Up in the clouds: A Taxonomical Analysis of Network Management Functionalities from a Network as a Service Perspective

Roberto Riggio[†], Slawomir Kuklinski[‡], Tinku Rasheed[†], David Andreatta[§], David Tacconi[§] and Fabio Antonelli[†]
[†]CREATE-NET Research, Trento, Italy ; *Email: first.secondname@create-net.org*
[‡]Orange Labs, Warsaw, Poland ; *Email: slawomir.kuklinski@telekomunikacja.pl*
[§]Futur3 Srl., Trento, Italy ; *Email: first.secondname@futur3.it*

Abstract—**The externalization of network control and maintenance tasks, enabled by the emerging Networking as a Service (NaaS) paradigm, is an appealing opportunity for large Telcos to expand their network management offering to SMEs. In this paper we propose an architecture for network control and management capable of efficiently supporting the NaaS paradigm. Our methodology is based on a set of orthogonal requirements that forms the platform to assess the admissibility of different network management functions in the remote management plane. Such assessments are realized based on the taxonomical evaluation of the management functions classified according to the requirements they impose on the remote management plane in terms of security and robustness. Experimental results obtained from a proof-of-concept implementation deployed over a large scale testbed composed of about 500 WiFi Access Points shows that the signaling traffic for a practical implementation supporting not trivial autonomic control loops is lower that 20 bytes/s for each node. Moreover, the latency to execute administrative actions is in most cases in the order of 70 ms.**

Index Terms—**network management, monitoring, networking as a service, mesh networks, proof-of-concept, prototyping**

I. Introduction

Managing internal network infrastructures can be a complex, labor–intensive and expensive task, especially for enterprises whose main business lies outside, or is only marginally related with, the ICT domain. Such pitfall led all but a few very large Telcos/Enterprises to move away from sophisticated and/or automated network management tools in favor of simpler solutions requiring frequent manual intervention, often using custom tools developed in–house. Similar considerations prompted SMEs to move storage, email, and in some case even computing infrastructure to the "cloud" following the Infrastructure/Platform/Software as a Service model.

The externalization of the network deployment and maintenance tasks, that is enabled by the emerging Networking as a Service (NaaS) paradigm, is an extremely appealing opportunity for large Telcos to expand their network management offering to mid/small enterprises [1]. In such a context, a cloud-centric approach can lower the entrance barrier to a Telco's network management portfolio by both allowing management software to be developed and maintained in a centralized fashion and by enabling effective and efficient infrastructure sharing among multiple customers.

In this scenario, the Telcos benefits from the amortization of hardware, software and human costs across their entire customer base while customers have the flexibility of paying only for what they effectively use. Such consolidation of functionalities has also technical advantages in terms of advanced adaptation and learning algorithms that can take advantage of a knowledge base collected from a multitude of nodes. In particular, the NaaS paradigm will:

- enable use of thin–data plane devices running a bare minimum hardware and requiring a separate control and management plane to operate. This reduces the energy footprint of the devices as well as their cost;
- reduce the cost to deploy the network (CAPEX), in that the cloud will be used for its initial configuration, *and* the cost to keep it operational (OPEX), in that it will be continuously monitored optimized from the cloud.
- pave way to innovative applications of cloud–computing in the management of Future Internet infrastructures.

However, in order to turn NaaS into a commodity, two sets of challenges need to be addressed. The former is related with the intrinsic nature of the cloud–based model that requires the management plane to run remotely raising several questions in terms of responsiveness, robustness, availability, security and privacy. The latter is instead posed by limitation current commercial solutions that only partially address the aforementioned problems by providing the customer with an hosted network controller. Such an approach suffers a few drawbacks:

- It requires a persistent connection to the datacenter.
- It does not scale well with the number nodes.
- It poorly integrates with legacy solutions.

The contribution of this paper are threefold: we introduce a taxonomy of network control and management tasks classified according to their requirement in terms of responsiveness, robustness, availability and security; we propose a novel NaaS architecture; and we report on a proof–of–concept NaaS implementation, deployed and tested over a large scale metropolitan WiFi network. It is worth stressing that such nodes are heavily resource constrained in that they completely lack persistent storage and have very limited processing capabilities.

The rest of the paper is structured as follows. Section II discusses the challenges associated with a remote management plane. A taxonomy of network control and management functionalities closely related with a typical metropolitan WiFi deployment is the subject of Sec. III. Section IV describes our NaaS concept. Implementation details and experimental results are reported in, respectively, Sec. V and Sec. VI. Section VIII draw the conclusions and summaries current and future work.

II. Challenges and Requirements

Where are control and management tasks typically placed in network deployments today? Some functionalities, like routing, are effectively distributed across the network, while others, like configuration management are centralized. More in general, such tasks are centralized for what concerns the management aspects (e.g. security policies) while mechanisms, such as encryption or tunneling, are embedded in the devices.

In order to address such misalignment, a re–factoring of functionalities has been suggested several times, the most recent example is the 4D networking architecture [2] which involves a radical design where network functionalities are organized in such a way to simplify the data plane while moving all the complexity to a centralized controller. However, it is worth noticing that centralized approaches raise the usual concerns in terms of scalability, security, and single point of failure, instead, the authors argue for an hybrid approach designed around the following requirements:

- It shall make no assumption on the reliability of the link to the remote management plane allowing the network to function properly in the last known state even if the connection to the controller is lost.
- It shall support aggregation and processing of monitoring information in order to improve scalability paving the way to innovative in–network management approaches;
- It shall support both cloud–aware and, using suitable proxies, legacy deployments.

Notice that, with 4D–like approaches we do have a logically centralized management plane, however such a plane typically lies within the domain of the network it is managing. Conversely, a remote management plane, such as the one that is required in order to enable effective NaaS, imposes different requirements on the features which can be moved to the cloud controller and features instead must be embedded within the managed network. To this purpose, we introduce the following four orthogonal requirements — on the management plane — that we use to assess if a particular network control and management functionality can actually be moved to the cloud controller, and, if so, under which constraints:

- *Time–sensitiveness* (R_A), a remote management plane implies larger (compared to in–house solutions) communication latency in both directions, i.e. to send monitoring information and to execute administrative actions, which in time implies that *time–sensitive* features must remain within the network domain while *non time–sensitive* functions can be offloaded to the remote management plane. As for example, forwarding and encryption are time–sensitive functionalities and thus cannot be moved to a remote network controller, on the other hand traffic engineering and routing (i.e. computing the best path between two endpoints) are characterized by soft requirements in terms of latency and are thus good candidates to be relocated to the cloud controller.
- *Robustness* (R_B), a remote management plane may be unavailable due to failures at the management plane and due to outages at the communication level. As a result, additional constraints must be put on the managed devices that must support a soft–state operation model, i.e. the device must be able work in a stand alone mode after an initial configuration has been fetched. As for example, routing imposes strict requirements in terms of robustness, as a result typical routing algorithms, such as OSPF, are implemented in a distributed manner in order to keep the network connected even in case of severe failures. Moving route computation to a remote control and management plane (see OpenFlow [3]) impose new constraints on the signaling channel that must support self–healing and self–configuration features.
- *Frequentness* (R_C), some services, such as Authentication, Authorization, and Accounting (AAA) typically involve *sporadic* network operations in order to access a database or a web service. As a result, even if the remote network controller is unavailable, the data plane functionalities may still be unaffected if suitable caching techniques are implemented on the managed network side. Thus, such requirement dictates the presence of suitable proxy mechanisms capable of decoupling the managed network from transient failures that may happen at the remote management plane level.
- *Security* (R_D), a remote management plane implies that an entity other than the network owner has access to potentially sensible information. Such situation, in time, raises security and privacy concerns dictating which functionality that can be hosted outside the domain of the managed network and which shall instead be internalized using, for example, private clouds solutions. The use of *homomorphic encryption* schemes could also allow to export part of the data to the remote cloud in an encrypted form while preserving the capability to run queries and gather statistical data.

We assume that, for each dimension, the functional requirements $R_{A,B,C,D}$ imposed by a particular management functionality F can be either *Strict* or *Loose*, i.e.:

$$R_x(F) \in \{Strict, Loose\}, \quad x \in \{A, B, C, D\}$$

Then, the outcome of the assessment is the tuple $\langle R_A(F), R_B(F), R_C(F), R_D(F) \rangle$. We identify the following categories into which a certain management functionality F can be classified according to its functional requirements:

- *Remote*. Management functionalities belonging to this group do not impose any *Strict* requirements along any of

TABLE I: Requirements and the management categories.

Responsiveness R_A	**Robustness** R_B	**Availability** R_C	**Security** R_D	**Category**
Strict	*Any*	*Any*	*Any*	Embedded
Loose	*Strict*	*Loose*	*Loose*	Embedded
Loose	*Loose*	*Strict*	*Strict*	Hybrid
Loose	*Loose*	*Loose*	*Loose*	Remote

the four dimensions and can thus seamlessly be offloaded to a remote management plane. Examples are traffic engineering, or routing.

- *Hybrid*. Management functionalities belonging to this group impose *Strict* requirements in terms of *availability* and/or *security*. As a result, the use of proxy mechanism is envisioned in order to shield the managed network from transient connectivity outages to the remote management plane and/or to avoid exposing sensible data to third parties. Examples are device self–configuration services, such as DHCP, or policy authentication checks.
- *Embedded*. Management functionalities belonging to this group impose *Strict* requirements in terms of responsiveness and thus must be embedded into each device. Examples include data forwarding, and encryption.

In Table I, we report the mapping between most relevant tuple permutations and the corresponding management categories. Such mapping allow us to distinguish which functionalities must be entirely embedded in the managed network, which can be offloaded to the remote management plane, and, finally, which requires an hybrid approach involving possibly some proxy mechanism within the managed network.

III. A Taxonomy of Network Management Functionalities

In this section we shall provide taxonomy of network management functionalities classified according to the requirements they impose on the remote management plane in terms of security and robustness. We also classify time–critical Vs. non time–critical operations and sporadic Vs. frequent operations. We focus our attention on a particular usage scenario, namely provision of city–wide broadband Internet access using light infrastructure networks such as WiFi Hotspots and Mesh Networks. Such kind of deployments are typically unplanned and can grow or shrink over a short period of time according to the user demands. As a result, a truly comprehensive NaaS solution shall be able to support the network during its entire life–cycle from planning, through deployment and operation and shall support an incremental deployment model allowing connectivity to be provided only where and when needed.

The first stage of the network deployment is planning. Due to cost and complexity reasons, the usage of advanced network planning tools is typically limited to enterprise deployments where the initial cost can be spread among a large user base. In such a scenario, a NaaS offering can effectively lower the entrance barrier to such planning tools by effectively spreading the development cost among thousand of small networks.

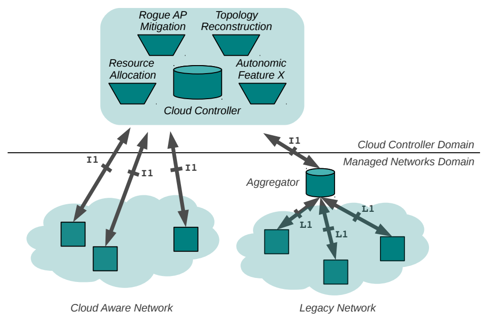


Fig. 1: A typical NaaS deployment where multiple networks are centrally managed by a single network controller.

The deployment and operation phase includes all well known FCAPS functionalities (fault, configuration, accounting, performance and security management):

- The **fault** management is based on alerts. This family of functionalities includes tasks related to prompt notification of network errors to the remote management plane as well as root cause analysis algorithms and heuristics. The former, requires a certain amount of intelligence to be put in the managed device that must be able to exploit a variety of media to notify the remote management plane of potential faults (e.g. SMS, email). The latter are typically executed within the cloud controller where suitable learning techniques can be exploited.
- The **configuration** management deals with proper configuration of the network (e.g. routing profiles, operators policies). It is strongly linked with the performance management and it is used during node/network initialization and during normal operations in order to improve network performance or to handle faults.
- The **performance** management functionalities involve passive sensing (packet capture/trace) as well as active tools to performs measurement campaigns aimed at pinpointing failures or performing profiling activities. Finally, periodic maintenance tasks such as remote firmware update, reboot, and remote diagnostics (e.g. ping) also belong to this family of functionalities typically implemented within the remote management plane.
- **Accounting** is a management feature naturally fitting within the remote management plane. This family of functionalities is, in traditional deployment, heavily customized in order to fit the customer's need. In a NaaS scenario, a trade–off is to be found between a tailored management dashboard and a standard minimal interface to be re-used across a wider customer base.
- **Security**–related features typically involve an initialization phase when an access to the actual remote management plane is required and a cruising phase when either the previously acquired credentials are directly exploited by the user or an intermediate proxy that can satisfy further authorization requests. On the other hand, features which involve tracking the user's behavior are bound

TABLE II: Functional requirements of typical network control and management tasks.

Functionality	**Category**
Planning:	Remote
• Radio coverage simulator	
• Map management	
• Clients simulator	
Faults:	Remote
• Event log	
• Alarm summary	
• Root cause analysis	
• Standard reports	
• Customize reports	
Configuration:	Embedded/Remote
• Install and Configure APs	
• Traffic shaping	
• Routing/Forwarding	
• Channel assignment	
• Deep Packet Inspection	
Accounting:	Hybrid
• RADIUS	
• Captive portal	
• Billing	
Performance:	Remote
• Packet capture/trace	
• Remote firmware update	
• Diagnostics	
• Statistics	
Security:	Hybrid
• WPA/WPA2 (Personal/Per-device)	
• 802.1x (Enterprise)	
• MAC filtering	
• Rogue AP detection and mitigation	
• NAT/Firewall	
• VPN	

to raise security and privacy concerns in that, unlike traditional deployments where the management plane resides within the enterprise administrative boundaries, in a NaaS scenario such information are exposed to entities other than the network owner.

Table II reports the mapping between the discussed management functionalities and their functional requirements.

IV. Our concept

In this section we introduce the blueprints for a remote network control and management plane capable of effectively supporting a NaaS architecture. Figure 1 depicts the system architecture backing up the envisioned remote network control and management plane. Central to this architecture is the shared knowledge base, built collecting network monitoring information from multiple managed networks, and exploitable by suitable autonomic decision policies.

Monitoring information are either collected directly from a cloud–aware network device, i.e. a device that implements the interface I_1, or trough a suitable *Aggregator* device which, on one hand, implements the I_1 interface, and, on the other hand, implements a set of legacy interfaces L_1 aggregating information coming from different sources (e.g. SNMP, IPMI, log files, etc). FCAPS functionalities as well as network–wide optimization algorithms are consolidated at the remote cloud controller where the use of a web–based management dashboard is envisioned for performing administrative actions.

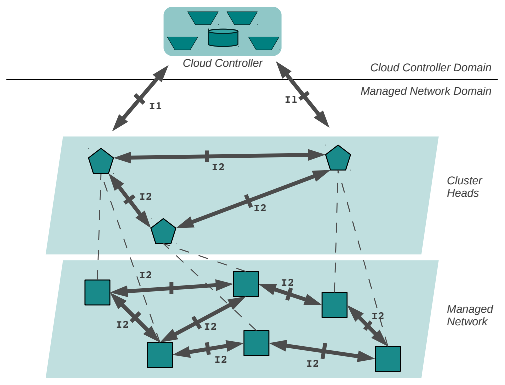


Fig. 2: A more advanced deployment exploiting distributed monitoring within the managed network.

The use of distributed monitoring functionalities within the managed network is also envisioned. Figure 2 sketches the system architecture for this scenario. Such a distributed monitoring approach is in charge of bringing network–wide information to the relevant decision points in an effective and robust manner effectively implementing a signaling layer on top of which autonomic/in–network management techniques can be effectively implemented. A hierarchical architecture is envisioned where *Cluster Heads* are in charge of gathering the local network state from their cluster and for propagating such information to the other *Cluster Heads*. A similar architecture has already been proposed by the authors and its viability has been proved over a small scale testbed [4].

It is worth noticing that a logically centralized decision point does not imply a centralized implementation. As a matter of fact, additional level of resiliency can be introduced into the system by using redundant and geographically distributed network controllers (see Fig 3). In such a scenario, managed networks can either exploit the closest controller as their management plane falling back on other controllers in case of outages or, they can use consensus protocols, such as Paxos [5], in order to take administrative actions in a distributed fashion. The use of private clouds in envisioned as a way to address privacy/security concerns.

V. The prototype

In this section, we shall describe the software prototype developed to validate the viability of a NaaS architecture in a realistic scenario as well as to collect the empirical data needed to characterize the cloud controller performance in terms of latency and scalability. The software prototype consists of a set of cloud–aware access points centrally managed by a

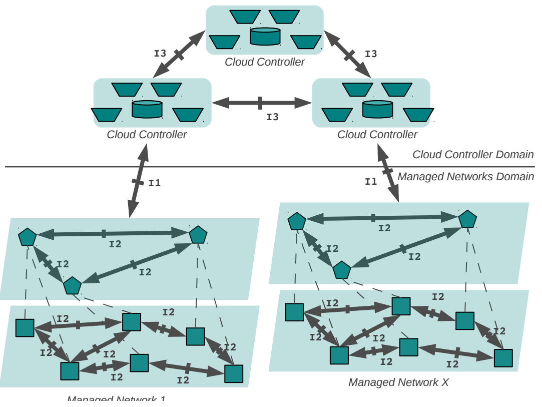


Fig. 3: A redundant network controller deployment.

cloud controller implementing basic monitoring functionalities together with two autonomic control loops which are in charge of optimizing the frequency allocation plans of the network and implementing rogue AP detection and mitigation.

These features have been selected by the network operator taking into account the amount of human resources currently spent to manually troubleshoot correlated issues, e.g. throughput degradation due an inefficient frequency allocation plan and Denial–of–Service (DoS) attacks performed by rogue APs.

The prototype has been tested over the LUNA (Large Unwired Network Applications) wireless mesh network deployed in the Trentino area in northern Italy. The network is composed of about 500 nodes implementing a two–tier architecture where connectivity is distributed using an hybrid 2.4/5 GHz wireless mesh backhaul while access to end–users is provided in the form of standard 2.4 GHz hotspots. Both distribution and access links uses standard WiFi technology.

The prototype implements a classical client/server architecture, however unlike traditional SNMP deployments, where a centralized Network Management System (NMS) periodically polls the managed devices, in our case, network state updates are sent by the managed device only when needed, i.e. when a change in a managed object has been observed or when an alarm condition has been verified. The building blocks of our prototype are sketched in Fig. 4, its features include:

- Real–time monitoring of the nodes in the network.
- Raising accurate events in case of failures (e.g. a node leaves the network) and gathering notifications (e.g. high link utilization or low memory).
- Optimizing the frequency allocation plan across the distribution and the access networks.
- Minimizing the overhead any access point incurs when in case of DoS attacks performed by rogue APs.

The information collected by the agents falls into the following categories: link quality (signal/noise level, retransmissions), traffic analysis (packets/bytes transmitted/received over each interface), and node-wide parameters (processor load, system uptime, available memory).

Each cloud–aware managed device runs a software agent

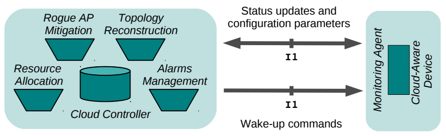


Fig. 4: The prototype architecture.

which implements the management interface I_1 used to communicate with the cloud controller. The agent periodically polls the local node for changes in the state of the managed objects and, if necessary, sends the updated values to the cloud controller. The I_1 interface is implemented according to the RESTful architectural guidelines [6] using the HTTP protocol.

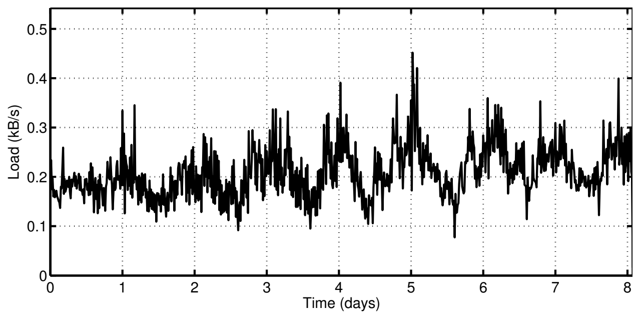
Administrative actions are always performed by the controller as a result of either human intervention or autonomic decisions. Each action may result in a set of commands to be scheduled for execution at different nodes. Each command can modify one or more managed objects (i.e. a wireless interface operating channel) belonging to a single node. Administrative actions involving more than one node requires the execution of multiple commands. Each command is atomic, i.e. all changes at a given node must be successful otherwise the entire transaction is aborted. All signaling traffic is carried over a secure SSL–enabled channel. Commands scheduled for execution are dispatched within the acknowledgment to the first update message coming from the node that is meant to run the command. The controller can force an agent to send an update message by connecting to a known port and passing a security challenge.

It is worth noticing that, the system has been designed in such a way to minimize both the amount of traffic and the number of active connections. The former goal has been achieved by sending updates only for managed objects that have changed since the previous update. The second goal has been achieved by using a stateless communication protocol that do not require persistent TCP connections, freeing precious entries in the NAT table at the aggregation gateways. Finally, minimizing the management traffic flowing across the network is of capital importance given the fact that, in the LUNA network, the control and management plane shares the same medium used by the data plane.

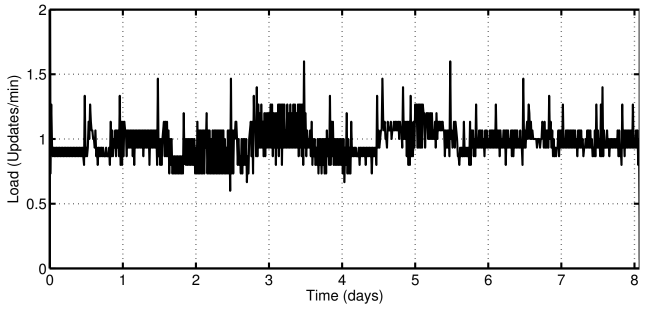
The cloud controller is implemented in the form of a Java EE Servlet ran by a Tomcat 6.0 web server. The cloud controller implements a multi–threaded model allowing it to gather updates coming from the managed devices while executing the autonomic control loops. It is worth noticing that, no self–interference mechanism between control loops is currently supported by the system, as a result the autonomic task developer is in charge of avoiding situations where concurrent optimization loops may take conflicting actions.

VI. Evaluation

In this section, we aim at assessing the scalability of our prototype by studying the relationship between an increasing



(a) Bytes per second.



(b) Packets per second

Fig. 5: Network state updates received by the controller over a 8–days period.

number of managed devices and the controller performance in terms of latency in reacting to changes in the network. The network over which our prototype is being tested is composed of about 500 nodes deployed over a metropolitan area, however, due to both engineering and logistics challenges, the results reported in this paper refers to a preliminary pilot consisting of only 10 nodes. In particular developing an agent capable of satisfying production–level quality constrains in terms of memory and CPU footprint proved to be particularity challenging. On the up side, we do believe that running a practical system over a production network composed of 500 nodes without affecting the operation of the network itself is a result that can speak in favor of an incremental deployment of a cloud-based network controller.

Figure 5 reports the load at the cloud controller over a period of 8–days. Updates are sent by each node every 300 seconds. As it can be seen from the figure, the amount of traffic exchanged is constant over time and the signalling bandwidth *for each AP* is roughly 20 bytes/s. Notice that additional traffic may be generated if an alarm is triggered.

Another important parameter to be considered is the transaction latency i.e., the time interval between the instant at which a command (e.g. changing a wireless interface's operating channel) is scheduled by the controller and the instant at which the corresponding acknowledgment is received.

It is worth noticing that, after receiving a command, the agent executes the operations specified in it, and then generates a new update message containing the values of the managed objects affected by the command. This latter update message

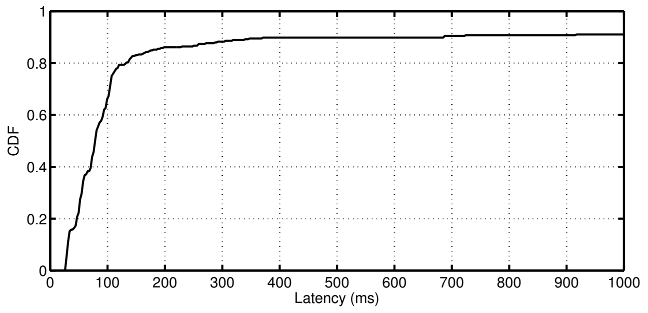


Fig. 6: Empirical cumulative distribution function of all transaction latencies over a period of 8–days.

plays the role of acknowledgment to the original command. During our measurement campaign, all commands have been issued as urgent, thus the latency includes the propagation delay, the signaling overhead, and the time spent by the device actually executing the command.

Figure 6 reports the empirical cumulative distribution function of all transaction latencies over a period of 8–days. As it can be seen from the figure, half of the transactions are completed within less than 80 ms, while 700 ms is the time required to complete 90% of the transactions.

VII. Related work and Research Challenges

In this section we first review the related works in both commercial and academic domains, highlighting open issues and research challenges, and later we will discuss the impact of the NaaS concept on tussle spaces.

A. Commercial Solutions

Similar solutions are already commercially available from vendors such as Meraki Networks in the form of a vertical offering which include both hardware and the network management services delivered through a web–based dashboard. Likewise, it is already possible to exploit proprietary and open-source software in order to implement an effective hosted NaaS solution. As for example, the open source software Zabbix allows to monitor distributed IT infrastructures by implementing an hierarchical architecture where each managed node reports its state to a master node where the global view of the network is reconstructed. In both cases however several drawbacks remain unaddressed.

In particular, the Meraki approach introduces a single point of failure in the system, namely the cloud controller by making the network vulnerable to failures at the cloud controller site as well as to outages at the communication layer. Moreover, being a vertical solution, the Meraki offering does not address the legacy network deployments.

Likewise, custom hosted network control and management solutions based on opensource tools suffer on one hand of the lack of sophisticated monitoring and optimization tools (due to cost/complexity considerations); and on the other hand of severe scalability issues as the number of nodes increases (due to the polling–based nature of typical software solutions).

The software prototype discussed in this paper demonstrates the viability of the remote management plane in a realistic scenario composed of more than 500 nodes serving more than 18000 unique clients. The prototype easily be extended to support additional monitoring information from both cloud–aware and legacy devices as well as to implement advanced control loops for both optimization and troubleshooting.

B. Academic Efforts

The authors in [7] envision a scenario where route computation is offered "as a service" by a centralized controller which in time dispatches the forwarding rules to the routers. Such kind of approach is generalized in the 4D networking architecture [2] where the operation of thin data plane devices is coordinated by a centralized decision plane. Is it worth noticing that, albeit the decision is centralized, the underlying logic can be distributed across several machines. As for example in [8] a Paxos distributed state machine [5] is used in order to allow control and management processes running on network endpoints to act as they were logically centralized. Ethane [1], SANE [9], and OpenFlow [3] are further focusing on refactoring management functionalities in order to make the centralized controller a viable alternative. Tesseract [10] is a prototypical implementation of the 4D concept.

Parallel efforts in the WLAN domain envision a variable degree of cooperation coming from client nodes in the network management tasks. Dyson [11] is a novel WLAN architecture built on top of standard WiFi devices and designed to enable extensive customization and control over several aspect of network operation and performance, such as how a client associates with an access point or how radio channels are allocated. Conversely other centralized systems such as DenseAP [12] and DIRAC [13] take a similar approach but assume that no change can be done on the client's software stacks. Centaur [14] foresee some modifications on the end–users' clients in order to tackle the hidden and exposed terminal issue in IEEE 802.11–based wireless LAN. Client cooperation using custom beacon probes is used in [15] in order to drive the association phase or in [16] in order to build a conflict graph for the network.

Particularly important are the efforts devoted to devise novel automatic diagnosis of network configuration errors both at run time [17], [18] and off–line [19]. Along the same line, the authors of ConMan [20] argue that a centralized solution is instrumental in keeping the management plane in sync with the rapidly evolving data–plane and thus to prevent ossification at the control and management level.

In [21], a conceptual architecture for autonomic computing is sketched. The authors analyze the motivation behind the quest for autonomic computing and focus their attention on the control loops introduced by the self-management routines. In their work, the authors envision a common knowledge of the system on top of which control algorithms are implemented.

More in general it is worth noticing that, when our cloud controller is put in a wider perspective, it becomes compulsory to embed autonomic and distributed monitoring functionalities within the network itself as well as to support an additional level of redundancy at the cloud controller level. Similar efforts are carried on by the CASCADAS project [22]. More recently, many hierarchical and federated structures have been proposed such as [23], [24].

C. Impact on tussle spaces

Particular consideration must be given to the new tussle spaces [25] introduced by a NaaS architecture. A tussle space is a portion of the network where adverse or at least different stakeholders play in order to achieve their respective interests.

In the envisioned NaaS offering, at least four adverse stakeholders are interacting on the playing field, namely: end–users, i.e. SMEs, public administration, individual businesses, and residential home networks, NaaS providers, and cloud computing infrastructure, and central government policies providers. In such a scenario it is of capital importance to both modularize the system design around tussle boundaries so that unrelated issues and conflicts remains isolated. Moreover it is important to design the system for choice in order to allow different players to pursue their interests.

The following types of tussle spaces can be foreseen:

- end–users are sceptical about outsourcing their entire networking infrastructure to a third party due to both the feeling that their control over the network will be reduced *and* the fear that sensitive information will be revealed to the NaaS provider. On the other hand, NaaS providers want to enlarge their customer base to both improve their economies of scale as well as to feed optimization and planning algorithms with a wider knowledge base.
- NaaS providers want to exploit flexible and scalable cloud computing infrastructures without risking of being locked with a single provider. On the other hand cloud computing providers already divide customers into different classes according to their subscription and willingness to pay; NaaS providers will probably be recognized as a high value tier due to the nature of the service offered (control and management plane can hardly be considered a commodity) and charged accordingly.
- Central government, but also end–users, can require special privacy constraints on how and where data coming from a network is stored. As a matter of fact, it is not uncommon for a government to mandate that some data, even when it belongs to private citizens, to be stored within the country's borders and be made accessible for monitoring in the name of national security.

The architecture devised in this paper does involve a modular design allowing tussle in each of these spaces to be logically separated from the others. In particular, the use of the platform agnostic interfaces I_1 to retrieve network status updates as well as to perform administrative actions, allow the customers to incrementally deploy a NaaS solution across their network and, through the aggregation point, to control which information are exposed to the NaaS provider. Likewise, the in–network interface I_2 allows critical or privacy–constrained functions to be embedded within the network itself without having to rely on neither the availability nor the trustworthiness of the cloud controller. Finally, the inter–controller interface I_3 enables scenarios where multiple redundant cloud controllers are first class citizens. It is worth noticing that, as suggested in [25], the choices that led to our proposed NaaS architecture took into consideration that tussle spaces evolve over time, thus the playing field, i.e. interface, algorithms, and protocols, must be designed in such a way to leverage variations instead of mandating for a rigid model that will inevitably break under pressure.

VIII. Conclusions

In this paper we argued for a split–network architecture where computationally intensive tasks as well as tasks that require an advanced domain knowledge (i.e. root cause analysis) are taken out from the network, moved to the cloud, and offered *as a service* by the NaaS provider. It is worth noticing that, management functionalities are decoupled from both devices and network management tools. It is the authors' standpoint that, with the exception of a few cases where a certain amount of refactoring is required, a significant part of the network control and management plane can be moved to the remote centralized controller with no significant changes.

Our proof of concept focuses substantially on the communication channel between the managed network and the cloud controller and aims at supporting the claim that non–trivial functions such as dynamic channel assignment and DoS mitigation can be effectively offloaded to a remote controller. In this work we also aimed at providing a methodology that can be used to assess which tasks can be offloaded to a remote management plane. Future work shall investigate the other two aspects that, in our opinion, characterize and NaaS offering, namely elasticity and manged virtualization.

In order to prove the feasibility of the proposed approach, a software prototype consisting of a cloud–aware agent and a cloud controller has been implemented. Experimental results, gathered over a production wireless network consisting of about 500 WiFi Access Points deployed in a metropolitan area, demonstrate the viability of our architecture in a practical scenario paving the way to innovative NaaS offerings.

Albeit the preliminary results are promising, several questions remains unanswered. In particular, it is not clear how the cloud controller will scale with the number of nodes nor if it is robust to outages at the signaling channel level. The authors are currently tackling this challenges by progressively enlarging the pilot presented in this work to a wider portion of the network. Moreover, while the current cloud controller was running a few hops away from the managed network, a more advanced version is currently being developed and will be deployed on commercial cloud computing platforms (i.e. Microsoft Azure, Google App Engine, and Amazon EC2). This version will allow us to validate the redundant cloud controller concept discussed in this paper. Integration with distributed monitoring solutions is also being considered.

References

[1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proc. of the ACM SIGCOMM*, 2007.

[2] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A Clean Slate 4D Approach to Network Control and Management," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, 2005.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008.

[4] R. Riggio, M. Gerola, D. Miorandi, A. Zanardi, and F. Jan, "A distributed network monitoring framework for wireless networks," in *Proc. of IEEE ManFI*, Dublin, Ireland, 2011.

[5] L. Lamport, "Paxos made simple," *ACM SIGACT*, vol. 32, pp. 51–58, December 2001.

[6] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002.

[7] K. Lakshminarayanan, I. Stoica, and S. Shenker, "Routing as a service," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-04-1327, 2004.

[8] H. Uppal, V. Brajkovic, D. Brandon, T. Anderson, and A. Krishnamurthy, "Ettm: A scalable fault tolerant network manager," in *Proc. of USENIX NSDI*, 2011.

[9] M. Casado, T. Garfinkel, M. Freedman, A. Akella, D. Boneh, N. McKeowon, and S. Shenker, "SANE: A Protection Architecture for Enterprise Networks," in *Proc. USENIX Security Symposium*, 2006.

[10] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, "Tesseract: A 4d network control plane," in *in Proc. USENIX NSDI*, 2007.

[11] R. Murty, J. Padhye, A. Wolman, and M. Welsh, "Dyson: An Architecture for Extensible Wireless LANs," in *In Proc. USENIX NSDI*, 2010.

[12] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill, "Designing high performance enterprise wi-fi networks," in *In Proc. USENIX NSDI*, 2008.

[13] P. Zerfos, G. Zhong, J. Cheng, H. Luo, S. Lu, and J. J. ru Li, "Dirac: a software-based wireless router system," in *Proc. of ACM MOBICOM*, 2003.

[14] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "Centaur: realizing the full potential of centralized wlans through a hybrid data path," in *Proc. of ACM MOBICOM*, 2009.

[15] I. Broustis and M. Faloutsos, "Mdg: measurement-driven guidelines for 802.11 wlan design," in *Proc. of ACM MOBICOM*, 2007.

[16] N. Ahmed and S. Keshav, "Smarta: a self-managing architecture for thin access points," in *Proc. of ACM CoNEXT*, 2006.

[17] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker, "Netprints: diagnosing home network misconfigurations using shared knowledge," in *Proc. of USENIX NSDI*, 2009.

[18] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Proc. of ACM SIGCOMM*, 2009.

[19] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in *Proc. of USENIX NSDI*, 2009.

[20] H. Ballani and P. Francis, "CONMan: A Step towards Network Manageability," in *Proc. of ACM SIGCOMM*, 2007.

[21] "Practical autonomic computing: Roadmap to self managing technology," IBM, 2006.

[22] "Cascadas project." [Online]. Available: http://acetoolkit.sourceforge.net/cascadas/index.php

[23] J. Famaey, S. Latre, J. Strassner, and F. De Turck, "A hierarchical approach to autonomic network management," in *Proc. of IEEE NOMS Workshops*, Osaka, Japan, 2010.

[24] B. Jennings, R. Brennan, W. Donnelly, S. Foley, D. Lewis, D. O'Sullivan, and J. Strassner, "Challenges for federated, autonomic network management in the future internet," in *Proc. of IEEE IM*, Long Island, NY, USA, 2009.

[25] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: Defining tomorrows internet," in *In Proc. ACM SIGCOMM*, 2002.