services names are global unique. A complete naming implementation is missing.	Employs both natural language and flat names, which are related via published name bindings. NG naming can accommodate the DS requirements.
Identification	Inside NetInf, contents are requested by their names. The closest copies are routed in the reverse request path or using underlay routing. At the OConS, the INC component maps destination host names to locators. Notwithstanding, OConS supports multi-path, multi-layer, multi-domain, multi-protocol routing for CloNe and NetInf.	All principals have unique XIDs. XIA routing employs alternative paths (fallback) to route content requests to some communicating target.	The scope of entities/content identification and routing is restricted to DIFs. Hierarchical or flat routing can be implemented. Routes are sequences of DIF process addresses. A node address at a certain DIF is considered as a node name by the underlaying DIF.	All routing and forwarding is name-based. Content requests and delivery are routed using SCNs in message header.
Name Resolution	The OpenNetInf provides two NRS implemented in Java: MDHT and Pastry. However, it is not used inside OConS, where the INC component is responsible to map OConS IDs to locators, such as IPv6 addresses. The OpenNetInf NRS can be used by DSs inside NetInf. Name resolution is duplicated at OConS and NetInf.	Provides a NRS able to map from human readable names to XIA IDs. Other services and distributed systems can take advantage of this NRS.	Provides a distributed application and process name resolution service called IDD, which is focused on RINA application search and discovery. DIFs map application names (including DIF process names) to underlying DIF processes addresses.	Name-to-name or name-to-content bindings are published/subscribed and stored on HTs. The PSS is available to services, including DS.
Data integrity and provenance	Self-verifying names are employed at NetInf, providing integrity and provenance.	XIDs are self-verifying names providing data integrity and provenance.	All the members of a layer are authenticated. Service data unit protection provides confidentiality and integrity. DIFs are containers with their own security mechanisms.	Employs self-verifying names for all entities, providing integrity and provenance. Employs integrity check not only for messages, but also for payload. Provenance is also related to authentication of publishers and established SLAs.
Mobility	NetInf decouples names from location. OConS provides a dynamic and distributed mobility manager (DDMM) component, where mobile terminal can access a number of access networks. The radio access technology (RAT) is selected on a flow basis approach. Handover decision includes available technologies, measurements per interface (throughput, delay, load), per flow QoS, operator and user preferences, etc.	Supports mobility of all entities through rebinding XIDs.	At the lowest levels, DIF process addresses are mapped to specific interfaces. Therefore, host mobility requires rebinding of process address to interface names.	Decouples identifiers (unique names in a scope) from locators (names that allows a notion of distance in a certain space) for all entities and content. Mobility support requires update of published name bindings at local domain PSS.
Resource Exposition	OConS resources are exposed to CloNe and NetInf to enable virtual networks establishment. Computing resources are orchestrated at CloNe. The support for computing hardware exposition is not clear. There is not a multi-path-on-top-of-OConS convergence layer at NetInf.	It is left to the user. The support for communication and computing hardware exposition is out of scope.	An underlaying DIF can perform flow admission control based on a list of QoS parameters passed by the IPC API. However, there is no native support for the exposition of substrate resources to DIFs. The support for computing hardware exposition is out of scope.	Substrate resources are exposed by proxy/gateway services, which represent physical resources during SLA negotiation. The support for computing hardware exposition is possible, but not implemented in current version.
Controllability and Elasticity	NetInf contributes for network functionality expressiveness, CloNe for deployment agility, and OConS for physical resources elasticity.	The relation with SDN is underexplored. Protocols are implemented using traditional approaches.	Networking functionalities can be changed at DIFs, providing the required flexibility. However, the set of networking functionalities available is limited by adopted process structure.	Controller-as-a-service with dynamic contracting of new hardware via representative services, i.e. resource proxies. New software-implemented networking functionalities can be deployed as new services, defining a dynamic stack.

ICT resources, including CPUs, etc.

5 Conclusion

This article put in evidence that the development of a new architecture for the Internet faces many challenges in the landscape of distributed systems' technologies. However, the research in DSs and new Internet architectures advances with less synergy than one could expect. Typically, the DSs research adopt the current Internet as the processes' communication stack. Thus, the FI research can be easily ossified, since it depends on contemporary DSs technologies, which in turn depends on the current Internet protocols. A vicious cycle is formed. Many FI initiatives adopt already established DSs solutions, which in general are tightly bound to TCP/IP. Thus, it is very difficult to advance with clean-slate FI approaches without breaking this technological cycle.

A possible solution to this *status quo* can be to develop a convergent information architecture that can explore the synergies between both fronts, eliminating unnecessary overlappings, and integrating adopted ingredients towards an unique design. Such an effort is feasible, since FIA and future DS design have several common challenges, that are recurrent in both domains. This is the aim of the NovaGenesis architecture — to address simultaneously the recurring challenges in DSs and FIA. To this end, NovaGenesis adopts a self-similar approach where the same design ingredients are employed at both levels. Considering all issues analyzed on this paper, we contend that NovaGenesis provides the best combination of ingredients to meet future DSs requirements, addressing them broadly, instead of adopting punctual solutions (that possible require functionality duplication) or application level implementations. As its main scientific contribution, NovaGenesis offers the PIaaS concept, where protocols are on-the-fly assembled and disassembled in order to compose a semantical communication between endpoints.

Finally, this article presented a theoretical view of FIA and DS synergies and challenges, presenting then the concepts of NovaGenesis. Future work includes the conclusion of the first NovaGenesis prototype and its

performance, functionality and scalability evaluations. In this way, besides a qualitative comparison with related work, we intend to demonstrate NovaGenesis features in a quantitative fashion by both running and comparing it with other FIA approaches.

Acknowledgements

Prof. Antonio and NovaGenesis design and implementation were supported by the Instituto Nacional de Telecomunicações (Inatel). Prof. Rodrigo was partially funded by the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico). Dhananjay was supported by Hankuk University of Foreign Studies research fund. In addition, this work was partially supported by Finep/Funttel Grant No. 01.14.0231.00, under the Radiocommunication Reference Center (Centro de Referência em Radicomunicações- CRR).

References

[1] TANENBAUM, A. S. *Operating systems: design and implementation*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1987.

[2] ZúQUETE, A.; FRADE, C. A new location layer for the tcp/ip protocol stack. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 42, n. 2, p. 16–27, mar. 2012. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2185376.2185379>.

[3] LOWE, H. Internet/osi application migration/portability. *StandardView*, ACM, New York, NY, USA, v. 2, n. 1, p. 46–49, mar. 1994. ISSN 1067-9936. Disponível em: <http://doi.acm.org/10.1145/224145.224152>.

[4] FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, ACM, New York, NY, USA, v. 2, n. 2, p. 115–150, maio 2002. ISSN 1533-5399. Disponível em: <http://doi.acm.org/10.1145/514183.514185>.

[5] CAROFIGLIO, G.; MUSCARIELLO, L. On the impact of tcp and per-flow scheduling on internet performance. *IEEE/ACM Trans. Netw.*, IEEE Press, Piscataway, NJ, USA, v. 20, n. 2, p. 620–633, abr. 2012. ISSN 1063-6692. Disponível em: <http://dx.doi.org/10.1109/TNET.2011.2164553>.

[6] CARPENTER, B. E. Ip addresses considered harmful. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 2, p. 65–69, abr. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2602204.2602215>.

[7] REN, Y. et al. Protocols for wide-area data-intensive applications: Design and performance issues. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012. (SC '12), p. 34:1–34:11. Disponível em: <http://dl.acm.org/citation.cfm?id=2388996.2389043>.

[8] COULOURIS, G. et al. *Distributed Systems: Concepts and Design*. 5th. ed. USA: Addison-Wesley Publishing Company, 2011.

[9] AL-JAROODI, J.; MOHAMED, N. Review: Service-oriented middleware: A survey. *J. Netw. Comput. Appl.*, Academic Press Ltd., London, UK, UK, v. 35, n. 1, p. 211–220, jan. 2012. ISSN 1084-8045. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2011.07.013>.

[10] TOOSI, A. N.; CALHEIROS, R. N.; BUYYA, R. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 47, n. 1, p. 7:1–7:47, maio 2014. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2593512>.

[11] TYSON, G. et al. A survey of mobility in information-centric networks. *Commun. ACM*, ACM, New York, NY, USA, v. 56, n. 12, p. 90–98, dez. 2013. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2500501>.

[12] DOBRE, C.; XHAFA, F. Parallel programming paradigms and frameworks in big data era. *Int. J. Parallel Program.*, Kluwer Academic Publishers, Norwell, MA, USA, v. 42, n. 5, p. 710–738, out. 2014. ISSN 0885-7458. Disponível em: <http://dx.doi.org/10.1007/s10766-013-0272-7>.

[13] ZHANG, L. et al. Named data networking. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 3, p. 66–73, jul. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2656877.2656887>.

[14] NORDSTRöM, E. et al. Serval: An end-host stack for service-centric networking. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2012. (NSDI'12), p. 7–7. Disponível em: <http://dl.acm.org/citation.cfm?id=2228298.2228308>.

[15] ALBERTI, A. M. A conceptual-driven survey on future internet requirements, technologies, and challenges. *Journal of the Brazilian Computer Society*, Springer London, v. 19, n. 3, p. 291–311, 2013. ISSN 0104-6500.

[16] ALBERTI, A. M. et al. Internet of information and services (iois): a conceptual integrative architecture for the future internet. In: *Proceedings of the 7th International Conference on Future Internet Technologies*. New York, NY, USA: ACM, 2012. (CFI '12), p. 45–45.

[17] PARTRIDGE, C. Helping a future internet architecture mature. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 1, p. 50–52, dez. 2013. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2567561.2567570>.

[18] IQBAL, H.; ZNATI, T. Overcoming failures: Fault-tolerance and logical centralization in clean-slate network management. In: *INFOCOM, 2010 Proceedings IEEE*. [S.l.: s.n.], 2010. p. 1 –5. ISSN 0743-166X.

[19] PAUL, S.; PAN, J.; JAIN, R. Architectures for the future networks and the next generation internet: A survey. *Comput. Commun.*, Butterworth-Heinemann, Newton, MA, USA, v. 34, p. 2–42, January 2011. ISSN 0140-3664.

[20] GROUP, G. Geni design principles. *Computer*, v. 39, n. 9, p. 102 –105, sept. 2006. ISSN 0018-9162.

[21] STUCKMANN, P.; ZIMMERMANN, R. European research on future internet design. *Wireless Communications, IEEE*, v. 16, n. 5, p. 14 –22, october 2009. ISSN 1536-1284.

[22] REXFORD, J.; DOVROLIS, C. Future internet architecture: clean-slate versus evolutionary research. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 9, p. 36–40, set. 2010. ISSN 0001-0782.

[23] DAY, J. *Patterns in Network Architecture: A Return to Fundamentals*. [S.l.: s.n.], 2008.

[24] SALTZER, J. *Name Binding in Computer Systems*. [S.l.: s.n.], 1977.

[25] SALTZER, J. *On the Naming and Binding of Network Destinations*. 1993. IETF Network Working Group RFC: 1498.

[26] SHOCH, J. F. Inter-Network Naming, Addressing and Routing. In: *17th IEEE Conference on Computer Communication Networks*. [S.l.: s.n.], 1978.

[27] HAUZEUR, B. M. A Model for Naming, Addressing and Routing. *ACM Trans. Inf. Syst.*, v. 4, n. 4, p. 293–311, 1986. ISSN 1046-8188.

[28] CHUN, W.; LEE, T.-H.; CHOI, T. Yanail: yet another definition on names, addresses, identifiers, and locators. In: *Proceedings of the 6th International Conference on Future Internet Technologies*. New York, NY, USA: ACM, 2011. (CFI '11), p. 8–12.

[29] DANNEWITZ, C. et al. Network of information (netinf) - an information-centric networking architecture. *Comput. Commun.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 36, n. 7, p. 721–735, abr. 2013. ISSN 0140-3664. Disponível em: <http://dx.doi.org/10.1016/j.comcom.2013.01.009>.

[30] WOUNGANG, I. et al. Mr-chord: Improved chord lookup performance in structured mobile p2p networks. *Systems Journal, IEEE*, v. 9, n. 3, p. 743–751, Sept 2015. ISSN 1932-8184.

[31] MEDRANO-CHAVEZ, A.; PEREZ-CORTES, E.; LOPEZ-GUERRERO, M. On the effect of peer online times on the lookup service of chord and kademlia p2p systems. In: *Communications (LATINCOM), 2013 IEEE Latin-America Conference on*. [S.l.: s.n.], 2013. p. 1–6.

[32] CONTI, J. P. The internet of things. *Communications Engineer*, Vol 4, 2006., 2006.

[33] PARDO-CASTELLOTE, G. Omg data-distribution service: Architectural overview. In: *Proceedings of the 2003 IEEE Conference on Military Communications - Volume I*. Washington, DC, USA: IEEE Computer Society, 2003. (MILCOM'03), p. 242–247. Disponível em: <http://dl.acm.org/citation.cfm?id=1950503.1950555>.

[34] KENT, S.; SEO, K. *Security Architecture for the Internet Protocol*. [S.l.], 2005.

[35] JACOBSON, V. et al. Networking named content. In: *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. New York, NY, USA: ACM, 2009. (CoNEXT '09), p. 1–12.

[36] GHODSI, A. et al. Naming in content-oriented architectures. In: *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*. New York, NY, USA: ACM, 2011. (ICN '11), p. 1–6. Disponível em: <http://doi.acm.org/10.1145/2018584.2018586>.

[37] FERNANDES, R.; PINGALI, K.; STODGHILL, P. Mobile mpi programs in computational grids. In: *PPoPP '06: Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*. New York, NY, USA: ACM Press, 2006. p. 22–31.

[38] PERKINS, C. Mobile networking through mobile ip. *Internet Computing, IEEE*, v. 2, n. 1, p. 58–69, Jan 1998. ISSN 1089-7801.

[39] FARINACCI, D. et al. *Locator/ID Separation Protocol (LISP)*. Fremont, CA, USA: IETF, 2010. Work in progress (draft-ietf-lisp-07). Disponível em: <http://tools.ietf.org/html/draft-ietf-lisp-07>.

[40] PAN, J. et al. Milsa: A new evolutionary architecture for scalability, mobility, and multihoming in the future internet. *Selected Areas in Communications, IEEE Journal on*, v. 28, n. 8, p. 1344 –1362, october 2010. ISSN 0733-8716.

[41] MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

[42] ALBERTI, A. et al. A novagenesis proxy/gateway/controller for openflow software defined networks. In: *Network and Service Management (CNSM), 2014 10th International Conference on*. [S.l.: s.n.], 2014. p. 394–399.

[43] OLIVEIRA, L. de et al. Service-oriented, name-based, and software-defined spectrum sensing and dynamic resource allocation for wi-fi networks using novagenesis. In: *Telecommunications (IWT), 2015 International Workshop on*. [S.l.: s.n.], 2015. p. 1–8.

[44] AHLGREN, B. et al. Content, connectivity, and cloud: ingredients for the network of the future. *Communications Magazine, IEEE*, v. 49, n. 7, p. 62–70, July 2011. ISSN 0163-6804.

[45] HAN, D. et al. Xia: Efficient support for evolvable internetworking. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2012. (NSDI'12), p. 23–23. Disponível em: <http://dl.acm.org/citation.cfm?id=2228298.2228330>.

[46] VRIJDERS, S. et al. Prototyping the recursive internet architecture: the irati project approach. *Network, IEEE*, v. 28, n. 2, p. 20–25, March 2014. ISSN 0890-8044.

[47] ALBERTI, A. M. Searching for synergies among future internet ingredients. In: LEE, G. et al. (Ed.). *Convergence and Hybrid Information Technology*. [S.l.]: Springer Berlin Heidelberg, 2012, (Communications in Computer and Information Science, v. 310). p. 61–68.

[48] GALIS, A. et al. Softwarization of future networks and services -programmable enabled networks as next generation software defined networks. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–7.

[49] DANNEWITZ, C.; HERLICH, M.; KARL, H. Opennetinf - prototyping an information-centric network architecture. In: *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*. [S.l.: s.n.], 2012. p. 1061–1069.

[50] VARDA, K. *Protocol Buffers: Google's Data Interchange Format*. [S.l.], 2008.

[51] KHARE, R.; RIFKIN, A. Xml: a door to automated web applications. *Internet Computing, IEEE*, v. 1, n. 4, p. 78–87, Jul 1997. ISSN 1089-7801.

[52] VANBRABANT, R. *Google Guice: Agile Lightweight Dependency Injection Framework (Firstpress)*. [S.l.]: APress, 2008.

[53] PRUD'HOMMEAUX, E.; HARRIS, S.; SEABORNE, A. (Ed.). *SPARQL 1.1 Query Language*. [S.l.], 2013. Disponível em: <http://www.w3.org/TR/sparql11-query>.

[54] BRICKLEY, D.; GUHA, R. *RDF Vocabulary Description Language 1.0: RDF Schema*. [S.l.], 2004. Disponível em: <http://www.w3.org/TR/rdf-schema/>.

[55] ROWSTRON, A. I. T.; DRUSCHEL, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. London, UK, UK: Springer-Verlag, 2001. (Middleware '01), p. 329–350. Disponível em: <http://dl.acm.org/citation.cfm?id=646591.697650>.

[56] D'AMBROSIO, M. et al. MDHT: a hierarchical name resolution service for information-centric networks. In: *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. New York, NY, USA: ACM, 2011. (ICN '11), p. 7–12. Disponível em: <http://dx.doi.org/10.1145/2018584.2018587>.

[57] KEENEY, J. et al. Extending siena to support more expressive and flexible subscriptions. In: *Proceedings of the Second International Conference on Distributed Event-based Systems*. New York, NY, USA: ACM, 2008. (DEBS '08), p. 35–46. Disponível em: <http://doi.acm.org/10.1145/1385989.1385995>.

[58] PIETZUCH, P.; BACON, J. Hermes: a distributed event-based middleware architecture. In: *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*. [S.l.: s.n.], 2002. p. 611–618.

[59] MOSKOWITZ, R.; NIKANDER, P. *Host Identity Protocol (HIP) Architecture*. IETF, 2006. RFC 4423 (Informational). (Request for Comments, 4423). Disponível em: <http://www.ietf.org/rfc/rfc4423.txt>.