Converging Future Internet, "Things", and Big Data: A Specification Following NovaGenesis Model

Antonio M. Alberti[1], Eduardo S. dos Reis[2], Rodrigo da R. Righi[2], Víctor M. Muñoz[3] and Victor Chang[4]

[1]*ICT Laboratory, Instituto Nacional de Telecomunicações - INATEL, Santa Rita do Sapucaí, Minas Gerais, Brazil*

[2]*PIPCA, Universidade do Vale do Rio dos Sinos, São Leopoldo, Rio Grande do Sul, Brazil*

[3]*Universitat Autonoma de Barcelona, UAB, Barcelona, Spain*

[4]*School of Computing, Creative Tech. and Eng., Leeds Beckett University, Leeds, U.K.*

Keywords: Internet of Things, Big Data, Cloud Computing, Future Internet, NovaGenesis, FIWARE, Spark.

Abstract: The convergence of Internet of "things" (IoT) with big data platforms and cloud computing is already happening. However, the vast majority, if not all the proposals are based on the current Internet technologies. The convergence of IoT, big data and cloud in "clean slate" architectures is an unexplored topic. In this article, we discuss this convergence considering the viewpoint of a "clean slate" proposal called NovaGenesis. We specify a set of NovaGenesis services to publish sensor device's data in distributed hash tables employing self-verifying addresses and contract-based trust network formation. IoT devices capabilities and configurations are exposed to software-controllers, which control their operational parameters. The specification covers how the "things" sensed information are subscribed by a big data service and injected in *Spark* big data platform, allowing NovaGenesis services to subscribe data analytics from *Spark*. Future work include implementation of the proposed specifications and further investigation of NovaGenesis services performance and scalability.

1 INTRODUCTION

Internet of "things" (IoT) (Conti, 2006) can be defined as to connect the ordinary things to the Internet. In the last years, many initiatives appeared to converge IoT with big data. Among them there are: FIWARE (Ramparany et al., 2014), SmartSantander (Sanchez et al., 2013) and SENSEI (Presser et al., 2009). FIWARE provides a platform to integrate services via next generation service interfaces (NGSIs). SmartSantander provides a platform for smart city services integration using RESTful (Richardson and Ruby, 2007) application programming interfaces. SENSEI focus was on wireless sensor and actuator networks interoperation, creating a market of sensed information via RESTFul interfaces.

FIWARE integrates *Cosmos* big data platform services, which are based on *Hadoop* (Hu et al., 2014). SmartSantander employed a combination of *Spark* (Zaharia et al., 2010) and *Cassandra* (Lakshman and Malik, 2010). These approaches solidly depend on current Internet support, since they use RESTFul APIs. However, many IoT challenges, like security, trust, privacy, software-control, to name a few, will require more than incremental solutions or the application of models already established. New architecture models will be demanded, since the current cloud and networking technologies and their protocols are inherently limited for several expected scenarios (Pan et al., 2011). "Clean slate" architectures for Internet (and consequently for IoT) are being engineered under the banner of future Internet design.

The convergence of IoT, big data, and cloud in "clean slate" future Internet architectures is an unexplored topic and the main contribution of this paper is on specifying an architecture to integrate them. In this context, we are developing NovaGenesis (NG[1]) (Alberti et al., 2014; de Oliveira et al., 2015), a convergent information model that covers information exchanging, processing, and storage. In this paper, we address IoT, big data, cloud computing, and software-defined (McKeown et al., 2008) technologies' convergence under the perspective of NovaGenesis. In addition, we specify a set of NovaGenesis services to publish sensor device's data in distributed hash tables (DHTs) employing self-verifying names (SVNs)

[1]http://www.inatel.br/novagenesis/

Alberti, A., Reis, E., Righi, R., Muñoz, V. and Chang, V.
Converging Future Internet, "Things", and Big Data: A Specification Following NovaGenesis Model.
DOI: 10.5220/0005890600730081
In *Proceedings of the International Conference on Internet of Things and Big Data (IoTBD 2016)*, pages 73-81
ISBN: 978-989-758-183-0

and contract-based trust network formation. In other words, we specify the data path form devices to the cloud, employing NovaGenesis emerging paradigms, like SVNs and contract-based service composition. IoT devices capabilities and configurations are exposed to software-controllers, which control their operational parameters following the model in (Alberti et al., 2014). Finally, we specify how the "things" sensed information are subscribed by a big data service (BDS) and injected in *Spark* big data platform. This allows NovaGenesis services to subscribe data analytics generated by *Spark*.

The remaining of this paper is structured as follows. Section 2 provides an overview of FIWARE IoT + big data + cloud computing integration proposal. The section also covers the requirements for "clean slate" FI architectures that can improve the state-of-the-art while converging these aspects. Section 3 presents NovaGenesis and its current implementation. Section 4 specifies a set of new services for converging IoT, big data, cloud computing, and software-controllers. A sequence diagram is provided to clarify new services joint functioning. The section also provides a discussion on how these services interact one another, ranging from "things" raw data to big data analytics results. The main contribution of the paper is on integrating future Internet services with already established big data technologies and software-controlled devices and gateways, advancing IoT architectures towards SDN and SOA. Section 5 provides a discussion on how the proposed model can help on addressing some open challenges in IoT and big data integration. Section 6 finishes the paper.

2 RELATED WORK AND REQUIREMENTS

The amount of future Internet (FI), IoT, cloud computing, and big data convergent scenarios is immense. The amount of technologies that support such scenarios is also impressive. One example of initiative aimed at this convergence is the European project FIWARE (Ramparany et al., 2014). Figure 1 illustrates a simplified overview of FIWARE architectural components for IoT, which include: (i) IoT broker; (ii) backend device management (BDM); (iii) context-broker (CB); (iv) big data analysis (BDA); and (v) complex event processing (CEP). Before describing them, it is important to notice that three kinds of "things" are supported: (a) devices that are compatible with next generation service interface (NGSI) version 9/10; (b) devices that are not compatible with NGSI 9/10, however the gateways are; and (c) devices and gateways that are not compatible with NGSI-9/10. The IoT broker recovers, collects and processes information from "things" exposing devices as RESTful application programming interface (API) resources. The BDM exposes legacy technologies (standardized or proprietary) as resources to the CB via NGSI-9/10. IoT agents are instantiated to handle, configure and monitor non NGSI devices and gateways. The CB provides a publish/subscribe context broker service via NGSI-9/10 interface. Contexts can be registered, updated, queried, notified, subscribed, etc. For example, a native NGSI-9/10 device can create a context that carries the current value of the temperature in a certain room. BDA is an extended version of hadoop (Hu et al., 2014) from Telefonica (called Cosmos). Finally, CEP is an IBM generic enabler (as FIWARE names its components) to correlate real time events according to programmed rules. The data generated either by CEP or BDA is published on CB. BDA is fed by CB and processed data from CEP. Therefore, FIWARE allows near real time map/reduce operations over large amounts of data from IoT.

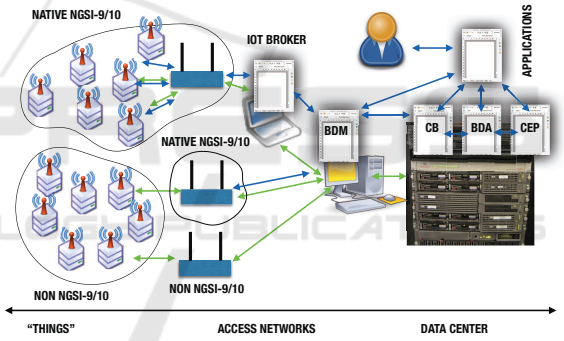

Figure 1: FIWARE scenario for converging future Internet, Internet of "things", big data, and cloud computing.

Considering the perspective of *clean slate* architectures for the future Internet (Pan et al., 2011), the convergence provided by the FIWARE project — that is supported by current Internet and web technologies — could be improved in the following aspects[2]:

Security - The current Internet approach of using natural language names (NLNs) to name uniform resource locators (URLs), hosts (IP addresses) and transport layer ports (in sockets) is human-friendly, but the intrinsic binding to physical world entities depends on the correct and unambiguous understanding of these names. Information centric networking (ICN) proposes self-verifying names (SVNs) as an alternative to NLNs (Ahlgren et al., 2012), so making possible to check the relationship between the named

[2]Many other aspects can be considered. However, due to limited space we decided to concentrate on these three.

entity and the name at any time. Furthermore, Ghodsi et al. (Ghodsi et al., 2011) argue that SVNs have better security, scalability, and flexibility than NLNs. SVNs can also support provenance, non repudiation, and integrity of IoT data and its sources. FIWARE architecture is based on the current Internet/Web design and therefore do not support SVNs.

Contract-based Composition - Web services can support contract-based services composition, e.g. using constrained application protocol (CoAP) observable resources. This enables services to form a trust networks, increasing security and privacy. Who would like to have your IoT services publishing private data to low trust services you do not know? The RESTFul interfaces (e.g. NGSI-9/10) are dynamic and allow a service to establish service level agreements (SLAs) with other instantiated services. However, contract-based operation is typically not obligatory. For instance, in FIWARE generic enablers are not required to establish SLAs one another. SLAs are employed to deal with cloud infrastructure.

Software-Defined Control - The role of software is increasing in emerging information and communication technologies (ICT) architectures. Software-defined networking (SDN) consists on decoupling control and data planes in networking equipment (McKeown et al., 2008). Unfortunately, current open SDN standards are concerned only with traffic forwarding, e.g. OpenFlow. There exists a huge potential behind software-controlling IoT and cloud devices/infrastructure. Controllers can be extended to deal with many other functions, like proxying, fire-walling, interconnection, etc. FIWARE’s backend device management is an example of software-controlling IoT devices. These software controllers can be exposed as services to other applications. Another example is the FIWARE’s protocol translation software that can be software-controlled. In short, FIWARE project already perceived the potential of software-controlling the IoT/cloud devices. However, what is missing is contract-based dynamic composition with exposed controllers-as-a-service (CaaS).

FIWARE and SmartSantander meet the software-defined control requirement, but lack on supporting SLAs among their services as well as on supporting SVNs and its name resolution. SENSEI does not meet any of these requirements. (Pan et al., 2011) surveys many future Internet architectures, including “clean slate” ones. Some of them individually support these requirements, but they are not focused on IoT.

3 NovaGenesis

NovaGenesis (NG) project started in 2008 with this question in mind: imagine there is no Internet architecture right now, how could we design it using the best contemporary technologies? However, we realized that emerging architectures should embrace not only networking aspects, but also computing, storage, and visualization. Thus, NG aims for redesigning both communication mechanisms and protocols for enabling a program running anywhere to address messages to programs anywhere else with acceptable performance and portability levels. Also, it looks for the convergence of computing and communications, merging technologies like cloud and mobile computing, Internet of things (IoT), service frameworks, software-defined networking (SDN), network function virtualization (NFV) and distributed systems (Alberti, 2012; Alberti, 2013).

3.1 Foundational Concepts

Figure 2 illustrates key ingredients we tried to converge in NG. We started from naming, name resolution, and life-cycling of contents and services. Thereafter, we included exposition, control and gateway for “things”. The idea was to create a generic framework that could deal with naming, life-cycling, persistent identification, location-independency, joint content, services, and “things” coordination (orchestration).

3.1.1 Existences

The physical world in the Earth is full of physical entities, *e.g.* cars, houses, etc. Furthermore, the human mind is capable of creating virtual realities, full of virtual entities, *e.g.* computer programs, files. Adopting these definitions, a convergent information architecture can be seen as the combination of physical and virtual existences aimed at dealing with information in a universal scale. An “individual” existence is anything that can be classified as independent or separated from others.

3.1.2 Binding Natural and Hash-Function-Generated Names

Names are symbols 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. However, there are names that are almost randomly generated, having weak semantics, e.g. the plate of your car. Consider the binary word obtained at the output of a hashing algorithm. This binary word (also called hash code) can

be used as a name a self-verifying name (SVN). In this case, 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 entity being named (e.g. entities' immutable attributes). In the first case, the name is said to be self-certifiable, because at any time the existence's binary words can be hashed again to get exactly the same name. In the second case, the perennial physical existence attributes can be digitalized again to certificate the name.

IoT means that any "thing" will be part of the Internet, therefore architectures should accommodate/recognize "things" names. Our approach employs an unlimited number of namespaces, that are linked by means of name bindings. A name binding is a mapping among two or more names. A name binding (NB) can relate a name to an object or a name to other names. NovaGenesis allows any naming structure and generalized name resolution as a service.

3.1.3 Names as Identifiers or Locators

There is no novelty on using names as identifiers or locators. This is intrinsic to ICT technologies. However, the adoption of SCNs as identifiers is more recent. NovaGenesis borrowed this idea from other FIAs, specially NetInf (Dannewitz, 2009) and XIA (Han et al., 2012). A locator should provide a notion of distance among existences in some space. As one can expect, it is not possible to derive such distance notion from SVNs — they are flat. Then, NG provides such distance notion by using SVN-bindings.

3.1.4 Contents, Services, and Contracts

A **content** can be defined as a piece of information, composed by chunks of binary data. For example, a textual file. A **service** is a data (or information) processor. Thus, a textual file can be the input or the output of a word processor service. In addition, a networking information processor, such as a frame forwarder, is a service like any other, since it receives information at the input, adds some control data, and forwards the processed information as an output. In this context, a process that is instantiated in an operating system provides in essence information processing, which in turn can be defined as a service. Therefore, in this paper we propose the following definition: a service is an existence aimed at processing, exchanging, or storing information. Services negotiate their job using contracts, which are contents that describe service requirements, clauses to be respected, evaluation, and finalization criteria, among other data. All the information required to describe

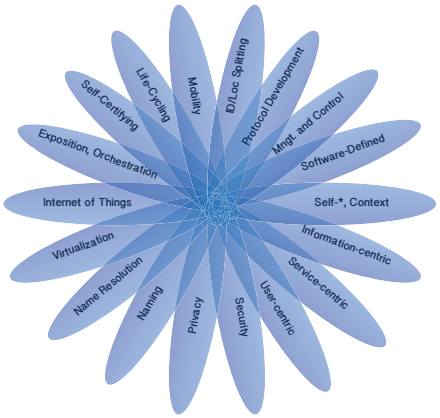

Figure 2: Key ingredients for Future Internet.

and regulate a service offer can be included in a contract. In service-oriented architecture (SOA) a contract is usually called service-level agreement (SLA).

3.1.5 Distributed Publishing and Subscribing of Name-bindings and Content

Names and NBs are central in NG. Thus, an important design choice was to decide how they are going to be stored and exchanged. Instead of a traditional communication model, such as a "receiver accepts all model", it was selected a loosely coupled, asynchronous, "receiver select" model — a pub/sub approach. In summary, services can publish name bindings or content to other services. For example, the binding of an operating system name to a host name can be published by a service that represents the OS. Another OS representative service can subscribe this binding (assuming it is public) to discover that there is a name relation between those two existences.

3.1.6 Name-based Life-cycling

The life-cycling of substrate resources, services, and content (including names) is populated by a dense entanglement of existential relationships. Operating systems inhabit hosts. Services are instantiated, combined in runtime, and removed from operating systems. Contents are created, exchanged, processed, and removed from services. To deal with this hierarchical life-cycling requirement, name bindings are categorized. There are NBs exclusive for: (i) physical world entities; (ii) software entities; (iii) contents; (iv) other names, e.g. a process ID in a Linux OS. There are also NBs to link names among physical-virtual, virtual-physical, etc.

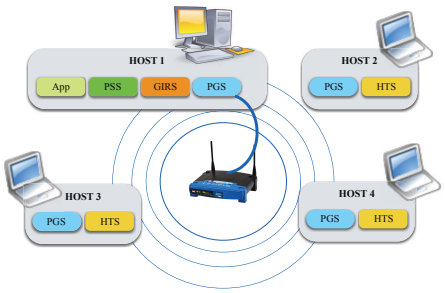

Figure 3: NovaGenesis current prototype.

3.2 Proof-of-concept Implementation

In 2012, we implemented a C++ prototype of NovaGenesis (NG) to run at user space of Linux Kernel as illustrated on Figure 3. This 40k line codes prototype already runs without TCP/IP, using a raw socket directly over Ethernet or Wi-Fi. In the figure, Host 1 is connected by Ethernet, while the remaining hosts use Wi-Fi. A proxy/gateway service (PGS) encapsulates NG messages over link layer technologies.

NovaGenesis web of names is build of distributed stored name bindings. NBs are published and subscribed by services. This is the most innovative aspect of NovaGenesis — a distributed, highly scalable, publish/subscribe, name binding and content storage service. NBs and content are published by any services through sending a message to an instance of a publish/subscribe service (PSS). The PSS stores which services are pre-authorized to access the published data. Also, it records a time to live (TTL) for any published data. There is also a notification functionality that enables PSS to inform other services about new publications or NB updating. However, the PSS does not store the data published. It forwards them to a generic indirection resolution service (GIRS), which selects a hash table service (HTS) to store it. Thus, published NBs and content are forwarded by the GIRS to an HTS instance, where they are stored behind a hash table data structure.

User services (or applications — App) can expose its NBs and content to other services via PSS. Thus, other services can subscribe a services’ NBs and content. The PSS does the rendezvous between publishers and subscribers, 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 NBs in local data structures, and routing information based on them.

GIRS and HTS instances form a distributed hash table (DHT). This DHT was designed from the scratch to support named-based and contract-oriented operation. Current DHTs do not employ self-certifying names to identify the DHT nodes. In addition, they relay on TCP/IP stack for communication. Our GIRS/HTS approach relays on NovaGenesis services to forward and route the data among DHT instances. This explains why we decided on a “clean slate” DHT design.

4 EXTENDING NG FOR IoT/BIG DATA

In this section, we specify an extension to NovaGenesis services that can provide similar support for converging IoT, FI, big data and cloud as FIWARE provides. However, our proposal meets the requirements presented in Section 2 to advance convergence towards more secure, trustable, contract-based, software-controlled architectures. Figure 4 illustrates how these new services interact one other by using NovaGenesis PSS/GIRS/HTS, which can be seen as a distributed name resolution system (NRS) with cache capabilities. All these services use a publish/subscribe (pub/sub) application programming interface (API), which has five primitives: (i) publishes a NB (and a content, if any); (ii) subscribes a NB (and content, if any); (iii) notifies peer services about NB (and content) published; (iv) revokes a publication; and (v) delivers name bindings and related contents. This interface is used alternatively to the NGSI-9/10 adopted on FI-WARE. In terms of components, the proxy/gateway/controller service (PGCS) is similar to the IoT broker, even tough it includes some of the FI-WARE’s BDM functions, like resource exposition, monitor, etc. NovaGenesis big data service (BDM) role is different from FI-WARE’s BDA. It provides an application level gateway to the big data solutions, such as *Spark*, while the BDA implements the hadoop distributed system. In addition, NG relays on a new name resolution system (NRS), while FI-WARE is

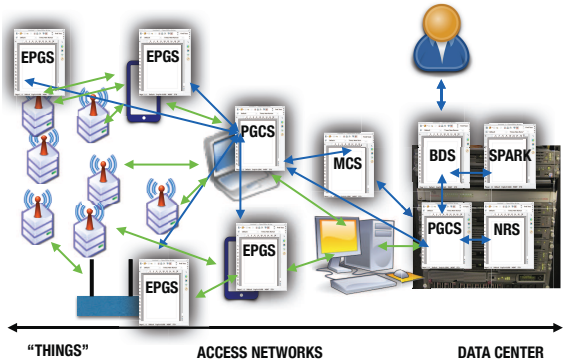

Figure 4: New NovaGenesis services for converging future Internet, Internet of “things”, big data, and cloud.

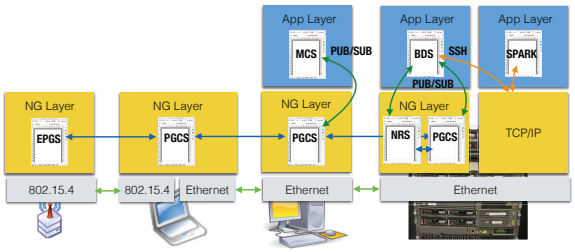

Figure 5: Layer stack for NG and *Spark* big data plus IoT.

based on DNS.

Figure 5 illustrates the protocol stack behind our approach. The NG layer employs a blank page messaging service implemented using SVNs. NG messages are encapsulated over IEEE 802.15.4 (to be implemented) or Ethernet/Wi-Fi (already implemented). The NovaGenesis pub/sub API can be seen as a service access point (SAP) between NG Layer and NG application layer. Observe that the original PGS presented in Figure 3 is modified not only to run embedded (EPGS), but also to include software-control capabilities (PGCS). To the best of our knowledge, the EPGS is the first future Internet gateway that: (i) employs SVNs, instead of "human readable" addresses; (ii) exposes hardware resources to services using a contract-based approach; and (iii) receives control commands from a software-controller (PGCS) using named-data paradigm.

4.1 Embedded Proxy/Gateway Service

This service is a minimal NG implementation to run embedded. It generates nodes' self-verifiable names as well as a descriptor of the hardware features and functionalities, storing them in a local hash table (HT) component. Also, it sends a periodic "hello" message that exposes its names as well as its MAC address (proxy function)[3] to possible peers (Figure 6 illustrates this procedure). The "hello" step is shown as transaction *a* in this figure. Since a PGCS receives this "hello", it answers back with its names and PSS's names, enabling the EPGS to publish data on the NG NRS (Transaction *b*). Then, the EPGS publishes its name bindings, descriptors, and a service offer to the NRS via the discovered PGCS (*c*). The NRS notifies the PGCS (*d*), which subscribes the service offer (*f*,*g*). The service offer exposes device features, names, and capabilities to the PGCS. The PGCS publishes a service contract acceptance object (*h*) to the NRS. The NRS notifies the EPGS about the acceptance (*i*), which subscribes it (not shown in figure).

[3]After contracting a PGCS the hello message can be suppressed to save energy

The EPGS is able to encapsulate NG messages over a link layer technology (gateway function) that is compatible with a PGCS (IEEE 802.15.4 as shown in Figure 5). The EPGS publishes its status, configuration, and raw data (measures) to the NRS (*p*), notifying other services interested in the raw data (BDS) or status/configuration (PGCS).

4.2 Proxy/Gateway/Controller Service

The PGCS has gateway, proxy, and controller functionalities. Similarly to the EPGS, it generates node's SVNs and descriptors and store them in a local HT. It also sends a periodic "hello" message with its names and MAC address[3] to other PGCS (*a*,*b* in Figure 6). The PGCS is an IoT gateway for one or more sensors and/or actuators nodes. The PGCS has the responsibility to disclosure PSS names, so the EPGS(es) can use NRS API (*b*). The PGCS is able to encapsulate message to the its contracted EPGS(es), using their link layer technologies, e.g. IEEE 802.15.4. Finally, the PGCS acts as a software-controller of the EPGS's represented devices, i.e. the PGCS can publish control messages to the EPGSes in its trust network, changing their configurations. For example, a PGCS can publish a control message ordering a controlled device to change its IEEE 802.15.4 channel.

4.3 Management and Control Service

This service is responsible to manage and control the infrastructure and it is implemented as an application that uses NG pub/sub API. It exposes its functionalities, which can cover the classical five management areas: fault, configuration, accounting, performance, security (FCAPS). It maintains a trust network (via SLAs) with many PGCSes. This model enables to propagate policies and rules in a distributed way to many software-controllers, keeping consistency, since configurations/rules/policies are all published in the NRS using SVNs. It addresses the load balancing and controller consistency problems of OpenFlow. In summary, the MCS performs gateway/proxy/controller management and control. Depending on the amount of ctrl. and manag. operations, this service can be fragmented.

4.4 Big Data Service

This service is a gateway to interoperate to *Spark* (Zaharia et al., 2010) — a popular big data implementation. The BDS exposes big data platforms as a service to other NG services/applications via pub/sub API.

Its aim is similar to FIWARE’s Cosmos generic enabler — even though Cosmos is based on *Hadoop* (Hu et al., 2014). NovaGenesis is a pub/sub environment, therefore the BDS will need to translate this communication model to request/reply, which is the typical model to access current big data implementations. Also, NG is implemented in C++ language, which makes interoperation with big data approaches complex, since typically there is no native support for injection/querying in C/C++. In this context, to interoperate NG with *Spark* the BDS will keep a secure socket shell (SSH) connection to the *Spark* console (Figure 5) using TCP/IP. Using this interface, the BDS will be able to send *Hive* structured query language (SQL) queries to the *Spark* cloud. Also, the BDS will be able to: (i) inject data subscribed at NovaGenesis side on *Spark*; and (ii) publish the obtained results on NovaGenesis NRS. Thus, data series from NovaGenesis distributed hash service will be injected into *Spark*, analyzed, and retrieved via BDS.

In Figure 6, the BDS searches for PGCSes that can provide useful data for analytics. It subscribes relevant NLN (e.g. “PGCS”, “gateway”, etc) from local NRS (transaction *i*). The NRS returns known PGCSes (*j*). The BDS publishes a service offer to a PGCS (*k*) that can provided the raw data for big data analysis. The PGCS subscribes the service offer (*l*,*m*). After analyzing the offer, the PGCS publishes a service offer acceptance object to the BDS (*n*), which is notified by the NRS. The BDS subscribes the acceptance object (*o*). Finally, the raw data provided by the EPGS is published to the BDS (*p*), which subscribes it (*q*). The subscribed raw data is injected by BDS on *Spark* via SSH. The BDS can now perform queries on *Spark* (*s*) and publish resulting analytics on NG NRS (*t*).

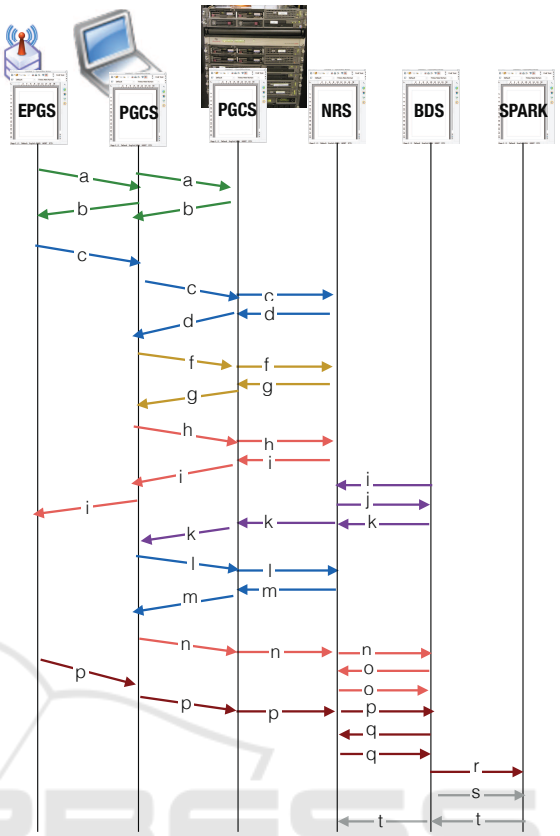

Figure 6: Sequence diagram for NG and *Spark* big data, including EPGS and laptop PGCS “hello” (*a*,*b*), exposition and service offering (*c*,*d*,*f*,*g*), service contracting (*h*) and acceptance (*i*). A similar discovery, negotiation, and contract establishment procedure is performed by the laptop PGCS and BDS pair (*i*,*j*,*k*,*l*,*m*,*n*,*o*). Raw data is published by the EPGS (*p*), notified and subscribed by NRS (*q*), and injected on *Spark* (*r*). Results are published by BDS on NG NRS (*t*).

5 DISCUSSION

Since there will be thousands of devices to be connected (in ideal situations), it is a huge task is to monitor, control and manage all network traffics and devices (mobiles, routers, sensors etc). This is highly important for all IoT and BD architectures. Factors affecting such performance will include latency, size of data and network speed (Chang and Wills, 2015). The proposed N:1 device control/management model based on software-controllers (PGCSes) is highly scalable, since more PGCSes can be elastically allocated when required. The proper selection of services’ hosts can decrease latency.

Current security and privacy methods are lagging behind sophisticated hacking techniques and just one single solution is far not enough. A multi-layered security solution can reduce the risk of unauthorized access and the use of back-end non-SQL databases should be implemented in core servers/clusters for IoT and BD architectures (Chang et al., 2016). NovaGenesis self-verifying names, pub/sub, and contract-based operation enhances security, privacy, and trust. Name resolution is very important to provide a multi-layered security solution.

Since a lot of data can be generated, smart means to recover data at any instance with advanced techniques to meet performance, accuracy and reliability are required. All sites should have the facility to perform multi-purpose and multi-site recovery to ensure business continuity (Chang, 2015). The distributed nature of NG model can help on maintaining coherence in case of disaster. Name resolution can be explored to find out alternatives to access some data or network. Name rebind is an important tool.

6 CONCLUSION

This work has proposed the concept and reported the specification of a model to integrate big data, cloud computing, IoT device-control-as-a-service, and IoT services. The proposed model is based on NovaGenesis, a "clean slate" FI proposal. A big data service was proposed to interoperate NG name resolution service (based on pub/sub and distributed hash tables) with *Spark* big data. The proposed specification also introduces embedded proxy/gateway services that encapsulate IoT devices traffic towards NG pub/sub, as well as represent IoT device's in the service tier. Software-defined IoT devices are controlled by proxy/gateway/controller services (Alberti et al., 2014). A domain level management and control service (MCS) was also specified to implement in a logical centralized way the classical areas of management and coordinate PGCSes activities. The architecture advances state-of-the-art by converging data, control, and management planes for FI with big data and IoT.

Future work includes: (i) implementation and performance evaluation of the convergent architecture specified in this paper; (ii) elasticity and scalability tests of the proposed control/management model including cloud computing resources; (iii) evaluated latency and efficiency of proposed control/management model; (iv) to monitor and evaluate service reputation; (v) explore security advantages of proposed model; (vi) implement and evaluate NG benefits for disaster recovery scenarios; and (vii) evaluate NG integration to other big data platforms besides *Spark*.

ACKNOWLEDGEMENT

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) project of the National Institute of Telecommunications (Instituto Nacional de Telecomunicações), Brazil.

REFERENCES

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36.

Alberti, A. (2013). A conceptual-driven survey on future internet requirements, technologies, and challenges. *Journal of the Brazilian Computer Society*, 19(3):291–311.

Alberti, A., de O Fernandes, V., Casaroli, M., de Oliveira, L., Pedroso, F., and Singh, D. (2014). A novagenesis proxy/gateway/controller for openflow software defined networks. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 394–399.

Alberti, A. M. (2012). Searching for synergies among future internet ingredients. In Lee, G., Howard, D., Slezak, D., and Hong, Y., editors, *Convergence and Hybrid Information Technology*, volume 310 of *Communications in Computer and Information Science*, pages 61–68. Springer Berlin Heidelberg.

Chang, V. (2015). Towards a big data system disaster recovery in a private cloud. *Ad Hoc Networks*, 35:65 – 82. Special Issue on Big Data Inspired Data Sensing, Processing and Networking Technologies.

Chang, V., Kuo, Y.-H., and Ramachandran, M. (2016). Cloud computing adoption framework: A security framework for business clouds. *Future Generation Computer Systems*, 57:24 – 41.

Chang, V. and Wills, G. (2015). A model to compare cloud and non-cloud storage of big data. *Future Generation Computer Systems*, pages –.

Conti, J. P. (2006). The internet of things. *Communications Engineer*, Vol 4, 2006.

Dannewitz, C. (2009). Netinf: An information-centric design for the future internet. In *Proc. 3rd GI ITG KuVS Workshop on The Future Internet*.

de Oliveira, L., Alberti, A., Favoreto Casaroli, M., Raimundo-Neto, E., Feliciano da Costa, I., and Cerqueira Sodre, A. (2015). 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*, pages 1–8.

Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, P., and Shenker, S. (2011). Naming in content-oriented architectures. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 1–6, New York, NY, USA. ACM.

Han, D., Anand, A., Dogar, F., Li, B., Lim, H., Machado, M., Mukundan, A., Wu, W., Akella, A., Andersen, D. G., Byers, J. W., Seshan, S., and Steenkiste, P. (2012). XIA: Efficient support for evolvable internetworking. In *Proc. 9th USENIX NSDI*, San Jose, CA.

Hu, H., Wen, Y., Chua, T.-S., and Li, X. (2014). Toward scalable systems for big data analytics: A technology tutorial. *Access, IEEE*, 2:652–687.

Lakshman, A. and Malik, P. (2010). Cassandra: A decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2):35–40.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Pan, J., Paul, S., and Jain, R. (2011). A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26 –36.

Presser, M., Barnaghi, P., Eurich, M., and Villalonga, C. (2009). The sensei project: integrating the physical world with the digital world of the network of the future. *Communications Magazine, IEEE*, 47(4):1 –4.

Ramparany, F., Galan Marquez, F., Soriano, J., and Elsaleh, T. (2014). Handling smart environment devices, data and services at the semantic level with the fi-ware core platform. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 14–20.

Richardson, L. and Ruby, S. (2007). *Restful Web Services*. O’Reilly, first edition.

Sanchez, L., Gutierrez, V., Galache, J., Sotres, P., Santana, J., Casanueva, J., and Munoz, L. (2013). Smartsantander: Experimentation and service provision in the smart city. In *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*, pages 1–6.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud’10, pages 10–10, Berkeley, CA, USA. USENIX Association.