**UNIVERSITÀ DEGLI STUDI DI PARMA**

*Dottorato di Ricerca in Tecnologie dell'Informazione*

*XXIX Ciclo*

# DESIGN, IMPLEMENTATION AND OPTIMIZATION OF INNOVATIVE INTERNET ACCESS NETWORKS, BASED ON FOG COMPUTING AND SOFTWARE DEFINED NETWORKING

Coordinatore:

*Chiar.mo Prof. Marco Locatelli*

Tutor:

*Chiar.mo Prof. Gianluigi Ferrari*

Dottorando: *Nicola Iotti*

Dicembre 2016

*To my wife*
*and my family*

# Contents

# List of Figures

# List of Tables

# Introduction

End devices, connected to the Internet, have experienced a significant evolution during the last ten years. Smartphones and tablets have been introduced, beside notebooks, and the concept of mobile applications (App), running on user's devices, has radically changed the behavior and expectation of Internet users. Data centers have also deeply evolved owing to the introduction of new technologies. The concept of virtualization brought a totally new generation of data center and opened the way to the evolution of the Cloud and to a new way to design applications and services. However, access networks, that are in the middle, failed to support this revolution.

Internet access networks in public areas, such as hotels, retail shops or town centers, have witnessed a remarkable evolution in terms of performance, Authentication Authorization and Accounting (AAA) features, registration options, and capability to localize and engage users however, they are basically still relying on a traditional approach, where connectivity is provided by deploying network devices, such as switches, routers, access points or middleware, able to implement specific network functions. These functions are typically used to connect clients to applications running on the web or in the Cloud. Also from the point of view of the business owners, that deploy Internet access networks in their locations for customers and guests (such as, for example, retailers or hotels), the network infrastructure remains a black box that just has to work. While these brands are developing highly specialized mobile App, web sites or core services, the access networks in their locations are far less specialized. If we consider two different types of business, such as stores and hotel chains, we can verify that they have very different mobile Apps, websites, cus-

tomer's databases, but basically the same Wi-Fi access network with access points, switches, (Virtual Local Area Networks) VLANs and similar authentication systems. This means that the network is still not optimized and the design of this network portion is totally different compared to the software elements.

Moreover, Internet access networks are now facing critical challenges related to the ever increasing number and type of connected devices and to the strict requirements of new generation of services. The number of connected devices is rapidly growing and recent forecasts estimate an acceleration of this trend [1]. Connected devices will be also more heterogeneous, including different types of users' devices and all kinds of smart objects. The evolution of the Internet of Things (IoT), as well as the changing of users' devices and habits, will increase the proliferation of services with different requirements, in terms of computation, storage, bandwidth, and real-time interaction capabilities.

In general, Internet access networks in public areas are exposed to critical issues, because of the rapid changes in connected devices and services, including the IoT revolution, because the Internet access technology did not evolve in parallel to end devices and the Cloud thus it failed to support the shift to a mobile lifestyle of the users. Access networks also lost the capability to play an active and specialized role in supporting the business of the locations owners. Building networks that are just able to provide more resources, or differentiating access networks to accommodate different types of services, is not the right approach to handle the workload and the complexity brought about by these new challenges. For this reason, academia and industry are putting significant efforts to develop new network architectures that take into account the very nature of these changes, rather than just blindly and naively boosting the underlying network bandwidth capabilities.

In this dissertation, we introduce a new approach to Internet access networks in public spaces, such as Wi-Fi network commonly known as *Hotspots*, based on Fog Computing (or Edge Computing), Software Defined Networking (SDN), and the deployment of Virtual Machines (VM) and Linux Containers on the edge of the network. The proposed infrastructure is based on these elements and exposes Representational State Transfer (REST) Application Programming Interfaces (APIs) in

order to make them able to interact with external applications, either running in the Cloud or on end-user devices, which can trigger and control the deployment of virtual environments.

Fog Computing, or Edge Computing, is a novel paradigm that aims at optimizing networking, computing, storage resources and improving the quality of service brought to users (in terms of latency and throughput), by moving resources at the edge of access networks [2]. This approach is set to avoid issues that affect traditional Cloud-based solutions, such as low-bandwidth, congestion of Internet connection, and infeasibility of real-time applications. In our approach, the consequence of the introduction of Fog Computing is the deployment of specific nodes in Internet access networks, called *Fog Nodes*, with virtualization and SDN capabilities. Recently, a specific consortium was created namely, the OpenFog Consortium [3]. While Fog Computing is becoming a well investigated topic mainly related to IoT services, here we apply this approach to Internet access networks in general.

SDN is an emerging networking paradigm [4], that decouples the control plane from the data plane, thus unlocking the ability of dynamically deploying on-demand Virtual Network Functions (VNFs). This networking approach, was developed for data centers, where the huge deployment of servers virtualization imposed the virtualization of networking infrastructure as well, but the design of SDN solutions for Internet access networks is still limited to specific use cases. In particular, SDN does not involve Wi-Fi Internet access in public spaces or business locations, such as retail stores, hotels, enterprise offices.

Containers [5] are the result of an evolution of virtualization technologies that bring more lightweight environments for fast and dynamic application deployment.

The logical development of our work starts with an analysis of data collected on field in existing highly crowded Wi-Fi Internet access network, in order to identify main issues in traditional deployments, and measure the potential benefit of an infrastructure based on Fog Computing, compared to a traditional approach. After this validation analysis, that has the important role of supporting and legitimate this approach, we focus on the design of the platform, describing the general vision and defining the details of the infrastructure of Internet access networks, based on this

vision. In order to prove the feasibility and extend the concept of Fog-based network infrastructure, we introduce two specific use cases: one related to Internet access services with users authentication and the second regarding a specific application of IoT. In the first case, we analyze data collected from a testbed, replicating Internet access services commonly deployed in public areas, in a laboratory of an Italian company called Caligoo, in collaboration with the University of Parma and another company named PLUMgrid, based in USA. In the second case we introduce a specific platform for IoT, where, the envisioned approach allows to design advanced services, related to the interaction with smart objects. In both cases we collected data in order to prove the feasibility of solutions, based on the envisioned approach, evaluating in particular performance and needed resources. We then describe some real use cases of application of Fog-based approach to the access networks, deployed by a service provider in Italy, analyzing the benefits brought by this vision. In particular we focus on three core services for Internet access service provirders: i) authentication; ii) location analytics; iii) computing resources optimization. After the validation analysis, the description of the proposed platform, the description of specific use cases and real deployment, we close our dissertation, focusing on the Fog Computing layer. This theoretical section aims to provide some advanced considerations on the design and optimization of the Fog-based platform, introducing the concept of static and dynamic overlapping clusters and identifying the elements that make one clustering method better than others.

This thesis was carried out in partnership with companies in Italy and in USA. In particular this work is partially based on a 6-month internship at PLUMgrid and Nebbiolo Technologies, two companies based in Silicon Valley, California, USA. The work during this Ph.D has led to several scientific publications and to USA patent submissions.

The structure of this dissertation follows the logical development of our work. In Chapter 1, we describe the state of the art analyzing the related works In Chapter 2, we validate our approach evaluating the potential benefit, analyzing real data collected on field, in traditional networks. In Chapter 3, we introduce our vision and evaluate feasibility considering a basic deployment of the envisioned infrastructure,

while in Chapter 4 we describe an advanced service for the IoT, made possible by the proposed architecture. In Chapter 5, we present some real use cases deployed by a Wireless Internet Service Provider (WISP) and in Chapter 6 we approach the problem of the optimization of Fog-based access network, proposing some models for the structure of this new layer. Chapter 7 concludes the dissertation.

# Chapter 1

# Related Works

## 1.1 Architecture

The evolution of wireless access networks is a topic that has been deeply investigated and is rapidly evolving from research to deployment on field. Fog Computing, or Edge Computing, is playing a key role in the design of future networks, introducing the capability to deploy and control storage and computing on the edge of the network and not only in the Cloud or on end devices. A well-defined description of Fog Computing architecture is introduced in [2], and, even if the concept of Fog Computing is general, the approach is mainly focused on IoT applications. Other works introduce an architecture, similar to the one we are envisioning, including a smart gateway or Hub deployed on the edge,focused on network management and data processing before or instead of sending data directly to the Cloud, [6] and [7], but specifically for IoT networks. In [8], an example of storage functions dynamically deployed in selected network sections, in order to speed up data transfer, is provided. Other works are focused on specific topics related to Fog Computing, such as security [9] and reliability [10], but without introducing a specific architecture for users access networks. In [11] Fog Computing architecture is applied to face performance issues on edge network, envisioning a very advanced platform to manage local servers in order to optimize web content rendering and formatting.

The possibility to use Fog-based Access Points and user equipment is envisioned in [12], in order to provide a very granular distribution and workload based on memory sharing. Other works are mainly related to network resource management. In particular, clustering techniques of sharing data and hardware resources in order to implement a flexible management of distributed infrastructure, are proposed in [13]. [14] presents a method for placement and migration of virtual machines, for Cloud and fog providers and [15] introduces a system to predict future query regions for moving consumers, able to provide information in order to process events early. The definition of the technologies involved in the Fog layer, is an important point for the design of edge networks and numerous studies are proposing and experimenting different architectures. [16] provides a good description of Fog Computing and possible applications with an introduction to SDN-based deployments, albeit limited to vehicular networks applications.

The deployment of dedicated nodes in a Fog Computing architecture is not new. In [17], a Fog-based architecture for access network with SDN is introduced, but specifically for Evolved Packet Core (EPC) networks for cellular operators: here, we have a specific focus on wireless Internet access network in public spaces, such as Wi-Fi hotspots. This advanced approach includes the use of SDN solutions, without the interaction with containers and external applications we are envisioning, and it can not reuse existing infrastructure or devices, while our platform can introduce novel Fog Node-based features without changing the existing wireless access infrastructure and users' devices. A method of moving computation from the Cloud to the network devices, deploying applications (mostly for data pre-processing) directly on network devices is described in [18]: unlike our approach, no SDN solution for networking nor containers for the deployment of applications are used. A good introduction to Fog Computing technology and challenges is proposed in [19], including the use of SDN and the deployment of applications at the edge of the network.

Many of the envisioned infrastructures have common points with the one we are proposing. However, compared to other similar architectures, we are introducing the usage of Linux Containers and Software Defined Networking (SDN) on an integrated platform able to deploy virtualized environments, including applications and

networking. We are also introducing the interaction with external applications for a more flexible control of the infrastructure. In this dissertation, we started from the analysis of real data in order to estimate potential benefits of this approach. This part, focused more on leveraging storage capabilities, locally provided by Fog nodes, and migrating contents, according to users' needs and connection resources, represents an important validation of the approach we are introducing. Then we focused on the evolution of the architecture including networking and applications deployment, as a result of the interaction with external services, and on a comparative performance analysis of the proposed platform with respect to a traditional platform.

## 1.2 Fog - Cloud Interaction in IoT

The interaction between the Fog infrastructure and the Cloud, is an important element in our analysis. In order to collect information about the performance of this interaction and about the resources needed by Fog nodes, in order to evaluate the feasibility of this approach, we studied a IoT-related application, developed introducing Fog Computing layer between smart objects and the Cloud.

The role of Cloud Computing in the IoT is gaining greater and greater attention. Most research has been so far smart object-driven, focusing mainly on the definition of IP-based, low-power, and efficient communication protocols and mechanisms. Many of these aspects have now been significantly addressed.

Several IoT solutions have been deployed and brought to the market in several application scenarios, from Home Automation to Smart Cities. Most of these fragmented and vertical solutions rely on the Cloud, in order to provide a centralized access to services exploiting data that are sent uplink from deployed sensors to Cloud storage. Typically, mobile apps "consuming" such services are made available to end-users. However, this approach, which is expedient to disseminate the concept of IoT, does not fully exploit the potential of the Cloud. As billions of smart objects are expected to be deployed pervasively, efficient data processing has highlighted the need to rely on the Cloud. The Cloud of Things (CoT) refers to the interaction between IoT and the Cloud [20].

In [21], an architecture for integrating Cloud/IoT is proposed, based on a network element, denoted as "Smart Gateway," which is intended to act as intermediary between heterogeneous networks and the Cloud. The role of the Smart Gateway is similar to that of the element we introduced in our lab experiment (see Chapter 4) called *IoT Hub*, in terms of supporting several heterogeneous networks. However, the role of the Cloud is mainly envisioned as a data storage and aggregator, which can be used by end-users to access data. According to this approach, data are sent uplink, making it impossible to directly address and act on smart objects, as the IoT is supposed to do. At the opposite, in our experiment, we envision that the Cloud, by hosting replicas of the IoT Hub, is also used as an enabler for direct and efficient access to resources, while providing desirable features, such as: seamless access by external clients; security; and high availability. Fog Computing aims at distributing and moving some Cloud-based computation and storage to the edge of the network. The Fog is a Cloud close to the ground and, as such, provides end users with functionalities closer to themselves, thus improving performance, by fulfilling real-time and low-latency consumers requirements, and enabling new applications which can also take into account location-related context information. Characteristics features of Fog Computing applied to IoT are the following:

- geographical distribution, in contrast with the centralization envisioned by the Cloud;
- subscriber model employed by the players in the Fog;
- support for mobility.

Fog Computing brings a new approach to Internet access networks by making computation, storage, and networking resources available at the edge of access networks. This improves the performance, by minimizing latency and availability, since resources are accessible even if Internet access is not available [22]. Fog-based solutions aim at introducing an intermediate architectural layer where resources and applications are made available in the proximity of end devices, thus avoiding continuous access to the Cloud. While Cloud-only architectures can provide a solution to

scalability and flexibility issues by distributing resources among multiple servers, this approach presents some weaknesses, such as: (i) latency; (ii) availability/dependence on Internet connectivity for operations; (iii) flexible networking; (iv) quality of service/experience; (v) security and privacy. Due to its benefits over Cloud-based architectures, especially if time is a critical issue or Internet connectivity is poor or absent, Fog Computing is expected to play a key role in the deployment of IoT applications.

The Fog is not intended to replace the Cloud, but rather to complement it, in order to provide location-aware and real-time services, thus enabling new applications that could have not been deployed otherwise. Fog-based access networks are based on the presence of highly specialized nodes, denoted as Fog Nodes, able to run distributed applications at the edge of the network. In particular, the deployment of computing resources on Internet access networks allows to dynamically activate Virtual Machines (VMs) dynamically on Fog Nodes. For this reason, the cloning and synchronization techniques of VMs, at the core of our IoT application (see Chapter 4), fit perfectly into Fog-based infrastructures. The proposed architecture can protect local resources by providing remote access to their replicas in a transparent way. Local resources are kept synchronized by multiple clones of the same machine, thus achieving a high level of reliability and load balancing. Smart management of the activation/deactivation of replicas and choice of the most appropriate Fog Node to run the clone allows to optimize the usage of CPU and memory available on the infrastructure, according to the specific real-time resources requirements by running applications. A suitable lightweight alternative to VMs is represented by containers, which provide a more flexible environment for "disposable applications," like the IoT Hub. Container platforms like Docker [23] are gaining increasing attention also for edge and Fog computing applications. In this challenging scenario, the possibility of moving from centralized to decentralized paradigm to offload the processing to the edge reducing application response time and improving overall user experience will play a fundamental role in Internet of Things. In [24][25], the authors present how a container-based architecture could be efficiently used for dynamic networking application. In [26], an interesting comparison about existing lightweight and hypervisor-based approaches for edge computing and efficient networking is pre-

sented. Furthermore, in [27] a novel approach for the application of a lightweight virtualization technology (such as Docker) to constrained devices with a negligible overhead is presented.

## 1.3 Resources Management and Infrastructure Design

Internet access network performance is an important topic, and it is possible to find many researches investigating different method of measurement. [28] introduces an interesting analysis about how different factors, from the modem chosen by the user, to the ISP's traffic shaping policies, can affect performance. This study shows many important characteristics related to resource utilization in existing access networks, deploying measurement infrastructure directly in user's home gateway. A more traditional approach, based on passive traffic measurement from DSL provider network can be found in [29], [30] and [31].

Resources optimization in the Cloud, is a well investigated topic, [32], for example, introduces server optimization techniques, based on load balancing design in the Cloud.

Techniques for resources and performance optimization, applied to Fog Computing networks, are introduced in [33] and [34], and are mainly related to improving Internet access performance.

Our analysis is more focused on resources management and design of the Fog infrastructure, rather than on the performance, and the scenario we are describing is similar to the one introduced in [35], even if, in this case, SDN capabilities are used on residential Internet access networks, and mainly for the deployment of Virtual Network Functions (VNF) rather than environment, including applications, in public Internet access networks.

A very interesting presentation of a mathematical approach to network optimization is provided in [36] and it can be considered a generic method for modeling many different optimization problems in networking.

# Chapter 2

# Validation Analysis of a Fog-Based Wireless Access Networks

## 2.1 Chapter Introduction

In this Chapter, we present a thorough analysis based on real data, in order to evaluate the potential benefits of a Fog-based approach to access networks. In particular we focus on Wi-Fi networks for Internet access in hotels and we investigate the benefits this approach brings, compared to traditional access networks, to the optimization of bandwidth usage and the improvement of user experience, due to a direct access to specific contents hosted in the Fog. The proposed model's idea is to host applications close to users by relying on virtual machines, in order to dynamically move Cloud or web contents to nodes located at the edge of access networks. This allows to perform proactive caching and to enforce traffic policies based on the interaction between access infrastructure and external applications. The goal of this analysis is to obtain information about the amount of traffic that could take advantage of this infrastructure, including an evaluation of the resources that need to be available. We analyze real data collected on field, in order to evaluate the benefits of Fog Computing on bandwidth optimization, related to functions that would infeasible through a traditional approach. The results of this work represent an important validation of a

Fog-based model for Internet access networks, which is the first preliminary and essential step to legitimize the adoption of new generation Fog-based access networks. This study brings a new meaning to Fog Computing, by transforming it from an effective architecture for IoT applications to a powerful approach that can potentially have a disruptive impact on Wi-Fi Internet access services, in particular in those locations where the following are relevant issues: i) Internet bandwidth; ii) high user density; or iii) shared infrastructure for Internet users and IoT nodes. Illustrative scenarios include cruise ships, trains, airplanes, hotels, and convention centers.

This preliminary evaluation of potential benefits on bandwidth optimization, aims to legitimate and justify the design of a new generation of Internet access networks, introduced in Chapter 2.

## 2.2 Validation Analysis

We analyze collected data in order to evaluate potential benefits introduced by the adoption of a Fog-based platform on quality of user experience, in terms of: i) how much data can be moved to the Fog Node for local access; ii) how it is possible to optimize the Internet bandwidth utilization, downloading data, for future local access, when the connection is underused, leveraging applications running locally on the Fog Node; iii) collect useful elements for a valid estimation of the storage, needed locally in a hotel, in order to deploy these services; iv) evaluate the impact of interactive bandwidth management.

### 2.2.1 Experiment Description and Collected data

We modified the Access Controller, that will play the role of Fog Node in the envisioned approach, in three large hotels, in order to collect data on the existing Wi-Fi Internet service, in the city of Milan, Italy, where Caligoo fully manages the Wi-Fi Internet access for hotel guests. We refer to the three hotels involved in this analysis as Hotel A, Hotel B and Hotel C. All used data are completely anonymous, in order to preserve users' privacy and customers data is not collected and not considered as part of the analysis. We collected more than 50,000 connections from over 13,800

hotel guests in February 2015. Since these hotels are mainly for business travelers, February represents a reliable perspective of normal utilization. All three locations and guests were not aware about the data collection, in order to avoid biasing user behaviors with respect to usual Internet utilization. By analyzing the connections, we were able to collect the following data:

- connection start time: ($t_0$);
- connection stop time: ($t_1$);
- connection duration, derived as ($t_1$-$t_0$);
- the amount of downlink data (dimension: [bytes]) during the connection;
- the amount of uplink data (dimension: [bytes]) during the connection;
- the actual bandwidth consumption calculating the average value over 1-hour interval on the number of bytes received and transmitted on the Fog Node interface connected to the Internet router;
- the Internet traffic generated and received by each of the three locations;
- the number of connected users.

### 2.2.2 Data Analysis and Traffic Optimization

The most important data, for the purposes of our analysis, is related to (i) the number of users; (ii) the amount of traffic generated by the connections; (iii) the bandwidth usage; and (iv) the type of traffic. In our analysis, we consider all connected devices as users. Table 2.1 reports the following data:

- the maximum number of devices connected in a single day;
- the average number of devices connected in a single day;
- the total number of devices that activated at least one connection during the considered period;

| | Max | Avg. | Total | Download [MB] | Upload [MB] |
|---|---|---|---|---|---|
| Hotel A | 767 | 607.76 | 9517 | 2285443 | 690781 |
| Hotel B | 297 | 180.48 | 1694 | 990151 | 91425 |
| Hotel C | 184 | 132.48 | 2609 | 379242 | 102130 |

Table 2.1: Data related to the numbers of connected devices: maximum per day; average per day; total, with at least one connection, during the entire monitoring period (one month) and total Internet traffic, in both download and upload.

- the total download traffic (dimension: [MB]);
- the total upload traffic (dimension: [MB]).

We decided to compute the average bandwidth utilization in 1 hour-long intervals. The results are shown in Figure 2.1.

Collected data clearly shows that the bandwidth consumption is not uniform but highly variable and it is possible to observe periodic oscillations with a period of 24 hours. The reason of this particular bandwidth utilization is due to the fact that during the day the largest part of hotel guests is not in the building, but in the evening, as they come back, a larger number of people in the hotel activate Internet connections. This behavior leads to two relevant considerations: i) hotels need high bandwidth Internet connection in order to handle the heavy traffic peaks; ii) during a large portion of the day the Internet connection is underutilized. This is the main evidence that this specific scenario would greatly benefit from an intelligent system of content management, able to move contents when the connection is not used by guests, in order to reduce traffic when the connection is heavily loaded.

The main goal of this study is to evaluate the effect of the capability to run applications on the network edge. For this reason, one of the key factors, is to estimate the amount of traffic that could be proactively cached (or, in general, managed) by applications running on the Fog Node, scheduling the download or upload, on the basis of the available bandwidth or the interaction with local or remote applications.

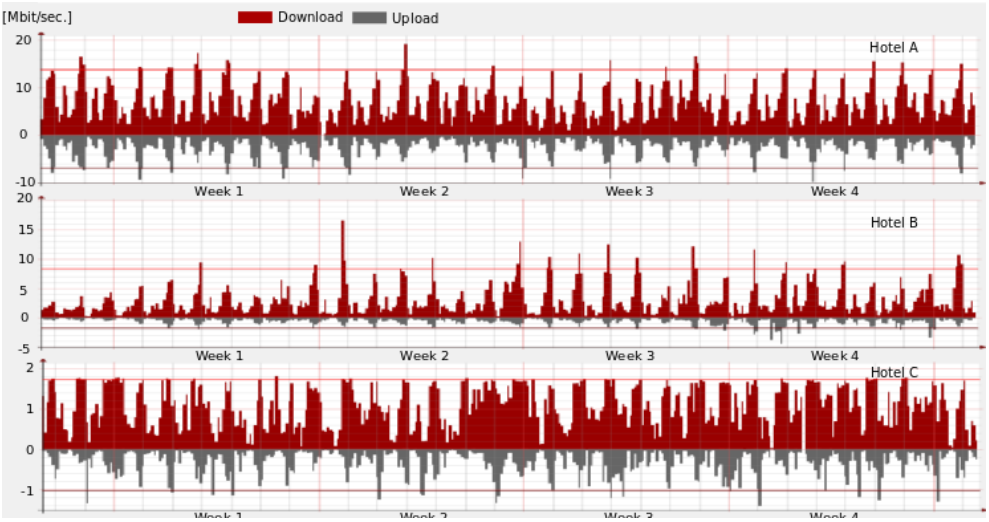



Figure 2.1: Average bandwidth utilization in 1 hour period, measured in hotel A, hotel B and hotel C.

We focus on the traffic that cannot be managed by traditional caching, based on the general idea of multiple access to the same content. We identify the traffic related to content that could be proactively moved on the local node, before the first request for this content has been issued, and sometimes even before the user, interested in this content, arrive at the hotel. We also detect traffic that could be avoided running applications on the local Fog Node. We refer to this traffic with the general term of *Manageable Traffic*. The capabilities to proactively move specific content or to avoid traffic are made possible by the Fog-based design of applications, such as the hotel room booking system or the Wi-Fi AAA service, that can run software modules in the Cloud as well as on the Fog Node, and trigger content download based on specific events. In order to perform the estimation of the Manageable Traffic, we analyzed the download traffic captured during one week, in one location, and then calculate the average over 24 hours in order to present meaningful results for an average day. Since user's data is not collected, it is not possible to associate traffic to users and all the analysis is performed in total respect of the privacy. Each user is associated to

unique anonymous identifier, in order to identify all the connections activated by the same user. The average amount of captured packets was more than 1200000 per day. For our purposes, there is not an immediate or simple method to identify Manageable Traffic by analyzing captured packets; we thus decided to start identifying DNS (Domain Name Server) queries and replies. On the basis of the resolved name, we classified all the traffic from the corresponding IP address as follows.

- *Well-known*: traffic from well known sources that is basically known in advance to be requested by users, such as popular non-real time multimedia contents platforms.
- *Frequent*: we detected specific source IP generating traffic to a large number of different users. In this group, we identify non real-time traffic related to the authentication system, that could be moved to the Fog Node, or to specific events that were occurring in the hotel. We can also envision systems able to automatically detect frequently downloaded content and automatically move it on the Fog Node.
- *User-specific*: large amount of non real-time traffic from a specific source to a specific user, such as multimedia or contents related to entertainment. This traffic could be moved in advance to the Fog Node for a local access, for example as part of the interaction between the fog-based infrastructure and the room booking system.
- *Other*.

The first three categories are considered Manageable Traffic, whereas the last one unmanageable. In the case of traffic from IP addresses not corresponding to any DNS resolution, we contacted directly the IP, whenever possible, in order to classify the traffic as described before. This analysis is complicated because of the presence of traffic from Content Delivery Network (CDN) platforms, such as Akamai or Amazon, that expose their Domain Name hiding the real content and making impossible to classify the traffic. We approached this analysis in a conservative way considering all the unidentified traffic as unmanageable. In order to manage automatically the

highly variable amount of captured traffic, we decided to organize the obtained data in blocks including the same amount of captured traffic. Every block includes about 1.3 GB and it has different time extensions depending on the time needed to collect the 1.3 GB.

Table 2.2 shows the percentage of total traffic, identified as Manageable in every part of the day, classified for every category, total and per category. This analysis shows that an average amount corresponding to 28.89% of the total traffic, could be managed in a smart way by a Fog-based infrastructure, in order to optimize the available resources.

We also noted that the main part of this Manageable Traffic, is downloaded from well-known sites related to entertainment, mainly movies, that cause high bandwidth consumption during the evening. Another considerable part of the Manageable Traffic is received from the Cloud-based AAA system, that could be designed in order to leverage the availability of local resources provided by the Fog Node. Traffic related to specific events is also a relevant part of the Manageable Traffic, and this also could be pre-loaded on the Fog Node when the bandwidth is available. We observed also a considerable amount of traffic related to specific well-known Cloud services or platforms such as Facebook, Skype or Microsoft Office 365. We did not consider this traffic as manageable, because part of these applications are real-time, encrypted, or, in general, out of control. But this traffic could be managed in collaboration with the service provider, leveraging the nature of Fog Node that could host third parties virtual machines. With this approach, all the entities that want to provide services could produce software agents able to run on the fog node in order to optimize the resources usage, mainly in terms of available bandwidth. We estimated this traffic, potentially manageable, as 29.74% of the traffic categorized as non-Manageable by the previous analysis. The manageable traffic could rise, thus, up to 50% of the total traffic.

Figure 2.2 shows the non-Manageable and the Manageable traffic, divided in classes, as part of the total traffic, distributed during the day, with the same approach of Table 2.2. Figure 2.3 shows a possible redistribution of the Manageable traffic, in order to improve bandwidth utilization, compared to the actual total traffic distribution dur-

| Day part | Well-known % | Frequent % | User Specific % | Total % |
|---|---|---|---|---|
| Part 0 | 16.63 | 10.81 | 3.52 | 30.97 |
| Part 1 | 27.24 | 7.35 | 0.647 | 35.24 |
| Part 2 | 11.41 | 11.21 | 0.77 | 23.41 |
| Part 3 | 5.05 | 12.48 | 8.76 | 26.30 |
| Part 4 | 7.97 | 7.20 | 7.44 | 22.63 |
| Part 5 | 19.61 | 7.95 | 0.96 | 28.53 |
| Part 6 | 14.02 | 7.76 | 7.77 | 29.57 |
| Part 7 | 11.39 | 6.27 | 7.827 | 25.49 |
| Part 8 | 14.30 | 6.48 | 9.59 | 30.38 |
| Part 9 | 10.46 | 5.12 | 7.91 | 23.50 |
| Part 10 | 17.17 | 4.55 | 9.21 | 30.94 |
| Part 11 | 16.43 | 3.97 | 9.82 | 30.23 |
| Part 12 | 12.38 | 3.57 | 9.16 | 25.12 |
| Part 13 | 15.29 | 3.54 | 9.32 | 28.16 |
| Part 14 | 15.67 | 3.45 | 13.36 | 32.49 |
| Part 15 | 17.82 | 6.00 | 9.10 | 32.94 |
| Part 16 | 11.29 | 5.26 | 12.54 | 29.11 |
| Part 17 | 18.21 | 7.09 | 9.76 | 35.07 |
| Day total | 14.62 | 6.67 | 7.59 | 28.89 |

Table 2.2: Percentages of Manageable traffic in an average day.

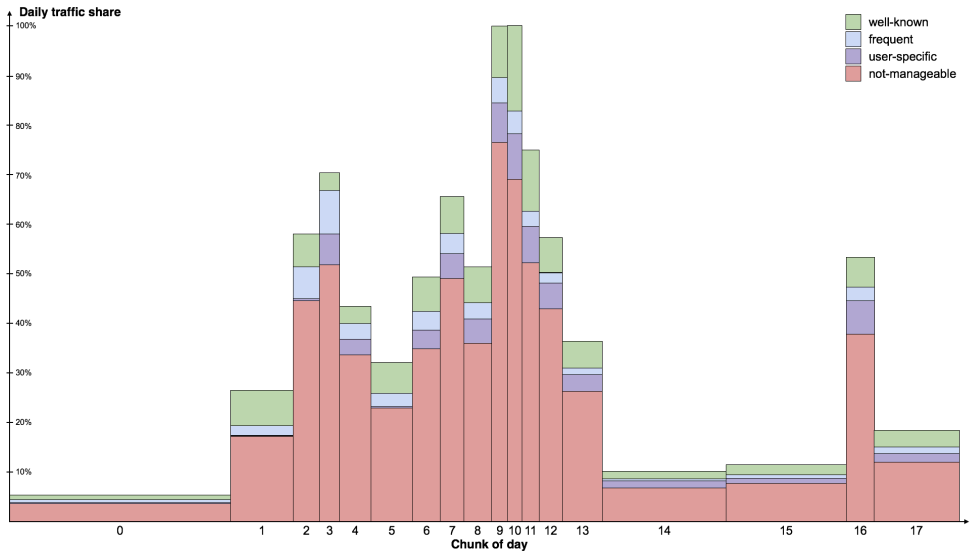



Figure 2.2: Manageable and non-Manageable traffic downloaded during the day

ing the day.

### 2.2.3 Resources Evaluation

Our captured data analysis allows to gather interesting information about the local storage capability, needed on the Fog Node, in order perform content management. Considering an optimal bandwidth usage, i.e., using all the available bandwidth in order to download data to be cached on the Node, we can set a theoretical upper limit of 687 GB for the local storage capacity on Fog Node at Hotel A, 817 GB at Hotel B and 47 GB at Hotel C. Analyzing the captured traffic, as described in Section 2.2.2, we can estimate a more realistic value of 5.1 GB of Manageable Traffic that has to be cached per day. The time of persistence of these data on the storage of the Fog Node depends on the average time of persistence of a specific guest at the hotel, which is usually a known information for a specific hotel.

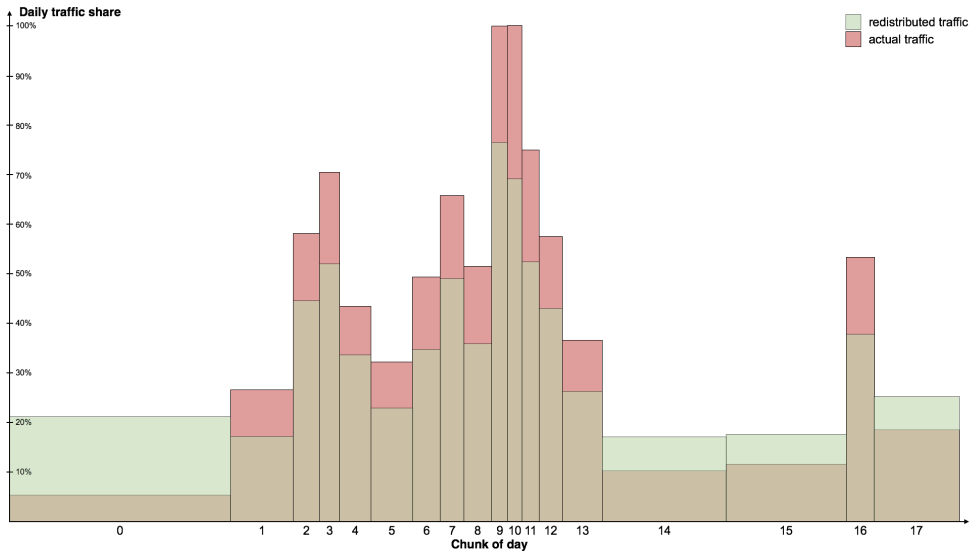



Figure 2.3: Possible downloaded traffic redistribution during the day, compared to the actual total traffic distribution

### 2.2.4 Bandwidth Management

Most hotels have conference or meeting rooms. This leads to violations of the 24-hour periodic pattern of bandwidth consumption, illustrated in Figure 2.1, because of specific events hosted in the hotel during the considered period of observation. Conferences and meetings are particularly critical because of the high density of connected users and the specific needs of the speakers who require reserved bandwidth for presentation or product demo. This situation cannot be easily managed using pre-configured policies or rules based on traffic recognition. We include the management of this highly challenging situation, into the benefit brought by a Fog-based approach, because the envisioned Fog Node is able to dynamically enforce selective bandwidth limitations, to specific connected devices, as a consequence of the interaction with external applications through APIs. With this function, a speaker, simply using an application running on his/her device, such as a smartphone, can ask for reserved bandwidth calling an API exposed by the Fog Node. The same function can be used

by a Smart Object, such as a fire alarm during an emergency, on a shared network. This approach makes Fog-based access networks able to locally manage network policies, as well as computation and storage, like a flexible and shared infrastructure.

## 2.3 Chapter Conclusions

In this dissertation, we are envisioning a new approach to the design and implementation of access networks, based on the deployment of Fog Nodes able to: locally host virtual machines; provide storage resources; create SDN networks; and enforce dynamic and interactive bandwidth management. This architecture is described in Chapter 3.

In order to evaluated the potential benefits, brought by this approach and legitimate this model, we analyzed the traffic collected at three hotels (50,000 connections from over 13,800 guests).We identify that about the 28% of content download could be scheduled or avoided, in order to optimize bandwidth consumption and limit latency, with a proactive approach allowed by Fog-based design of applications such as hotel's rooms booking system or Wi-Fi authentication. This portion could increase beyond 50% if also Cloud-based service providers were able to run agents on the Fog Node. Our analysis also leads to a realistic evaluation of the amount of data that could be cached per day. The interaction between Fog Node and local applications allows a dynamic and flexible management of the available bandwidth, which is expedient to deal with complex situations such as: conferences, with high density of connected users with different needs; and emergencies, when selected Smart Objects (i.e. fire alarms) may need to have priority in shared networks. The collected experimental results show that this new approach to the design of access networks has great potential benefits, in terms of resources optimization and performance.

# Chapter 3

# Internet Access Networks based on Fog Computing, SDN, Containers and APIs

## 3.1 Chapter Introduction

The swift evolution of end-user devices, such as smartphones and tablets, and the widespread adoption of Cloud-based applications have deeply changed the characteristics and requirements of on-line applications, as well as the very nature of the Internet. However, Internet access networks, which are in between these evolving endpoints, are still based on traditional approaches and have failed to support this revolution. In this chapter we introduce the main vision of this dissertation, proposing a novel approach to public Internet access networks based on Fog Computing, Software Defined Networking (SDN), and lightweight virtualization technologies for local deployment of applications in Linux containers. SDN could give access networks the flexibility needed by these new challenges, but the implementation seems really slow and complex, primarily in networks not managed by a single operator and highly diversified such as public Wi-Fi hotspot. The deployment of SDN-based solution outside the data center seems limited to specific operator's cellular networks

or test environments, not involving Wi-Fi access networks in hotels, airports, retail shop, smart cities or even houses.

Just introducing SDN features by replacing the existing networks, is not an effective approach so far, and SDN-based infrastructures on the edge probably needs a new design of the whole architecture of access networks. Fog Computing can provide the new approach to access networks needed by SDN. This concept is based on the idea to deploy Fog Nodes on the edge networks, able to provide computing, storage and networking dynamically, to the applications as an intermediate layer between users and Cloud. Fog Computing and SDN have extremely interesting and powerful interactions, basically because Fog needs SDN in order to offer a flexible and dynamic networking based on the interaction with applications, and SDN needs Fog Nodes to run VNF and virtual topologies, possibly without replacing the existing infrastructure.

A new generation of access networks based on Fog Computing can boost the implementation of SDN-based solution on the edge and provide a flexible and controlled infrastructure for the next generation of services and users. Fog Computing appears to be the right approach to leverage the flexibility of SDN and push the deployment of this technology. The union of Fog and SDN can generate innovative IoT-ready access networks, based on the interaction between the infrastructure and the applications and able to tackle future challenges. In this chapter, we describe the architecture we developed for Internet access networks, that is the main focus of this dissertation, and introduce some experimental results based on a test implementation of this kind of networks. The main goal of the experimental evaluation, is to prove the feasibility of the envisioned approach, for this reason we considered a typical use case, involving Internet access network, in public areas, with users authentication, such as Wi-Fi hotspots. A second feasibility analysis, considering a use case related to services for IoT, is described in the next Chapter.

## 3.2 Architecture

The basic idea is to provide the flexibility, needed in access networks, using the ability to build networking and deploy application as a result of the interaction with external applications using Application Programming Interfaces (APIs). The architecture proposed here can be considered as an evolution of legacy Internet access network, typically deployed in Hotels, retail shops or cities. From the physical point of view, the infrastructure is based on nodes, called *Fog Nodes*, deployed on field. The Fog Node con be a physical appliance or a virtual machine and it is able to provide computing and storage to specific applications, through the activation of containers, and able to deploy SDN-based networks. This infrastructure also supports the activation of Virtual Machines (VM) on the Fog Node and the capability to connect VM to the mentioned SDN network, but we are more focused on containers because of their lightweight more useful to fast deployment on edge nodes.

Physical access devices, such as access points or switches, can be integrated in Fog Nodes or deployed using dedicated hardware and connected to the Fog Nodes. All connections are performed through either switched or routed network. In order to enable the interaction with external applications, Fog Nodes are exposing REST APIs using standard technologies such as HTTP or HTTPS. THe envisioned platform includes two other elements. The first element are functions deployed in Cloud, such as centralized AAA services, analytics, statistics and monitoring. Basically we deploy in Cloud, non-real time applications, or in general without tight limitations on latency, or related to global functionalities, such as authentication roaming, statistics, analytics or Big Data. The second element is a Software Development Kit (SDK) for mobile applications (App).

The interaction with external application is a key point of the infrastructure, because the flexibility needed by future services and expected by future mobile Internet users, can not be achieved with pre-set configuration or policies based on traffic recognition. In order to exploit the potential of Fog Computing at its fullest, we have developed the capability to trigger and control the deployment of full virtual environments, including networking and applications, directly at the edge of access

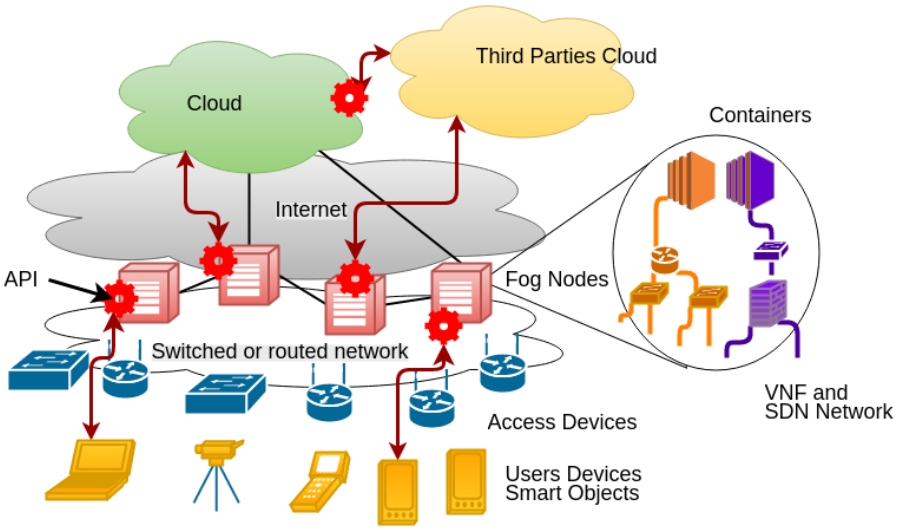



Figure 3.1: Representation of the proposed architecture for Internet access networks.

networks. The entire infrastructure, including virtual connections, Virtual Network Functions (VNFs) and containers, is deployed and managed automatically as a result of the interaction with external applications through Application Programming Interfaces (APIs). This interaction can occur with external Cloud-based applications or in general with any kind of application. The interaction with applications running on devices connected to the Access Network, appears a very interesting and powerful case. For this reason we developed the SDK for mobile App. The SDK, once embedded in the mobile App, makes the App, running on connected devices like spartphones, able to interact with the network infrastructure and control the deployment of environment including VMs, containers and the SDN-based networking. The client device or in general the network user become an active part in the control of the Network. Figure 4.3 shows the described architecture.

This approach allows external applications to call APIs in order to trigger and control the deployment, the modification and the destruction of virtual dedicated infrastructure on the access network, connecting end devices, smart objects, containers

running on the Fog Node, and the Internet. External applications could be running in the Cloud or even on end devices: by calling proper APIs, they can provide a description of the network topology and containers that have to be created on the Fog Node. As consequence of the described interaction between external application and access network, it is possible to:

- build and deploy applications as containers;
- create and deploy network topologies with specific VNF, in order to connect selected clients, including smart objects, and dedicated applications;
- enforce specific QoS policies.

The envisioned platform is able to connect clients with applications running directly on the access network, implementing a very flexible infrastructure for a new generation of services, compared to the traditional approach where applications are running in the Cloud and the access network is basically used to provide Internet connectivity. The novelty of the proposed approach are the following:

- usage of technologies developed for different fields, such as data center (SDN) and IoT (Fog Computing), in Internet access network interconnecting human users and smart objects;
- considering the access network as a flat virtualized environment where to build the needed infrastructure in real-time, including Virtual Machines, Linux Containers and SDN networks;
- giving the control of the deployment of virtual infrastructure to external application and in particular to the clients of the platform, through mobile App running on connected devices. This approach turns upside down the more common vision of a centralized control layer above the infrastructure, introducing the idea of clients able to interact with the infrastructure directly, controlling the deployment of virtual environment on the Edge of the network, close to them.

The main advantages brought by the proposed approach can be summarized as follows: i) flexibility, as needed by the new generation of services; ii) security improvement, especially in terms of the capability to isolating traffic creating dynamically different networks for different clients; iii) simplification of new service deployment, turning on containers and creating networking just as a result of the call to specific API; iv) capability to use the existing access devices and a simplified flat physical connectivity to implement more complex logical topologies. Leveraging the flexibility introduced by this platform, a new generation of services, which could be very challenging for a traditional approach, can be introduced. We now provide a few examples to highlight the foreseen advantages. Deploying this infrastructure on trains, planes or cruise ships, where constant and reliable connectivity to the Internet is a critical issue, it is possible to interact with the booking system in order to transfer to on-board Fog Nodes in advance, when the connectivity is good (such as at the airport or in stations or harbors), dedicated applications or contents for every single passenger. It is also possible to create dedicated networks connecting each passenger, or a group of passengers, with their applications or files. In hotels, it is possible to create dedicated network for each room in order to connect guests' devices and smart objects in the room, as a consequence of the check-in process. This will allow guests to have Internet access but also to control their room's door, cooling/heating system, TV, etc, for example with a mobile App, in an isolated network. Exposed APIs can also be used directly by smart objects or smart systems, such as security cameras or alarms. Any security application can be installed on Fog Nodes and a dedicated network can be created for every service. Camera or security systems can call APIs in order to have assigned traffic priority in the case of an alarm. In general, it is possible to dynamically deploy networks for IoT services, connecting end users' devices and smart objects. This approach can also unlock the market of third parties applications to be installed on Fog Nodes, with the same approach used on Google Play Store or Apple App Store.

We are mainly focused on wireless access networks in public areas, such as retail shops, hotels or smart cities. For this reason, we consider the implementation of virtual infrastructures, on Fog Nodes, automatically triggered and controlled by

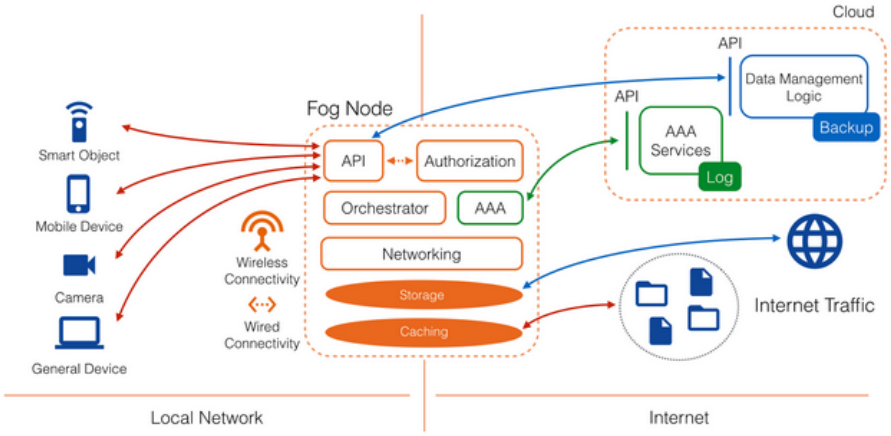



Figure 3.2: Specific Fog-based architecture for proactive caching described in Chapter 2, not including SDN-based network, VMs and containers deployed on the node. Data flow for dynamic content transfer is shown in blue, while local caching traffic flow in red.

the authentication process. From this point of view, AAA services are the external applications able to trigger and control the automatic implementation of virtual environments as described in users' profiles, including networking and applications. This platform also supports, in a very effective way, all the services where the Cloud platform is not necessary, such as the management of lights, heating or cooling systems, and it is able to deploy procative caching services evaluated in Chapter 2, leveraging the capability to dynamically move data in order to make them locally available, both for access and upload, introducing an important intermediate point of control, to the more common interaction between end devices and the Cloud platforms. In this case the Fog Node is also able to perform AAA functions, like any hotspot access controller. A simplified Fog-based architecture, not including SDN and virtualization, able to implement the proactive caching, described in Chapter 2, is shown in Figure 3.2.

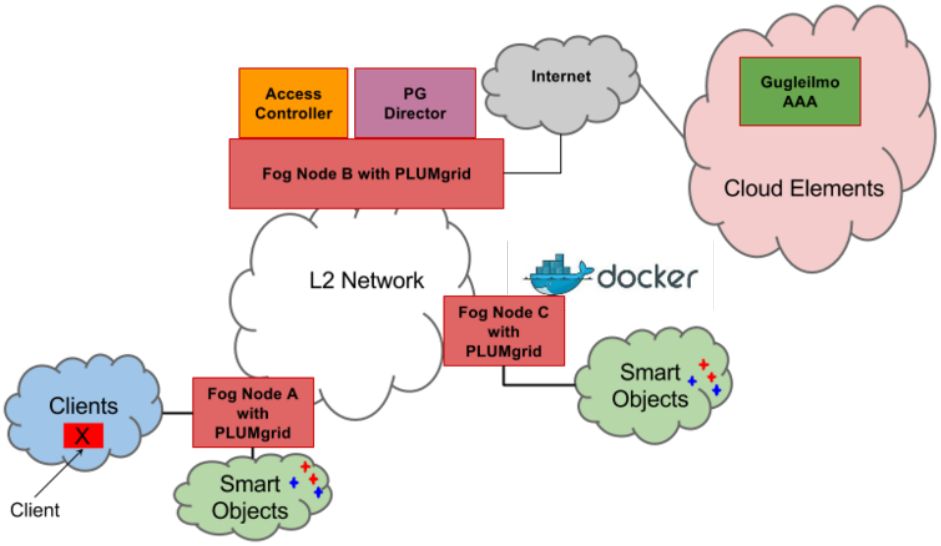



Figure 3.3: Scheme of the lab architecture.

## 3.3 Lab Description

In order to test the capabilities of the described infrastructure and measure the impact on common operation performed using a traditional approach, we deployed a test infrastructure in our laboratory. In particular, we implemented an infrastructure based on three Fog Nodes connected over a flat Layer 2 (L2) network, as shown in Figure 3.3. These nodes indicate three different access locations, with the Fog Nodes having SDN capabilities provided by our industrial partner in this evaluation, PLUMgrid [37]. PLUMgrid core product, ONS, provides a intra and inter-Cloud SDN Platform that is extensible, high performance, with security and micro-segmentation as the building blocks of its control infrastructure. While PLUMgrid has deep integration with OpenStack Cloud infrastructure [38], it is architected to run over any distributed infrastructure. While we implemented as L2, the PLUMgrid technology uses a VXLAN overlay and works seamlessly over any Layer 3 (L3) connectivity.

PLUMgrid, as an SDN solution has its control plane (CP) represented by a Di-

rector cluster. This cluster can be stationed in any of the Fog Nodes, or even a remote Cloud. PLUMgrid CP is accessible to create, deploy, modify, update, and delete virtual network topologies through RESTful APIs. In our case, the Director exists as a Linux container (LXC) running on one of the Fog Node. The data plane of PLUMgrid ONS exists in the form of a IOVisor [39] [40] module installed on every Fog Node that makes the data-plane of the SDN connectivity overlay programmable rather than just configurable through flow-tables. Thus, PLUMgrid DP consists of a network of programmable network functions, like DNS, DHCP, NAT etc, that exist at each edge in the form of a connected graph created by the PG Director. This distributed approach, i.e., pushing topologies to each Fog Node, significantly reduces CP packet punting and enables our deployment to remain scalable even across wide area deployments.

Fog Nodes also provide the computing and storage resources to specific applications used by connected clients and dynamically deployed using linux containers based on the Docker platform [23], to provide containerized micro-service solutions. The infrastructure includes an authentication system, based on RADIUS, that is the actual platform of one of our partner, namely Caligoo, which is a technology provider for one of the largest Wireless Internet service providers in Europe. This platform is able to provide AAA functions and consists of a centralized system, running in Cloud, and a node, acting as Access Controller for the clients and RADIUS client, running on-site. In our implementation, this node is basically a gateway used to manage Wi-Fi hotspots, running as Virtual Machine on a Fog Node and modified in order to make it able to call API, exposed by PLUMgrid Director, and to control activation and destruction of containers on the Docker engine running on Fog Nodes. Clients are simulated by Linux based hosts, are connected directly to Fog Nodes through layer 2 bridges. We also introduced Smart Objects, simulated by Linux hosts, directly connected to Fog Nodes through bridges as well. The whole infrastructure is connected to the Internet and is deployed using virtual machines on a Linux KVM (Kernel-based Virtual Machine) hypervisor, so that L2 connectivity is provided using Linux bridges.

The key point of this approach is the idea to perform an interaction between the

infrastructure and the authentication process in order to build a specific network for each client and connect to this network specific application dynamically deployed as result of the authentication process. The final result is that once a specific user is authenticated, we deploy on each Fog Node a specific environment , including networking and applications for the specific user, according to the user's profile provided by the authentication system.

## 3.4 Evaluation

### 3.4.1 Test Description

In this section we describe the sequence of operation executed during a single execution of the testing process.

1. A client is physically connected to Fog Node A and an SDN-based network is deployed. Therefore, from the logical point of view the client is connected to the Access controller through a switch implemented as a PLUMgrid VNF. Figure 3.4 shows this virtual network, in this starting configuration.

2. Like in any Internet public access network, the client tries to reach the Internet and triggers the authentication procedure, based on the redirection to a login page through a captive portal, with username and password verified by a remote RADIUS server. Once this process has successfully completed, the client is redirected to a welcome page on the Internet and the Access Controller knows the client's profile by reading the attributes in the RADIUS response from the server.

3. On the basis of the client's profile and an internal database, the Access Controller determines the network topology, the network functions and the applications that have to be connected to the specific user.

4. The Access Controller sends instructions to the Fog Node, where a Docker Engine has been installed, in order to activate a Docker container. The type of container and the application running in it, are described in user's profile and

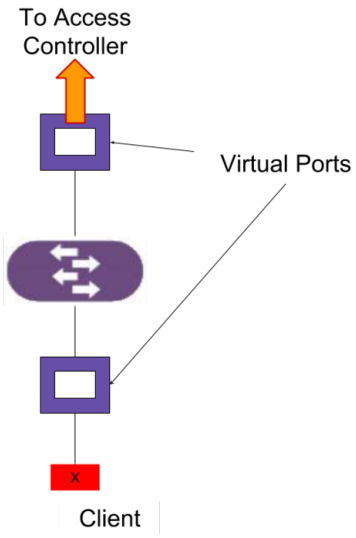



Figure 3.4: Starting configuration of the virtual network

can be different for every user. These instructions could include a Dockerfile, with all the instructions required to assemble a container from scratch, or just the command to launch a script on the Fog Node to turn on a container already existing on the node. Instructions also include information about the virtual interface associated with the containers.

5. The Fog Node with Docker services receives the instructions and turns on the Docker container.

6. The Access Controller builds a description of the needed network in a JSON format and uses the PLUMgrid APIs to deploy the network.

7. The PLUMgrid Director deploys on each Fog Node, the network, defined by the Access Controller, interacting with the PLUMgrid API. At this point, the network, the client is connected to, has changed: it includes other VNFs, link to Smart Objects, physically connected to Fog Node, as shown in Figure 3.3, and it includes a virtual interface, ready to be connected to the Docker container.

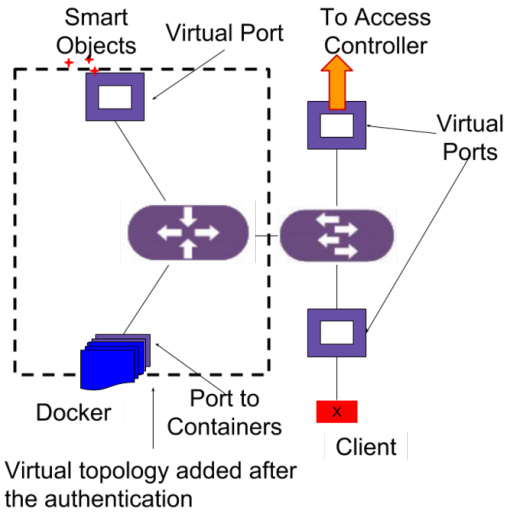



Figure 3.5: Final configuration of the virtual network

8. The Access Controller launches a script on the Docker-enabled Fog Node, ordering the PLUMgrid module to connect the Docker container, linking container's interface to the virtual network.

An example of final network topology is shown in Figure 3.5, where the client is connected to a dedicated network infrastructure that provides connectivity to the Internet, to Smart Objects and to a specific application, dedicated and specifically created for this client, running in a container. All the dedicated infrastructure can be destroyed as a consequence of the logout process of the client.

### 3.4.2 Experimental Data Analysis

The purpose of this test is to prove the feasibility of the envisioned architecture, collecting data in order to verify if the deployment of dedicated virtual infrastructure introduces delays or, more generally, affects the quality of experience of Wi-Fi users. We performed 500 authentication procedures, measuring the time interval between the starting instant of the authentication process and the redirection of the users to

a welcome page, after successful authentication, without the deployment of any virtual infrastructure; this is exactly what happens now in a common Wi-Fi hotspot. We also performed 500 authentication procedures where a specific virtual infrastructure, including network and containers, is deployed after a successful login, measuring the time interval between the starting instant of the authentication process and the ending instant of infrastructure deployment. In this set of measurements, we considered the implementation of a virtual network infrastructure that actually causes the interruption of connectivity between two existing containers and the user. Then, we performed 500 authentication procedures where a different virtual infrastructure is deployed, measuring the time interval including the infrastructure deployment. In this case, the network infrastructure actually includes the activation of containers and the networking needed in order to connect them to the user and Linux hosts, representing Smart Objects, such as sensors or actuators, in our deployment .

In all three cases, the measured time interval starts in correspondence to the authentication starts and ends, respectively: in the first case, when the user is redirected to a welcome page; in the second case, when the connectivity between containers is lost; in the third case, when the connectivity between containers and user is up. In order to test connectivity we considered the transmission of a ping command, with timestamp, every 0.1 sec, for this reason, 0.1 sec is also the maximum error in our measurements.

The distributions of the obtained time results are shown in Figure 3.6, whereas the average durations and standard deviations are shown in Table 3.1.

The authentication time is not always the same because some elements have variable effects, such as: the latency to the Cloud-based authentication system, the actual workload on the Fog Nodes. However, in our implementation these effects are very limited and, for this reason, the standard deviation is small, namely approximately equal to 0.75 sec. Data shows that the implementation of virtual topology increases the average duration of the authentication process, but without altering the distribution in time. This means that the process of building a dedicated environment for the user, is stable and predictable, without introducing additional variability to the authentication process. The interruption of connectivity appears faster than deploying

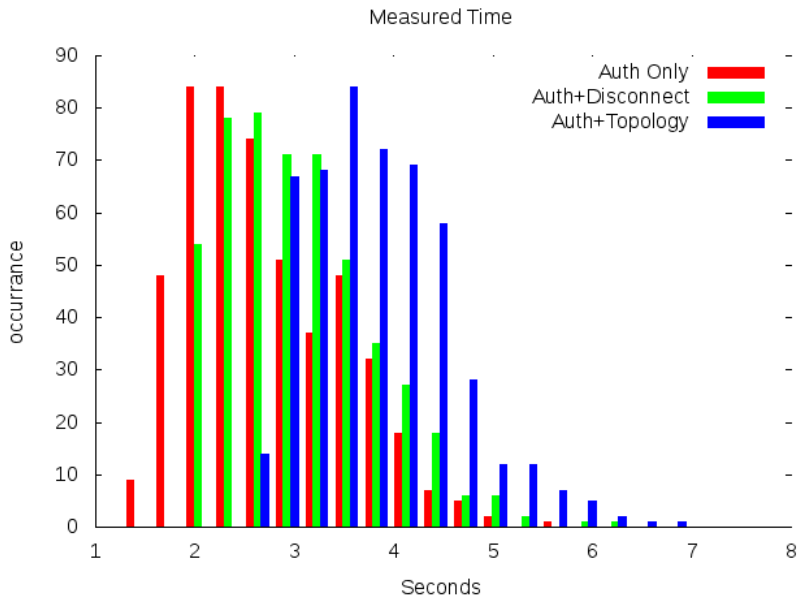



Figure 3.6: Duration distribution of the following operations: (i) authentication only, (ii) authentication and disconnection of virtual network and (iii) authentication and deployment of virtual infrastructure including SDN-networking and containers

a virtual network in order to connect hosts, because as soon as the virtual topology starts to be modified, the connectivity is immediately lost, while a full topology has to be completely implemented, including the population of routing and switching tables of VNF, in order to have full connectivity.

Collected data shows that the implementation, on a locally deployed Fog Node, of a virtual environment, including application deployment in form of containers and a virtual SDN network, adds about 1.12 sec to the average duration of the user authentication process on a “classical” Wi-Fi Internet access network, while the modification of an existing topology, removing connectivity, adds less than 0.33 sec. Considering that, according to [41], the average loading time of a web page depends on several factors but is rarely shorter than 2 sec, we can say that the implementation of a virtual environment, directly on the access network, does not affect the user experience since

| | Auth. Only | Auth.+ Disc. | Auth.+ Infrastr.. |
|---|---|---|---|
| Average [sec.] | 2.64 | 2.96 | 3.76 |
| Standard Dev. [sec.] | 0.76 | 0.75 | 0.74 |

Table 3.1: Average duration and standard deviation of the distribution of durations of the following operations: (i) authentication only; (ii) authentication and disconnection of virtual network; and (iii) authentication and deployment of virtual infrastructure including SDN-networking and containers.

the network topology and applications will be ready before a first web page is fully loaded by the user.

Collected data are not the result of performance measurement for specific platform, such as Docker Engine or PLUMgrid solution, that can achieve much shorter deployment time, but it appears extremely useful to evaluate the possibility to introduce new services based on Fog Computing. Collected results, show that, even using nodes with limited computing resources, the deployment of an articulated Fog Computing solution, able to provide a new generation of services, based on the capability to build tailored virtual network environment for the users, has a very low and predictable impact on common user experience, such as the connection to a public Wi-Fi network. The obtained results depend on the type of containers activated and, in particular, if the container's image is already on the node or has to be downloaded from the Internet. In our test implementation, containers are simple hosts, able to execute and reply to ping command, and they are launched from images already present on the node.

## 3.5 Chapter Conclusions

In order to exploit the potential of Fog Computing, we have developed a platform for Wi-Fi Internet access networks, able to deploy fully virtual dedicated environments for connected users, as a consequence of successful authentication. This environment,

created on Fog Nodes connected to the access network, includes Linux containers and SDN networking. We performed more than 1500 tests and the collected data show that the automatic deployment of virtual infrastructure does not affect the user experience. Considering that we based our testbed, on the evolution of a platform actually used on field, for managing Wi-Fi Internet access networks on field, this result supports the feasibility of Fog-based access networks and the idea that Fog Computing can play a key role in radical changes needed by access network in order to support the revolution of mobility and IoT.

# Chapter 4

# Feasibility Analysis: Managing the Connection of Heterogeneous Smart Objects through Fog Nodes

## 4.1 Chapter Introduction

In previous chapters we validated the benefit brought by a Fog-based approach to Internet access networks, then we introduced a new approach to access networks, describing and deploying in a lab, the full platform based on Fog Computing, SDN and Containers and exposing APIs, in order to interact with external applications. In this chapter we investigate the feasibility of this approach, considering a specific application, of the envisioned platform, in the traditional field of Fog Computing, which is the IoT.

With more than 50 billion devices by 2020, the Internet of Things (IoT) is expected to have a huge impact by connecting people with everyday objects. The advent of this pervasive worldwide network of unprecedented size will represent a groundbreaking factor in the development of innovative applications and in the redefinition of how people interact with their surroundings. The IoT will be, by nature, characterized by the interactions of extremely heterogeneous devices. Constrained devices,

in terms of computational power and available memory, denoted as "smart objects" (SOs), will represent the majority of IoT devices. These devices will likely be battery-powered, thus imposing even more constraints related to energy consumption. From a communication standpoint, energy consumption can be minimized by adopting of low-power communication protocols, such as low power Wi-Fi, IEEE 802.15.4, and Bluetooth Low-Energy. Low-power and Lossy Networks (LLNs) will therefore be the typical environment for SOs.

As many options are available for low-power communications, interoperability among IoT devices will be provided through the use of the Internet Protocol (IP), which will also guarantee integration with the Internet. LLNs are typically accessed by external hosts through *border router* nodes, which are equipped with a network interface with the LLN and others with different network type (e.g., Wi-Fi or Ethernet). Border router nodes will therefore enable IP communication with objects in constrained networks, typically by acting as network coordinators for the LLN (e.g., as root of a IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [42] tree). Due to this important role, it is important that these nodes are not limited so that they do not represent bottlenecks for the architecture. In our vision, border routes are running on Fog Nodes, leveraging the benefit of low latency links to SOs.

The issue of bringing IP to constrained devices has been addressed by standardization organizations, such as the Internet Engineering Task Force (IETF), and research projects, and has led to the definition of the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) protocol [43]. This activity has also resulted in the definition of a specialized application-layer protocol – the Constrained Application Protocol (CoAP) [44], specifically targeted to constrained devices, which cannot afford the high overhead of HTTP. CoAP is a lightweight, UDP-based, binary protocol designed to bring the REpresentational State Transfer (REST) paradigm to the IoT and to be used as the reference protocol for the evolution of the Web of Things (WoT). According to the request/response model of the REST paradigm, SOs expose their resources using CoAP.

As traditional Internet hosts, such as smartphones, are going to interact with the IoT, CoAP has been designed in order to map easily with HTTP for integration with

the Internet. The extremely high number of IoT devices, together with their heterogeneity, will require proper mechanisms for the management and seamless interaction with SOs.

Although the adoption of IP would enable to address and interact with SOs directly, or through Cloud-based application, these approaches may not always be the best fit. For instance, Cloud-based interaction could not meet the applications requirements about latency or real-time, or SOs may implement duty-cycling mechanisms in order to extend their battery lifetime, thus making it impossible to reach them at any time. Moreover, due to their limited capabilities, SOs may not be able to handle large numbers of concurrent requests. This may lead to potential service disruptions or make them unavailable through Denial-of-Service (DoS) attacks. In other cases, extremely limited devices might not even be able to act as servers providing resources, but would rather act as clients delegating other hosts to maintain resources on their behalf.

In order to cope with these potential drawbacks, the introduction of an intermediate network element, which integrates application-layer functions and border router functions, might introduce several advantages. In this chapter we introduce a specialized application, responsible for the management of heterogeneous SOs in complex IoT scenarios, denoted as "*IoT Hub*". The Iot Hub is hosted on the Fog Node, running as VM or container, and integrates several functionalities with the purpose to overcome some of the problems discussed above and enable a truly seamless interaction with SOs, from addressing/routing and application-oriented perspectives. The IoT Hub can also limit the processing load on SOs and moves some of the processing load at the edge of the network. IoT is the traditional field of application of Fog Computing, the adoption of the envisioned platform, introduces two main elements of innovation. The first is the possibility to leverage the virtualization capability of the Fog Node, in order to host the IoT Hub on the node, without needing to deploy dedicated hardware, just like any other application. The second element of innovation is related to the envisioned possibility of interaction between the Fog and the Cloud, which is an important element of our vision and make us able to introduce the concept of *virtual IoT Hub replica*: a Cloud-based "entity" replicating all the functions

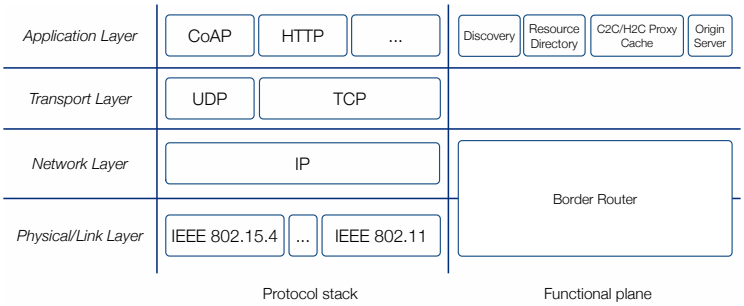



Figure 4.1: Protocol stack and functional modules implemented by IoT Hub.

of a physical IoT Hub, which external clients will query to access resources.

IoT Hub replicas are constantly synchronized with the physical IoT Hub through a low-overhead protocol based on Message Queue Telemetry Transport (MQTT) and they are able to protect smart objects (which cannot, because of their constrained nature, serve a large number of concurrent requests) and the IoT Hub (which serves as a gateway to the constrained network). The IoT Hub is the application we use, in this experiment, in order to verify the capability of leveraging an intermediate Fog layer, evaluate the resources needed on the edge of the network and investigate the interaction between Fog and Cloud on a real application. This analysis is mainly related to the investigation of the feasibility of the envisioned infrastructure of Internet access networks and to its capability to support new ways to design services.

## 4.2 IoT Hub Architecture

The proposed IoT Hub runs on a Fog Node placed at the edge of multiple physical networks with the goal of creating an IP-based IoT network to be used as an infrastructure for the deployment of IoT applications. As opposite to SOs, the IoT Hub runs on the Fog Node that does not present strong limitations on the computational power and available memory, nor strict requirements on energy consumption. The IoT Hub plays a fundamental role by implementing the following functions at different layers of the protocol stack, as shown in Fig. 4.1:

- *Border Router*: the IoT Hub is the gateway/bridge between one or more constrained networks (e.g., IEEE 802.15.4);
- *Service and Resource Discovery*: the IoT Hub is able to discover which SOs are available in the network and, subsequently, to discover the resources that they host;
- *Resource Directory (RD)*: the IoT Hub maintains a list of all CoAP resources available in the constrained networks;
- *Origin Server (OS)*: it provides a CoAP server where resources are hosted or are to be created;
- *CoAP-to-CoAP (C2C) Proxy*: it provides proxying capabilities for CoAP requests coming from external clients that should reach internal constrained nodes;
- *HTTP-to-CoAP (H2C) Proxy*: it provides HTTP-to-CoAP cross-proxying (i.e., protocol translation) in order to enable access to CoAP resources by HTTP clients;
- *Cache*: in order to avoid unnecessary load on SOs and to minimize latencies, a cache is kept with the representation of most recently accessed resources.
- *Replica Manager*: a software module responsible for the coordination and the synchronization between the IoT Hub and its replicas.

### 4.2.1 Link-layer Functions

The border router function implemented by the IoT Hub are two-fold. On the one hand, the IoT Hub provides intra-network IP routing, for instance by acting as coordinator of a RPL-based IEEE 802.15.4 multi-hop network. On the other hand, it provides a single addressing space for multiple networks it is attached to, by acting as a bridge among the networks. The Link-layer functions result in the creation of an all-IP infrastructure, which includes SOs that belong to different physical networks, as shown in Fig. 4.2.

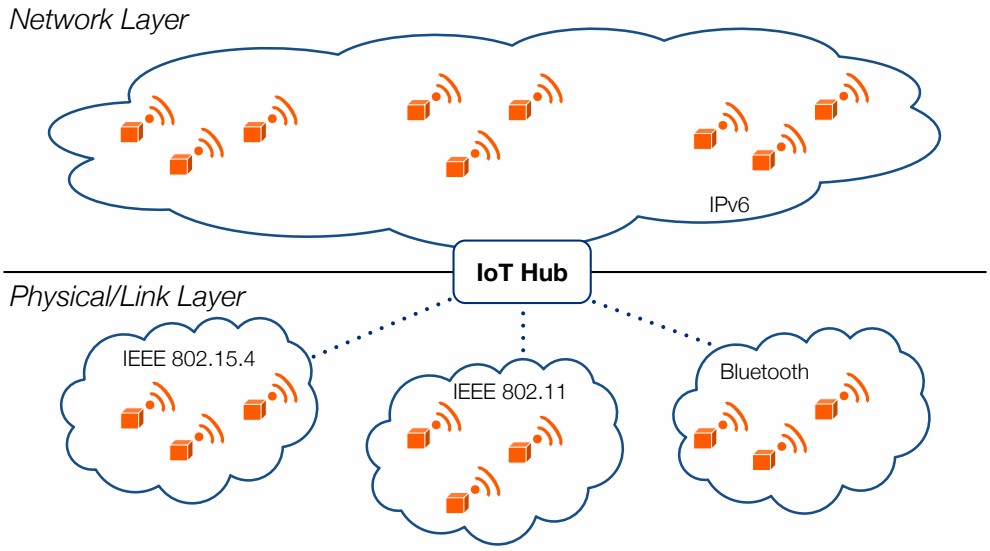



Figure 4.2: The border router function of the IoT Hub has the goal of creating a single all-IP network bridging different physical networks.

### 4.2.2 Application-layer Functions

The IoT Hub plays a significant role in the management of SO-hosted resources, leveraging its privileged position, running on a Fog Node with low latency to SOs and clients. Moreover, the IoT Hub can be used by external clients to discover and access SOs without requiring any a-priori knowledge of their details (e.g., communication protocol and IP address). A high-level architectural view and interactions of the application-layer modules of the IoT Hub are shown in Fig. 4.3. Functions implemented by each module and inter-module interactions are detailed next.

#### Service and Resource Discovery

The IoT Hub implements Service and Resource Discovery in order to constantly have knowledge of the SOs that it is supposed to manage. Service Discovery aims at retrieving the scheme, IP address, and port at which a SO endpoint is reachable. This function is implemented according to multiple mechanisms: i) Zero-configuration

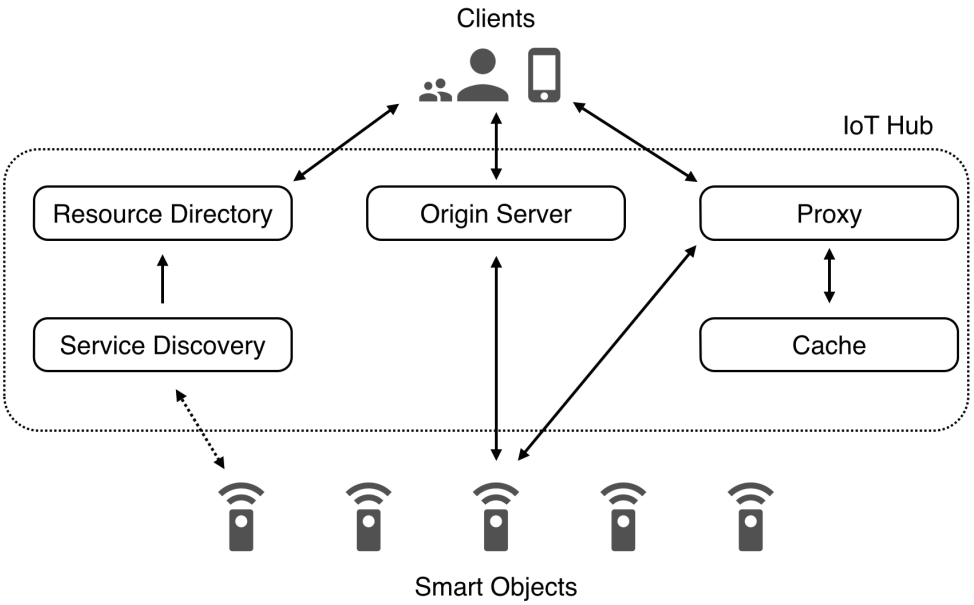



Figure 4.3: Modules and interactions of application-layer modules of the IoT Hub.

mechanisms based on Multicast DNS (mDNS – RFC 6762) [45] and DNS-Based Service Discovery (DNS-SD – RFC 6763) [46]; ii) the use of NMap [1] for network scanning; iii) other mechanisms, such as those described in [47]. Once endpoints have been discovered, they can be requested to retrieve the list of resources that they host. For CoAP-based SOs, the `/.well-known/core` URI can be used for this purpose.

The list of resources is returned in CoRE Link Format (RFC 6690) [48]. For HTTP-based SOs, the list might be returned using Web Linking (RFC 5988) [49] format. Support for proprietary SOs that are not standard-compliant, other mechanisms must be integrated. An example of possible alternative methods is the registration function provided by the Session Initiation Protocol (SIP).Moreover, the IoT Hub advertises its presence in the network using Multicast DNS with the `_coap._udp.local.` service type, in order to let devices discover its presence.

[1] https://nmap.org

### Resource Directory

The Resource Directory (RD) is an specialized element that serves as a registry for all the resources in the overall IP network managed by the IoT Hub. Resources are characterized by a link (i.e., a URI) and a set of attributes that can be used to describe them, as defined in RFC 6690. The RD provides i) a function set for the registration, update; and ii) deletion of resources and a function set for the lookup of resources. The RD can be queried by clients to discover links to resources. Requests can contain query parameters to filter the results. The RD can be accessed using CoAP, as defined in [50]. The Resource Discovery module interacts with the RD: when resources are discovered, they are registered to the RD and then periodically updated. The interaction between the Resource Discovery module and the RD is shown in Fig. 4.4. The Resource Discovery module registers resources on the RD by sending a CoAP POST request with the links to the discovered resources. The RD replies with the URI of the newly created resource, which is then used for any update or deletion operation. The registration is also repeated by reporting links that allow access through the CoAP-to-CoAP proxy and HTTP-to-CoAP proxy, which can be used by clients that need a proxy (with optional protocol translation).

### Origin Server

In some cases, SOs are characterized by very limited capabilities and cannot act as a server and host resources. When this occurs, SOs act as clients and need an external node to maintain resources on their behalf. In order to support this kind of situation, the IoT Hub provides an Origin Server (OS) function. Clients can request the IoT Hub to create and maintain resources on their behalf. In this case, clients also need to take care of the registration of the newly created resource on the RD, since its attributes are unknown to the OS. The IoT Hub is then going to be used to access resources.

### C2C/H2C Proxy and Cache

The IoT Hub can act as an intermediary for communication with SOs. The IoT Hub can act either as a CoAP-to-CoAP proxy (to allow external CoAP clients to access

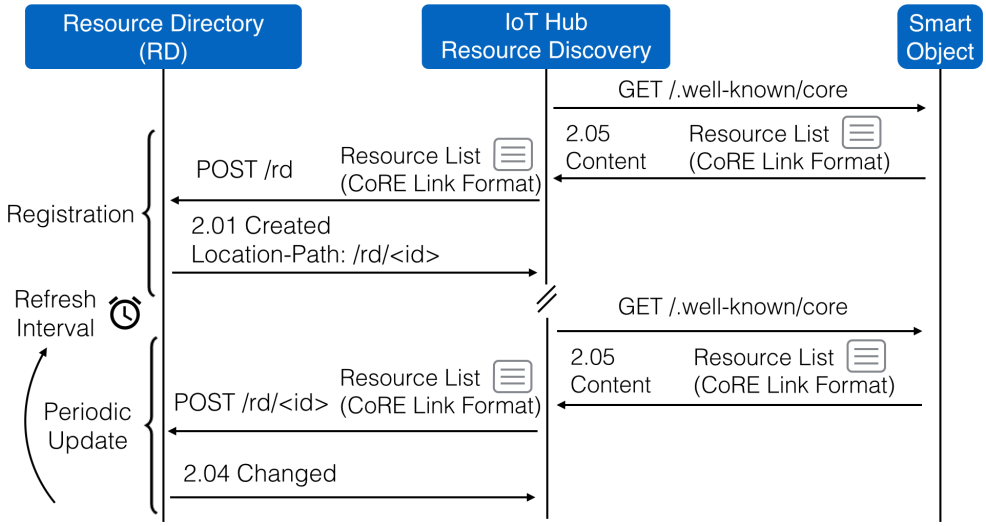



Figure 4.4: CoAP-based interaction between the Resource Discovery module and the Resource Directory for resource registration and update.

constrained resources that cannot be reached directly) [44], or as a HTTP-to-CoAP proxy (to allow HTTP clients to allow access to CoAP resources) [51]. There are several reasons that motivate the presence of a proxy in IoT scenarios: i) to shield the constrained network from the outside, e.g. for security reasons (DoS attacks); ii) to support integration with the existing web through legacy HTTP clients; iii) to ensure high availability on resources through caching; iv) to reduce network load; v) to support data formats that might not be suitable for constrained applications, e.g. XML.

The IoT Hub also implements a cache, which is particularly useful in presence of proxies. In fact, the IoT Hub keeps a “fresh” representation of resources that can be used to serve requests without requiring to forward the request to the SO. This is expedient to avoid unnecessary computation on constrained devices and also to make it possible for clients to get information also in case of duty-cycled devices, which may not be reached at all times.

### 4.2.3 Synchronization Protocol

In order to increase the robustness of this Fog-based architecture, in this experiment we propose a Cloud-supported replication mechanism for IoT Hubs in order to efficiently manage CoAP resources in a scalable and secure way. The proposed mechanism relies on the possibility to exploit the interaction between the Cloud and the Fog platforms, to clone and virtualize IoT Hubs. Replicas are full copies of IoT Hubs, thus implementing all their functionalities, and can thus be accessed on behalf of actual physical nodes. This brings several benefits: (i) a unique Cloud-based interface for accessing resources is exposed; (ii) the actual implementation details of the IoT Hub are hidden from communication, thus protecting the IoT Hub and the smart objects behind it; (iii) the Cloud platform may introduce balancing policies in order to scale up or down the number of replicas according to current needs, depending on the number of incoming/estimated requests and resources to be managed. This approach allow us to study the performance and the characteristics of applications developed on three layers (end devices, Fog and Cloud) and to evaluate the feasibility of the interaction between Fog and Cloud.

The synchronization protocol used in the architecture implements a pub/sub communication model. In fact, communication follows a one-to-many pattern from the IoT Hub to all of its replicas. All messages are sent by the IoT Hub to an MQTT message broker (hosted on the Cloud platform and illustrated in Fig. 4.5 (a)) using specific MQTT topics (which can be used to selectively target one, many, or all replicas in order to implement unicast, multicast, or broadcast communications respectively) that is managed by the Cloud platform, using a VPN connection, for security and addressing reasons, as shown in Fig. 4.5 (a). The IoT Hub and its replicas are all identified by a system-wide identifier, assigned by the designed Cloud platform.

Each IoT Hub includes a Replica Manager (RM), which is a dedicated software module responsible for the synchronization among the IoT Hub and its replicas. The RM is composed by the following items, as shown in Fig. 4.5 (b).

- A *Replica Registry* (RR), which contains the list of the identifiers of all the replicas of the IoT Hub.

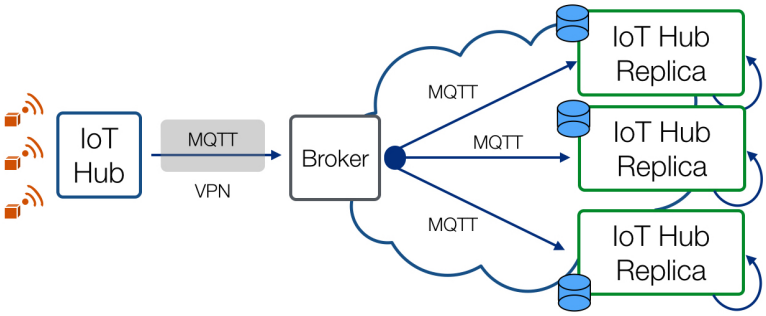
- An *MQTT subscriber*, which registers to the broker to receive messages related to two topics: (i) its own identifier ($id_i$) and (ii) the identifier of the actual IoT Hub ($id_{hub}$); these may coincide in the case of the actual IoT Hub.

- An *MQTT publisher*, which publishes messages to the broker, using the method $pub(t,m)$, where $t$ is the topic and $m$ is the message to be published.

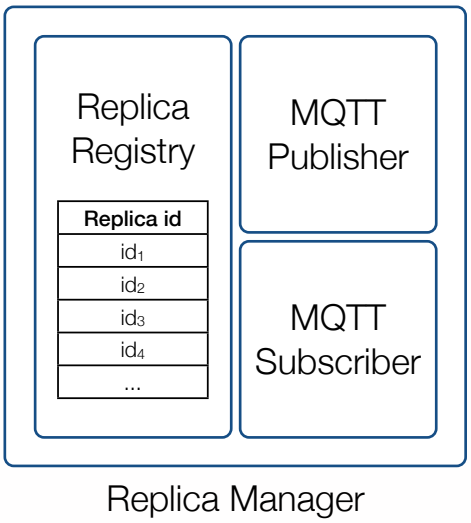
The IoT Hub is in charge of keeping full synchronization of its resources with its replicas, in order to ensure that all requests are served in the same way, regardless of the specific replica that was targeted by the client. Synchronization comes into play every time a resource on the IoT Hub changes. This can be caused by different events.

At startup, a replica of the IoT Hub needs to synchronize with the actual IoT Hub. The procedure is shown in Fig. 4.10. The RM of the replica publishes its $id_i$ to the topic $id_{hub}$, in order to inform of its creation ($pub(id_{hub}, id_i)$). At this point, the IoT Hub updates its RR by adding $id_i$ and then starts publishing to the broker all the resources (R) using the topic $id_i$ ($pub(id_i, R)$), which guarantees that the new replica will receive the resources. When the synchronization procedure has ended, the replica will be automatically kept synchronized with the IoT Hub during the normal system lifecycle.

When resources are polled for, the IoT Hub might find out that a resource targeted by some requests has changed. A request targeting a resource that either has not been cached or is not considered fresh must be forwarded by the IoT Hub to the smart object. Upon receiving the response from the smart object, after updating its cache and forwarding the response to the requesting replica, the IoT Hub uses the synchronization protocol to publish the updated information to all of its replicas. When observing resources, the synchronization procedure resembles the same of the polling case. When resources are pushed by smart objects to the IoT Hub, the IoT Hub uses the synchronization protocol to publish the updated information at all replicas. Note that this synchronization strategy is needed only for those replicas that are part of the request/response loop: in fact, as all resources are automatically synchronized with the request-issuing replica by design, as the replica is perfectly reproducing also the behavior of the actual IoT Hub.




(a)




(b)

Figure 4.5: The broker-based message flow between the IoT Hub and its replicas is shown in (a), while the internal structure of the Replica Registry module of the IoT Hub is shown (b).

## 4.3 Resource access through the Fog

The presence of the IoT Hub on the Fog Node locally deployed, enables clients to effectively discover and access resources hosted by heterogeneous SOs. The added

value of the IoT Hub is its capability to hide completely the diverse nature of SOs, in terms of hardware, wired/wireless communication protocol, and application-layer protocol used, to clients, which can interact with them using uniform interfaces and without requiring any prior configuration. Interaction with SOs through the IoT Hub occurs according to the following steps, as shown in Fig. 4.6:

1. the client discovers the IoT Hub through a suitable mechanism, such as DNS-SD;
2. the client queries the RD of the IoT Hub to get a list of resources that match its interests (resource discovery); the RD returns the list of links in CoRE Link Format;
3. the client selects the resource to access; since each resource has been registered with multiple links (direct, through CoAP-to-CoAP proxy, or through HTTP-to-CoAP proxy), the client can also select to use the URI that best fits its characteristics;
4. the client sends requests to the SO according to the URI selected at the previous step (resource access) and according to the type of interfaces exposed by the resource (e.g., a sensor resource might support only read – GET – operations, while actuators might also support write – POST/PUT – operations), as specified in [52].

Note that the interaction procedure does not make any assumption on the nature of the client nor on that of the SO hosting the resource. This is possible thanks to the mediation of the IoT Hub, which takes care of hiding all the low-level details of communication, thus enabling full and standard interactions among clients and connected SOs.

## 4.4 Experimental Analysis

In order to validate the feasibility of the proposed Fog-based IoT architectural solution and to evaluate its feasibility and performance, an extensive experimentation has

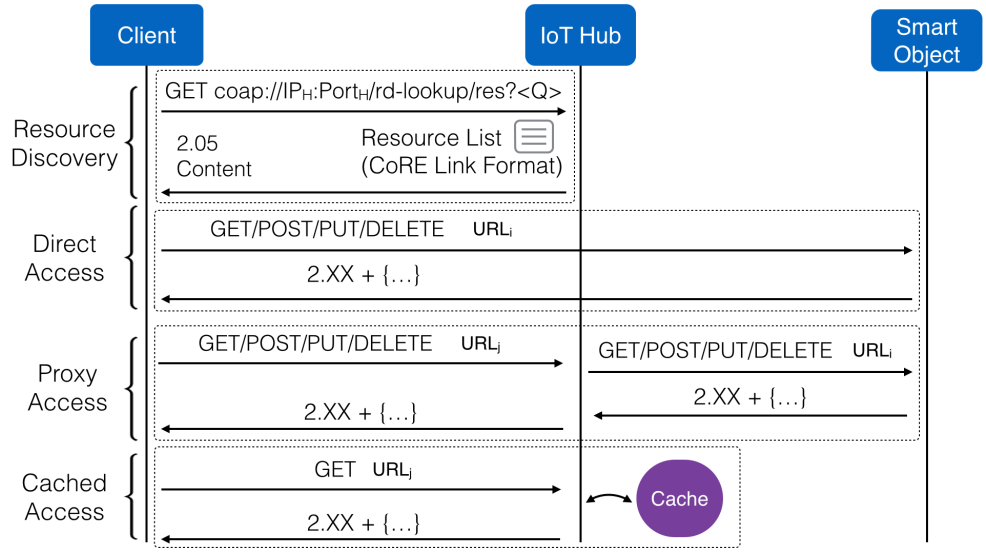


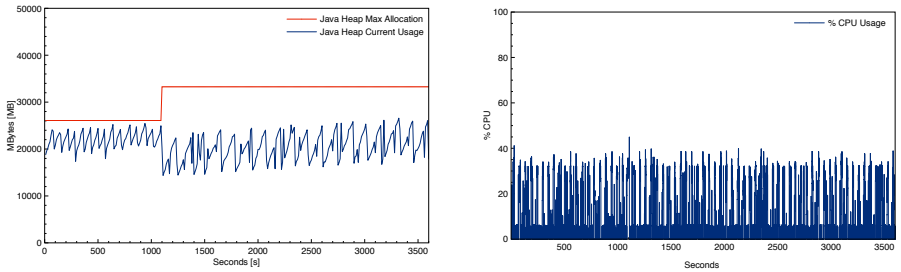

Figure 4.6: Interaction between clients and heterogeneous smart objects with the mediation of the IoT Hub.

been conducted. The evaluation focuses on the resource management on both local and remote IoT Hubs and the synchronization mechanisms described in Section 4.2.

We set up two different environment, the first one, described in Subsection 4.4.1, is dedicated to the evaluation of the resources needed by the IoT Hub, in order to evaluate the possibility to run this element on low-end Fog Nodes with constrained capabilities, similar to the same nodes already used for Wi-Fi Internet public access networks (Hotspots). The second laboratory, described in Subsection 4.4.2, is focused on the evaluation of the replica and synchronization system, which represents a clear example of interaction between Cloud and Fog layers.

### 4.4.1 Implementation and Evaluation of IoT Hub resources

The IoT Hub has been implemented using Californium, a Java-based implementation of CoAP and related drafts [53]. The IoT Hub has been deployed to a Fog Node running on a Raspberry Pi (RPi) Model B single board computer and used to manage




(a) (b)

Figure 4.7: Performance evaluation: (a) Heap memory used (dimension: [MB]); (b) CPU usage (adimensional).

resources hosted by heterogeneous SOs within a real-world IoT testbed within our department.

The IoT testbed consists of several types of nodes which differ in terms of both computational capabilities and radio access interfaces. Nonetheless, the considered nodes can be grouped into two main classes: 35 Constrained IoT nodes and 35 Single Board Computers (SBC) nodes. Constrained IoT nodes are Class-1 devices, according to the terminology introduced in [54], based on the Contiki operating system [55]. SBC nodes are Class-2 devices, according to [54], typically running a Linux operating system and with multiple network interfaces. The test have been conducted in order to verify the compliance with functional requirements. Moreover, the following performance metrics have been used to assess the processing load required by the IoT Hub on a low-end device: i) amount of heap memory used (dimension: [MB]); and ii) percentage of CPU usage. The tests considered the execution of an IoT Hub instance in the testbed, where 100 resources were already deployed over a 1-hour observation period with a 60-second resource refresh interval. The results are shown in Fig. 4.7.

It is possible to see that the amount of heap memory used is extremely low and does not exceed 26 MB. The sawtooth wave-like trend is due to the periodic clean-up procedure performed by the Java Virtual Machine. As for the CPU usage, the results

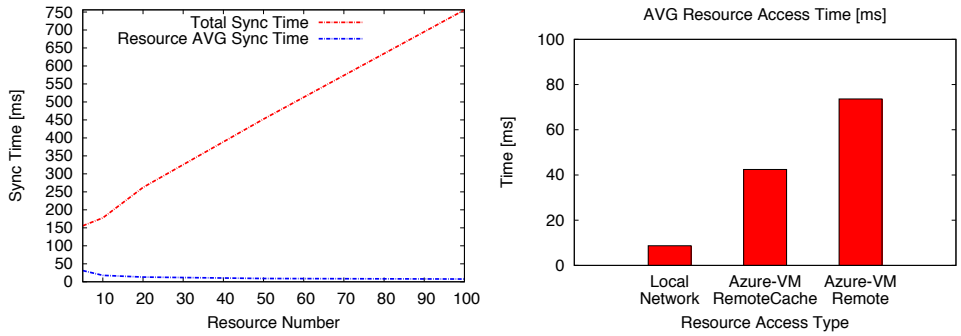
show that the processing load of the IoT Hub is around 5% during normal operation, while peaks at an average of $\sim$35% occur when serving requests from external clients. These values are low as they refer to the limited ARM1176JZFS/700Mhz CPU of the RPi. Moreover, these values correspond to lower delays in accessing resources and significant energy savings on constrained nodes.

### 4.4.2 Experimental Setup for the Evaluation of the Replica and Synchronization System

The experimental setup has been designed and deployed with the aim of creating a realistic scenario with heterogeneous components and nodes in a local IoT network and the Cloud. The main components can be summarized as follows.

- *Smart objects*: Real and virtual nodes with CoAP modules based on the Californium framework [53].
- *IoT Hubs*: Raspberry Pi Model B [56] or independent VM instance running all the functional modules presented in Section 4.2 (Resource Discovery, Proxy CoAP/CoAP and HTTP/CoAP, Border Router, Cache and Replica Manager).
- *Virtualization Platforms*: Four different virtualization configurations on both local and Cloud platforms are considered: (i) Microsoft Azure [57]; (ii) Amazon EC2 [58]; (iii) Open Stack [38] on Microsoft Azure; and (iv) Open Stack on a local physical machine.
- *Resource External Consumer*: Real and virtual external consumers implementing HTTP and CoAP modules to dynamically interact and consume available resources managed by the platform and active IoT Hubs and smart objects.

We have configured and tested multiple virtualization configurations in order to evaluate the performance of the designed IoT architecture both on local and remote VMs. In particular, the Open Stack layer has been initially tested on a local installation at the Department of Information Engineering of the University of Parma and, at a later stage, on Microsoft Azure in order to obtain and measure more realistic




(a) (b)

Figure 4.8: (a) Average synchronization time (dimension: [ms]) respect to the number of synchronized resources; (b) Average remote resource access time (dimension: [ms]) in different application scenarios.

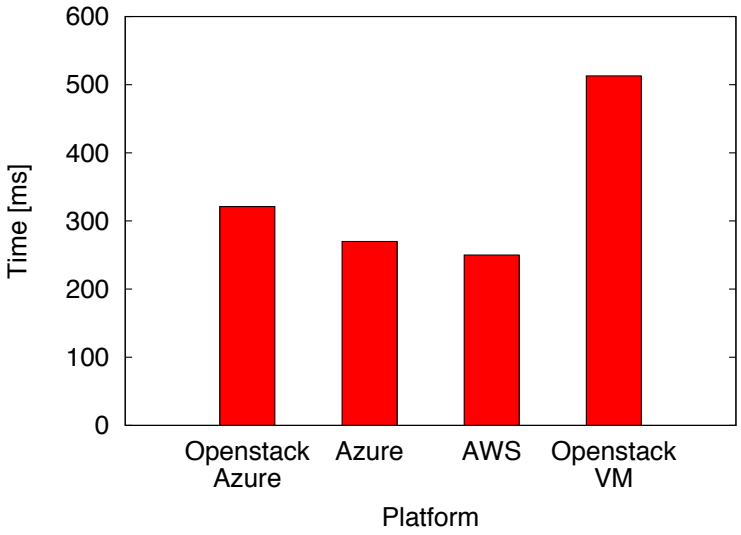



Figure 4.9: Average IoT Hub creation time (dimension: [s]) on different Cloud platforms.

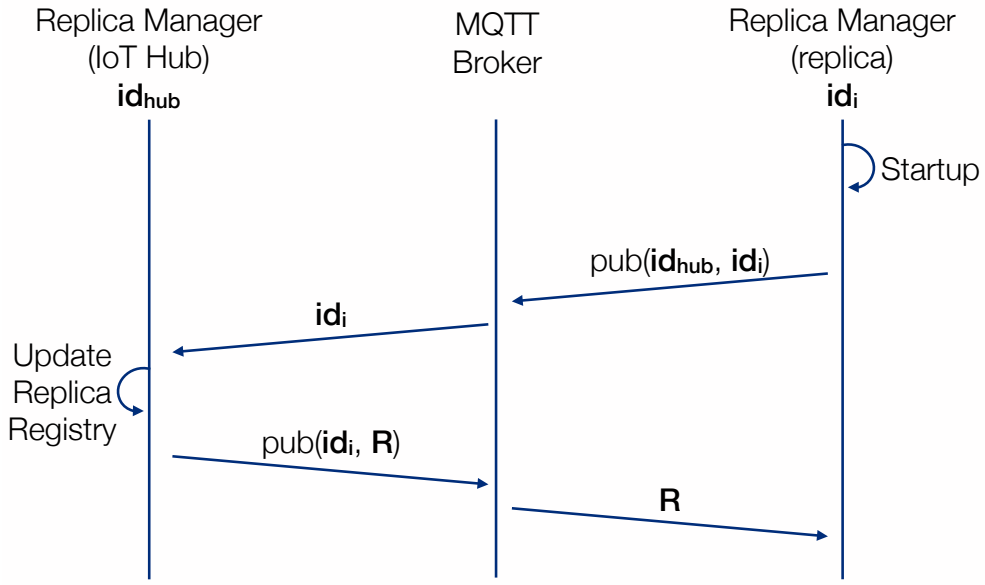



Figure 4.10: Synchronization procedure performed at startup of the replica of an IoT Hub.

results on a professional Cloud infrastructure. The local Open Stack installation runs on a physical machine with two 1.6 GHz processors and 3 GByte RAM, while the Azure configuration is characterized by a Virtual Machine with four 2.0 GHz cores and 8 GByte RAM. Both platforms have been used to dynamically manage replicas of active IoT Hubs and properly handle resource synchronization and remote data access. A virtual instance of an IoT Hub replica is characterized by an hardware profile with: a single core 2 GHz processor; 1 GByte RAM; and a 8 GByte disk space. The internal IoT Hub runs a Linux Ubuntu 14.04 LTS Operating System with SSH remote access, Oracle Java VM [59], and all the required functional software modules already installed and properly configured.

The following key metrics are defined to measure the performance key values at different architectural layers.

- *IoT Hub Replica Creation Time* (dimension: [s]): the time required to create

and run, on the target Cloud infrastructure, a new instance of an IoT Hub Replica.

- *Resource Synchronization Time* (dimension: [ms]): the elapsed time needed to synchronize a new resource between two IoT Hubs.
- *Resource Access Time* (dimension: [ms]): the time required to access and retrieve a response for a resource of interest. It can be associated with different configurations: (i) direct and local access to the CoAP Server smart object (e.g., if the consumer and the node are in the same network); (ii) remote access through the Cloud and communication with the physical Hub; (iii) remote access to the cached value stored on an IoT Hub Replica.
- *CPU Usage %* (adimensional: [percentage]): the percentage of CPU used by the IoT Hub core process.
- *Memory Usage %* (adimensional: [percentage]): the memory percentage used by the core process of the IoT Hub.

### 4.4.3 Performance Evaluation

The first phase of experimental performance analysis focuses on the evaluation of the amount of time required to create (from scratch) and run a new IoT Hub Replica instance on different virtualized Cloud infrastructures. The obtained results are shown Fig. 4.9. Each value has been obtained averaging over 10 different VM creation runs with a confidence interval of 99%. It can be observed that: (i) the average costs on remote and professional Cloud infrastructures is comparable; and (ii) the cost is higher on local and not optimized solution (such as the Open Stack instance running in our department). We remark that the metric corresponds to the total amount of time required to create a new VM from scratch (starting from a pre-configured image and adding the time to start all the required services and architectural software processes). This cost should be considered only once for each IoT Hub Replica and is significantly written off with the increasing of the hub lifetime. Native VMs on Microsoft Azure and Amazon EC2 present approximately the same creation time, while the use

| **Hub** | **Initialization** | **Resource Add** | **Sync Resource Group** | **Sync Single Resource** |
|---|---|---|---|---|
| [Local] Raspberry Pi CPU | 96% | 17.45% | 15.15% | 6.93% |
| [Local] Raspberry Pi Memory | +2,7% | +0.02% | +0.01% | +0.002% |
| [Remote] Azure VM Remote CPU | 97% | NA | 34.22% | 1.93% |
| [Remote] Azure VM Memory | +1.9% | NA | +0.01% | +0.001% |

Table 4.1: Average CPU and memory utilization percentages related to specific IoT Hub procedures on both local and remote instances.

of the Open Stack platform introduces a small delay associated with the additional overhead (of the platform itself) required to manage multiple s.

The second phase of the experimental analysis has been entirely focused on (i) the evaluation of the required time for synchronization of resources between two IoT Hubs (running as application or VM on the Fog Node); and (ii) the time needed by an external consumer to access a target resource of interest in different scenarios and configurations. In Fig. 4.8 (a), (i) the total amount of time required for the synchronization of a set of resources between an IoT Hub and its new Replica and (ii) the average time to synchronize a single resource in the same set are shown as functions of the number resources. The obtained results show that the average synchronization cost for a single resource is stable and, consequently, the total required time is proportional to the number of resources to be synchronized. Moreover, results show that for a reduced number of resources, the impact of the cost of MQTT connection creation is slightly relevant compared with the payload. Obviously, by increasing the number of resources synchronized using the same MQTT connection it is possible to reduce this effect and reach a stable value below 25 ms/resource.

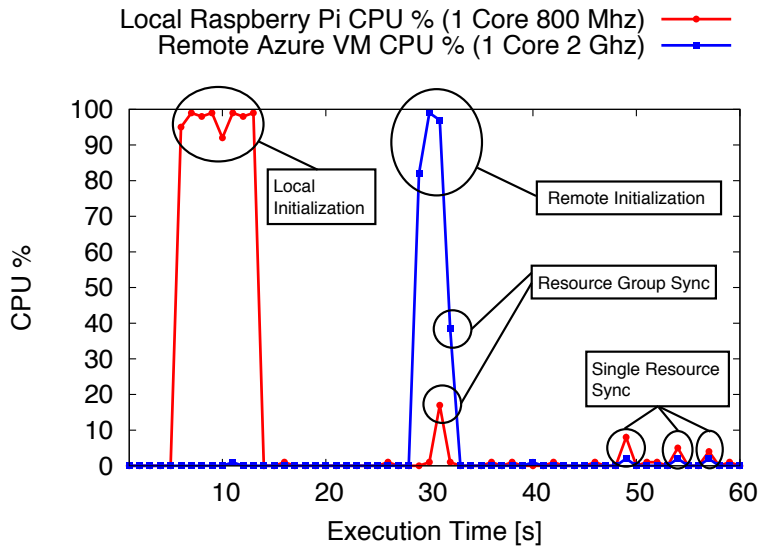



Figure 4.11: IoT Hub process CPU percentage usage (dimension: [adimensional]) on a local (Raspberry Pi node) and remote instance (Microsoft Azure VM).

In Fig. 4.8 (b), the average time required by a consumer to access a resource of interest provided by a smart object is evaluated in different scenarios. In particular, we have considered three different configurations where: (i) the consumer is in the same local network of the target smart object; (ii) the external actor accesses a cached value on the active IoT Hub Replica on the Azure platform; and (iii) the consumer accesses a resource that is not cached on the IoT Hub Replica and, consequently, requires a direct communication between the virtual Replica and the real IoT Hub. The presented results show, as intuitively expected, that the quickest access is obtained if the consumer is in the same network and does not require additional communication with a remote infrastructure. However, if the consumer is an external actor the average cost, still considering a realistic deployment through Microsoft Azure, is below 80 ms/resource. This value decreases if we consider a resource that is cached on the Replica Hub, which does not require any additional communication with the local Hub and, eventually, with the smart object.

Our experimentation has also investigated the cost, in terms of CPU usage of an IoT Hub process, both on local (Raspberry Pi node) and remote (Microsoft Azure VM) instances. In Fig. 4.11, the percentage of CPU usage during a run of 60 seconds of core activities is shown highlighting: (i) initialization of the main process and the creation of a set of 5 new resources on the local Hub; (ii) activation of the remote Replica and its initialization; (iii) synchronization of the group of 5 initial resources between local and remote replica Hub; and (iv) sporadic addiction of single resources until the end of the experiment. The obtained results show how the initialization of the Hub represents an intensive activity on both local and remote instances. The CPU usage incurs a significant one-time cost due to: setup the Hub configurations (such as node identification and software module selection); establishing the VPN connection activating the CoAP Server and the MQTT module (listener & publisher on specific topics); and starting up the Resource Directory. After the initialization phase, the CPU usage significantly reduces for both resource group synchronization and sporadic management of new single resources. These activities represent common and frequent tasks for an active IoT Hub that typically handles small variations of the set of managed resources during its lifetime. The main software process consumes a reduced amount of CPU (under 10%) for a small amount of time on both local and remote instances.

In order to complete the presented analysis, Table 4.1 shows the average values (obtained through multiple independent runs and a confidence interval of 99%) of CPU and Memory usage related to each specific Hub procedure. The presented data confirm the cost distribution, with a percentage peak due to initialization phase and lower values for group and single resource synchronization. Memory utilization has been measured as the offset respect to the previous value and depends on the Java Virtual Machine memory management [60]. In particular, when an object is no longer used, the Java Garbage Collector reclaims the underlying memory and reuses it for future object allocation without an explicit deletion (no memory is given back to the operating system).

An important aspect for a truly scalable architecture is the ability to quickly react to a dynamic load, characterized by the rate of incoming requests that need to

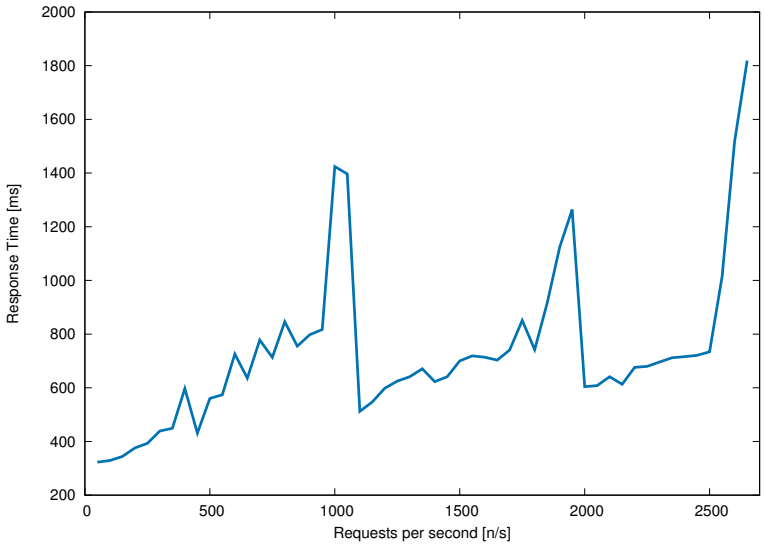



Figure 4.12: Effect of replica management with respect to the increasing number of requests per second on Microsoft Azure Infrastructure.

be served. The replication strategy proposed in this experiment aims at providing a flexible and efficient management of IoT Hub replicas in order to guarantee that the response time remains below a given threshold. The virtualization approach provides a flexible solution to guarantee significant availability and efficient load balancing in highly dynamic IoT environments. In order to validate the designed replica management scheme, an additional experimental phase has been carried out. Leveraging the same Microsoft Azure Infrastructure used for all the other experiments, we measure the response time for incoming requests for Smart Objects resources managed by the IoT Hub.

In Fig. 4.12, the effect of replica management in investigated in terms of response time (dimension [ms]) as a function of the incoming requests rate. In particular, the results refer to the shortest value according to which a new replica of the IoT Hub is activated whenever the response time exceeds a threshold set to 800 ms. The graph clearly shows that the response time tends to increase linearly as a function of the rate

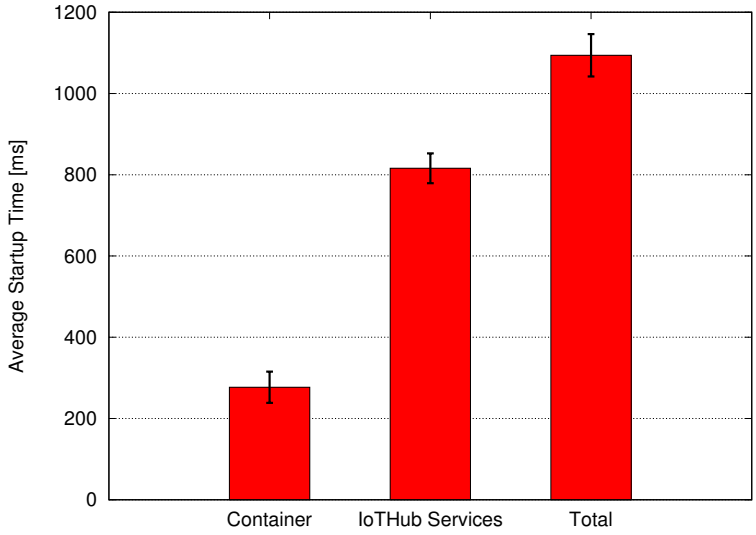



Figure 4.13: Average IoT Hub startup time (dimension: [ms]) on a Docker container.

of requests until the IoT Hub reaches a breaking point associated with the maximum number of requests it can handle. This is clearly visible in the areas of the graph characterized by steep slopes. When a slope is detected, we activate a new replica, which brings the response time back to a value that meets our operational requirements. As the request rate increases, new replicas are created. As shown in Fig. 4.12, we stopped our experimentation after the activation of 3 replicas. It is worth noting that: (i) a new replica is activated almost periodically (every 800 ms, according to the set threshold) and (ii) the slope of the response time between two consecutive activations is inversely proportional to the number of active replicas.

The definition and the widespread adoption of container-based technologies have significantly changed and re-designed the way Cloud and Fog applications can be deployed and delivered to final users. In particular the capability of hosting containers, is one of the main characteristics of the Fog Node we are envisioning. In order to provide a thorough performance evaluation of the proposed solution, we have also created a container-based version of the IoT Hub using the Docker platform [23]. The

container has the same networking configuration, features, and services running on the VM-based version but shares the Operating System Kernel and features with other running container instances. The local experimentation has been conducted using a Virtual Machine running Ubuntu 14.04, with one 2.0 GHz processor and 1 GByte RAM, on top of which Docker 1.11 has been executed with the aim of evaluating the startup time of the dockerized IoT Hub. This low-end hardware profile, compared to realistic data center facilities, has been purposely chosen to show the small footprint of the IoT Hub.

In Fig. 4.13, the average total startup time (dimension: [ms]) required to activate a fully operative IoT Hub instance, as well as the breakdown of container and IoT Hub services startup time, are shown. The presented results have been averaged on 1000 runs on the configured setup. The container startup time simply considers the activation of a Docker container instance. On top of this running instance, we measure the time required to activate all IoT Hub services and to respond to incoming HTTP and CoAP requests. The results show how a container-based approach can efficiently be adopted both on Cloud or Fog infrastructures (according to the target application scenarios) to support efficient and highly dynamic creation and management of IoT Hub replicas on top of existing host machines. Unlike a VM-based approach, container-based IoT Hub instances can be instantiated and removed dynamically, depending on the instantaneous load and without affecting the host machine or requiring infrastructure-related efforts (such as file/template management, configuration and activation of a new machine).

## 4.5 Results Analysis

We are evaluating a particular service, called IoT Hub, activated on the Fog Node, leveraging the flexibility introduced by the Fog-based vision of Internet access networks, object of this dissertation. The proposed IoT Hub operates at two different layers of the protocol stack: i) at the link layer, it is able to bridge different physical networks and merge them into a single all IP network; ii) at the application layer, it provides several functions that are used to enable resource discovery and interoper-

ability among applications.

An implementation of the IoT Hub has been realized in order to evaluate its practical feasibility and performance. The experimentation has been conducted in a real-world IoT testbed comprising several heterogeneous devices. The IoT Hub introduces significant benefits and has proved to be an enabler for resource discovery and seamless interaction with IoT applications. The performance evaluation has shown that the IoT Hub is able to manage a number of heterogeneous physical networks, each with several devices, with a limited use of resources, in terms of processing and memory, thus making it possible to deploy an IoT Hub even on low-end devices, such as RPis. This result is particular important, because it show that this application can be hosted on Fog Nodes, with the same capabilities of the Access Controllers already deployed for the management of public Wi-Fi Internet access. This ensure the feasibility of this approach, with a deployment effort that does not involve the Hardware replacement.

A second deployment was implemented in order to evaluate the abstraction of an IoT network by automatically cloning and virtualizing the physical network through a Cloud platform. As the IoT Hub plays a central role in the management of smart objects, an architecture based on Cloud (or Fog) IoT Hub replicas has been proposed in order to provide a scalable and efficient management of resources and to protect the IoT Hub from extremely high processing loads due to concurrent requests and/or attacks. The overall architecture has been entirely implemented using Open Source software libraries and relying on different Cloud platforms.

## 4.6 Chapter Conclusions

An extensive experimentation has been conducted in order to prove the feasibility, of the proposed solution, and evaluate its feasibility and performance, according to key metrics and under several conditions, in terms of number of managed resources, hardware used, and Cloud platform. The experimental results show that IoT Hub replicas can be created and synchronized, between Fog and Cloud, with good performance, thus enabling an efficient and scalable management of resources, while providing several benefits, such as seamless and secure access to resources, load balancing on

replicas, and protection of the real IoT Hub.

The information we gathered in this analysis show that the introduction of Fog and the consequent design of the application on three layers (End device, Fog Node and Cloud) is not just feasible, but it also introduces clear advantage without requiring to increase the resources needed, compared to a traditional managed Internet access Network.

# Chapter 5

# Real Cases Benefits Evaluation

## 5.1 Chapter Introduction

In this chapter we evaluate the first effects of the deployment of Internet access networks, based on the approach introduced in Chapter 3, on large scale. Working in partnership with an Italian Wireless Service Provider (WISP), called Guglielmo, and the technology provider Caligoo, we started deploying a Fog-based solution for the management of Wi-Fi Internet access network, and we are able to identify and measure the first effects of this new approach, and explore the main consequences of this shift, envisioning future developments. Guglielmo operates in Italy since 2004 and, at the time of this writing, is one of the main WISP in the country, managing Wi-Fi Internet access networks, counting more than 38000 Access Points, with more than 12'000'000 registered devices that made at least one connection. Guglielmo supported our research project and, with a spin off company called Caligoo, elaborated a road map for the development of a technical solution, based on the idea of Fog Computing, SDN and virtualization systems, managed through API, as described in Chapter 3. Even if not all the capabilities are ready yet, we started the deployment on field of solutions embracing this new approach, and we are able to collect data in order to perform an evaluation of the main changes introduced by this new vision.

## 5.2 Traditional WISP Approach

In general, the main services provided by WISPs, that manage Wi-Fi hotspots, are related to the Authentication, Authorization and Accounting (AAA), and to the policy enforcement, based on user's profile. WISP's traditional platform, is based on two main elements: i) an access controller, deployed on site and able to redirect users to a login page and enforce policies on the traffic; ii) a remote AAA platform. The interaction between the access controlled and the remote platform is usually based on RADIUS protocol [61] that supports an authentication process based on username and password. In general, a large number of different user's registration methods have been developed and, in some cases username and password are automatically created by the system and they don't need to be typed on provided directly by the users.
Main limitation of this approach are the following.

- Lack of flexibility: traditional approach just provides Internet access with limited capability to enforce policies only on basic parameters, such as connection time and bandwidth control.

- Lack of interaction with third parties applications, mainly limited to the possibility to statically authenticate, using some kind of static white list, end devices such as smart object vertical or silos applications.

- High traffic from Access Controllers, deployed on field and the centralized platform for AAA services, statistics and analytics. If Access Controllers are not autonomous in any decision and they act like clients of the remote platform, they generate service and management traffic that is part of the "well-known manageable traffic", measured and reported in Chapter 2.

- Not optimal resources usage. The sizing of the Access Controllers, in general, is based on the expected maximum number of simultaneous connected users, the maximum number of connection attempts received in the same time and the total available bandwidth of the Internet link. In particular, the access Controller has to be designed to support the peak of the expected workload for the

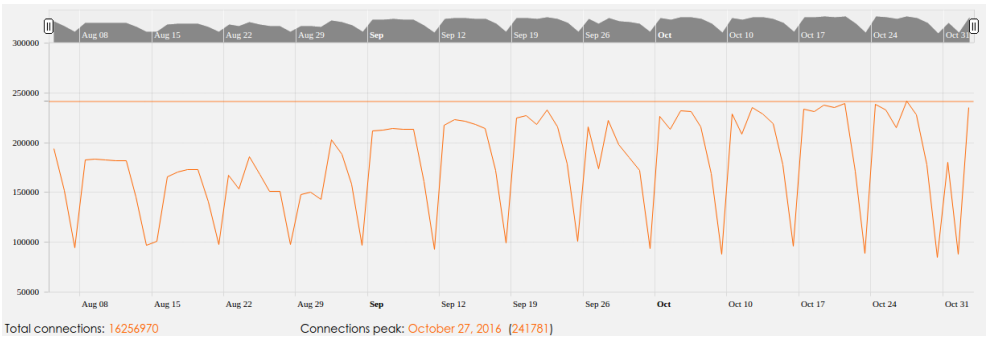



Figure 5.1: Number of simultaneous connections on a country wide network of Hotspot, during a period of three months.

specific location. Some locations have a highly variable usage, and often the workload is predictable, because it is variable with daily or weekly period, like we can clearly observe in Hotels (see Chapter 2), restaurants or public offices. The net result is that a high number of Controllers are under-utilized, or even in idle, during a big part of their life. Fig. 5.1 shows overall number of simultaneous connections on a country wide network of Hotspot during a period of three months. In this representation it is clearly visible the variability of service usage, and thus of the workload on Access Controllers on field.

## 5.3 Main Changes

Introducing Fog Computing on a traditional Hotspot management platform, involves many different layers and it requires to redesign several parts. Working on the system in production, we mainly focused on two main elements:

- the evolution of the infrastructure, involving the Access Controller and introducing the SDK for mobile App;
- the redesign of provider's internal applications in order to leverage the capabilities introduced by Fog Computing and SDN.

### 5.3.1 Fog Node: the Evolution of Wi-Fi Access Controller and the introduction of Mobile SDK

The evolution of Wi-Fi Access Controller is the key point of the introduction of Fog Computing and SDN in access network, because it is deployed close to the users, exactly where the Fog is. We transformed the Access Controller in a Fog Node, basically installing an hypervisor, in order to add the capability of hosting VMs, and an SDN module (adapting the solution developed by our partner PLUMgrid) based on IoVisor [39]. We also added storage, activated a VM with Docker [23] engine for the management of Docker containers and a VM able to control the infrastructure and expose API for external applications. We also developed an SDK for mobile App, running on smartphones or tablet, in order to make Apps, running on mobile devices, able to interact with the Fog Node.

### 5.3.2 Application Re-Design

The evolution of the infrastructure has a huge impact also on the provider's internal applications design. THis process is still ongoing, and in particular we worked on the AAA system and the statistics and analytics systems, in order to make them able to use resources on the edge of the network and not just on the Cloud layer.

## 5.4 Main Consequences

Changing the platform of a country wide network for Internet Wi-Fi access control, in order to introduce a new active layer on the edge of the network, can not be made in a single step, so, for this preliminary study, we mainly focus on two main functions, in order to evaluate the consequence of the introduction of the new approach we are introducing. In particular we worked on: i) the authentication process; ii) the hosting of third party application on the Fog node, and iii) the resources optimization.

| | |
|---|---|
| Daily average connections before authentication with App | 104 |
| Daily average connections after authentication with App | 297 |
| Daily average users before authentication with App | 78 |
| Daily average users after authentication with App | 239 |

Table 5.1: Average daily number of connections and users, before and after the introduction of the seamless authentication based on mobile App, in a real location.

### 5.4.1 Authentication

Introducing Fog Computing, we developed AAA services on three layers: i) the SDK on the mobile App; ii) the Fog Node and iii) the remote AAA system. Leveraging the direct interaction between the App and the Fog Node it is possible to implement a seamless authentication that improves the quality of the user's experience and makes the usage of the WiFi service more easy. Table 5.1 and Fig. 5.2 show the improvement in Wi-Fi network usage, in terms of number of connections, after the introduction of this approach to authentication. In particular, data in Table 5.1 shows that the average number of users and connections are increased by a factor of three.

### 5.4.2 Hosting Applications on the Fog Node

One of the main topics in the Wi-Fi Internat access market, is the capability to use the Wi-Fi network to collect data about users and perform some kind of engagement. The growing market related to these features, pushed manufacturer to develop wireless access solutions able to provide these services. The available platform, generally, are based on an enhanced access points controller, or a hierarchy of controllers, where data collection capabilities are added to traditional Access Points control and optimization functions, and on centralized platforms able to expose or elaborate collected data and perform some kind of engagement on the clients. This kind of platform, usually requires dedicated appliances and allows poor interaction with other functions, such as AAA services or users registration, performed by external applications. In-

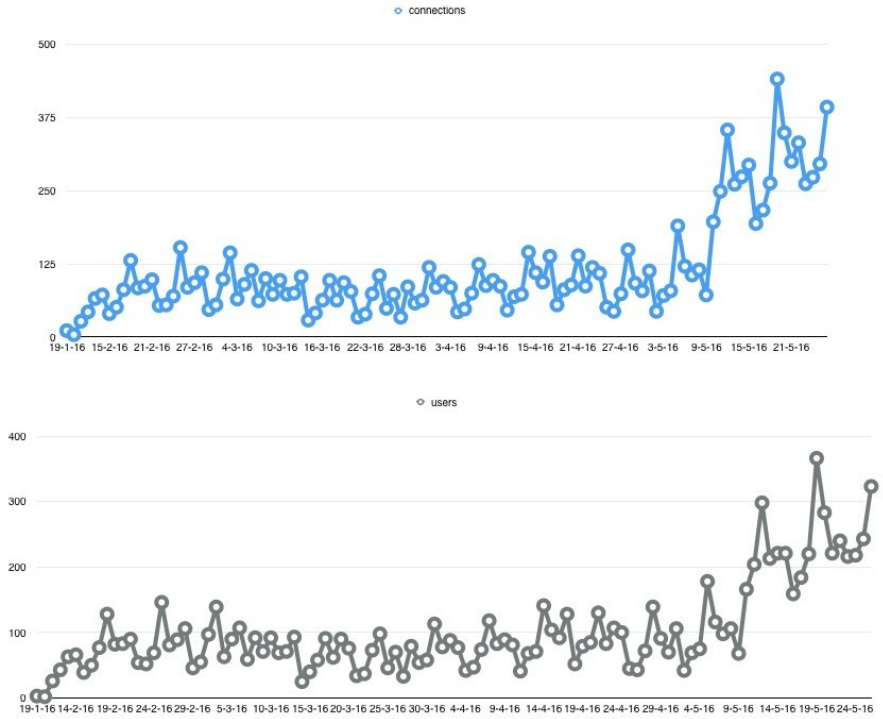



Figure 5.2: Improvement in Wi-Fi network usage, in terms of number of connections, after the introduction of seamless mobile App-driven authentication system.

troducing Fog Computing concepts, we are able to deploy these functions with an high level of integration with the rest of the platform. In particular, leveraging the possibility to run VMs on the Fog Node, we deployed virtual version of access points controller [62] and virtual machines for data analysis, directly on the edge of the network, able to provide pre-processed data and interact with the AAA system, without generating traffic on the Internet connection and saving extra hardware. We measured that, redesigning the provider's core application relate to location analytics in order to leverage the Fog layer, deploying this new version of the application in a location where the wireless infrastructure is detecting more than 500,000 events every day as average, we are able to generate statistical data about indoor users localization,

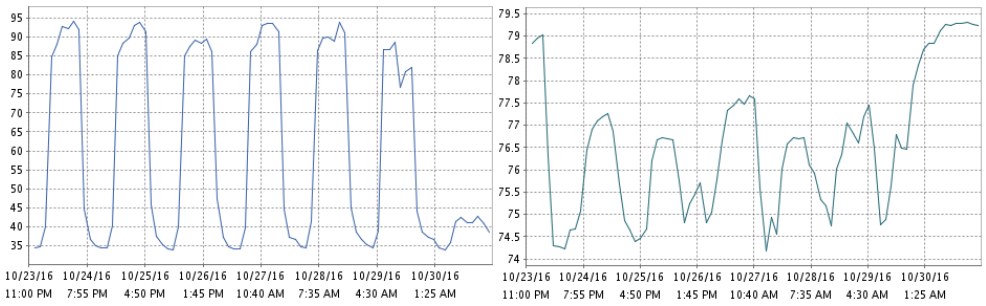
| | |
|---|---|
| Average number of events locally processed daily | 532,177 |
| Total number of events locally processed since the implementation | > 55,000,000 |
| Average traffic saved by local processing of data daily | 300 MBytes |

Table 5.2: Data related to the workload locally managed and the traffic saved by the version of provider's analytics, able to leverage the Fog Computing approach.

saving about of 300 MBytes of Internet traffic every day, leveraging the capability to process events data on a VM running locally on the Fog Node. Table 5.2 shows data collected after the implementation of this new version of the analytic system.

### 5.4.3 Resources Optimization

One of the most innovative features, introduced by the Fog-based approach, is the possibility to optimize the usage of the resources deployed on field, reducing the workload of the centralized platform in the data center. As described in Section 5.2, in a traditional platform for hotspot management, the nodes deployed on field are sized for support a peak of activity, but actually they experience high variable workload and spend a big part of their time, underused or even in idle. Redesigning applications, like statistics data aggregation or report generation, as queue of tasks and software workers able to pick up a task from the queue and generate a task result, or put back the task in the queue in case of failure, and deploying workers, as Docker containers, on Fog Nodes, we are able to use resources on the Fog for non-real time tasks, traditionally managed in the Cloud. In particular we analyzed a set of 3,500 locations managed by two Fog Nodes, this deployment was possible because all the locations are venues of the same brand, connected through a star network, they can be managed by a cluster o two Fog Nodes in the hub site. During 3 months, we measured a total number of 5,677,120 connections, with an average of 63,079 connection per day and a peak of 98,024 connections in one day. During the same period, we also counted 974,818 different users, with a peak of 43,138 in one day and an average number of 29,601 per day. In order to manage this service, we deployed two Fog Nodes with the




(a) (b)

Figure 5.3: (a) CPU usage on the considered Fog Node; (b) RAM usage on the considered Fog Node.

following hardware characteristics:

- Processor: 2.40GHz.
- CPU: 16 physical, 32 virtual.
- RAM: 64 GB.
- Hard Disk: 600 GB
- Network Ports: 4 ports 1 Gb, 2 ports 10Gb

The considered locations are basically public offices, and the Wi-Fi service is for the clients, for this reason, in general from 8 pm to 8 am, Saturday afternoon and Sunday, there are not any connection. This means that we have the Fog Node available for other tasks. Fig. 5.3 shows the CPU and RAM usage on the Fog Node, and it is easy to see the idle periods.
Every night the platform generates statistical data, about the Wi-Fi usage during the last day, and a report for the locations owner. Traditionally these activities are performed by a dedicated virtual machine on the WISP data center. The characteristics of this VM are the following:

| | |
|---|---|
| Number of locations involved | 3,500 |
| Duration of the experiment | 3 months |
| Total number of connections | 5,677,120 |
| Daily average number of connections | 63,079 |
| Maximum number of connections in one day | 98,024 |
| Total number of different users | 974,818 |
| Daily average number of different users | 29,601 |
| Maximum number of users in one day | 43,138 |

Table 5.3: Data related to the experiment performed in order to show that Wi-Fi hotspot provider's core applications, related to users and connections statistics, can be redesigned in order to locally perform their task on Fog Nodes, with the same capabilities of the access controllers already deployed, fixing the lack of resources optimization of the traditional approach.

- 4 CPU
- RAM: 20 GB

The elaboration at the data center of the data, for the statistics and the report, takes an average time of 40 minutes every night with peak of 8 hours, when the process is not optimized or other particular activities are ongoing in the data center at the same time. Considering the available resources on Fog Node and the resources needed by the elaboration in data center, we can conclude that, for this kind of locations, a Fog version of the statistics and reporting application, is not just feasible, with the existing capability of the Fog Node, but could lead to a notable savings of resources, at the data center, and of Internet bandwidth, improving client's experience, reducing the time of elaboration, and decoupling this activity to others scheduled activities that take place on the centralized infrastructure, such as backup or other customers' statistics elaboration. Table 5.3 shows the size of the experiment performed on field. For smaller locations, where more constrained Fog Nodes are deployed, it is possible to not have enough resources for statistical elaboration on the edge. But considering the

minimal resources, needed by the software to run, and matching this information to the sizing of the Fog Node, on the basis of the characteristics of the location, it is possible to estimate that the Fog approach, for this kind of services, is possible without changing the Node characteristics, compared to a traditional Access Controller.

## 5.5 Chapter Conclusions

In this chapter, we describe and analyze the consequences of the deployment if Fog-based solutions, in a production platform of a real WISP in Italy. We noticed that the change of the infrastructure, introducing the Fog layer, must be supported by a re-design of the applications, in order to fully exploit the potential of this new approach. The main consequences are related to the following points:

- a more efficient Authentication process, that leads to a more usability of the service and an improvement of the number of connections and user about three times larger, compared to traditional authentication systems;
- the possibility of bring on the edge of the network analytics and indoor users localization and engagement services, without deploying a parallel infrastructure;
- new approach to resources optimization, based on the usage of Fog Nodes to perform elaboration, traditionally handled at the data center.

The last point is particularly promising, since Fog Nodes appear to be largely under-used, and a future scenario, where Fog Nodes are allowed to process data related to other locations than the one where are deployed, the platform could virtually work without a data center, using the same type of nodes already deployed for access control and hotspot management.

# Chapter 6

# Fog Architecture Design

## 6.1 Chapter Introduction

Introducing the concept of Fog computing and SDN in Internet access networks for Smart Objects and user's end devices, it is not just a way to reach better performance, leveraging enhanced capabilities of edge networks, but it has a direct impact on the development of all the elements involved in networking. For this reason the way to exploit the potential of this new approach is extending and applying these concepts, not only on network elements, but also on end devices and applications, exploring the consequence, of this vision, on network design and optimization.

In this chapter, we develop the scenario of Fog-based access network with SDN capabilities, extending the vision, from a single deployment to a global design of a *Fog layer* between the Cloud and end devices. We explore the consequences of this new approach to access networks, on all the element involved in the Internet connectivity and computer networks in general, from the application design to a new concept of end device. Considering the changes introduced by this interpretation, we also focus on the definition of the architecture of the Fog layer, supporting consideration about design and optimization, with experimental data and introducing the concept of dynamic overlapping clusters for Fog-based access networks.

In the envisioned scenario, different types of locations, such as hotels, airports

or shops, deploy Fog-based interoperable access networks, for their guests. The final result is a large scale distributed Fog layer where data and applications can move directly between Fog nodes, even if managed by different owners. Having this layer well organized, allows to redesign the applications and to redefine the role of end devices. The final goal is not just support heavy workload bursts, but to provide a more scalable and flexible support, for forthcoming services, than a traditional Internet access network able to connect end devices to the cloud with good performance.

## 6.2 From End Devices to User Virtual Environments

Networks are, in general, developed in order to exchange data between applications running on specific devices or in the Cloud. This traditional vision is strictly related to the concept of end device, as the host where applications are running on, and reachable leveraging the multiplexing capabilities of layer 4 protocols, such as TCP or UDP. In a simplified vision, the network basically routes the traffic, for the user's set of application, to the user's device, identified by a layer 3 address.

A Fog-based approach, allows user's application to run directly on the access network and not only on the end device. On the basis of this capability, we can extend the concept of *end device*, hosting user's applications, introducing the evolved idea of *User Virtual Environment*, identifying a network user with a complex environment including one or more networks of applications. Every User Virtual Environment is, in general, a virtual infrastructure, associated to a specific user that, in our vision, could be human or a smart object. The User Virtual Environment, implemented with applications running on VMs or Containers connected by SDN-networks, is deployed not necessarily on a single Fog Node, but potentially distributed on the virtualized infrastructure of the Fog access network involving several Fog Nodes, while the user just need an interface in order to interact with his own environment. Figure 6.1 shows the evolution from the concept of user's end device to the User Virtual Environment.

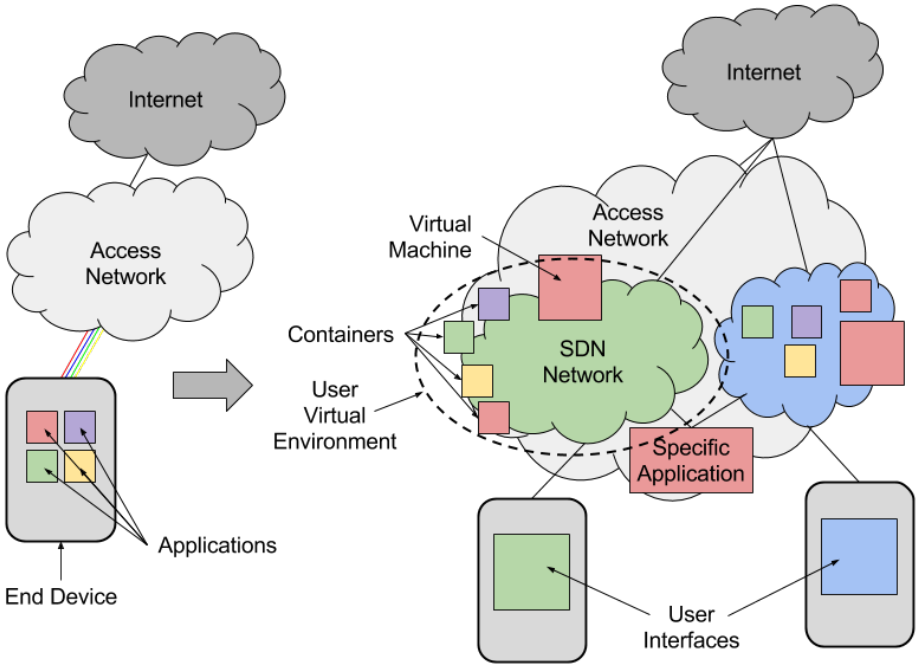



Figure 6.1: Graphical representation of the shift from end device to the concept of User Virtual Environment.

## 6.3 Models

This shift from connected devices to the deployment and the interconnection of complex environment on a virtualized access network, has consequences on network design and optimization. An important element for future considerations, is the capability to define and handle the deployment of environment, for this reason we propose a modelization of the elements involved in the Fog layer, on the basis of the needed resources.

### 6.3.1 Model for User Virtual Environment

Providing a model for the User Virtual Environment has the main purpose of giving a useful element to develop strategies for the design of fog-based access networks and for mapping User Virtual Environments on those networks, in an efficient way. Important elements for planning the deployment and evaluate possible strategies for network optimization, are the resources needed by the User Virtual Environment. In particular we used the following set of parameters in order to provide a description and define a simplified model of the environment.

- Total CPU needed by applications, basically the number and type of virtual CPU on the Fog layer, needed by the User Virtual Environment.
- Total RAM memory needed by applications.
- Total memory storage needed by applications or directly by the user, such as personal files.
- Maximum latency tolerated between applications and user interface or between different applications.
- Minimum throughput needed between applications and user interface or between different applications or between applications and their storage.

The environment could include different applications depending on external factors, for this reason, basically the environment is an open infrastructure and in its description can be structured as a basic part related to the user, and extra applications, added in specific situations. Since extra are added for specific purposes, thus in controlled situations, the description of the basic environment appears to be more important, than the modeling of specific applications, in order to plan the deployment and optimize the management of the Fog layer.

### 6.3.2 Model for Application

Even if the resources needed by applications, are considered in the environment modeling, it is important to have an application description for two main reasons: i) if an

environment is distributed on multiple Fog Nodes, in our vision actually applications are running on different nodes, but a single application can not run on multiple nodes, for this reason a way to describe application is useful for planning the distribution of the environment on multiple nodes; ii) we use this description in order to modeling extra applications that could be added to a specific environment. In order to provide a modeling approach compatible with the environment description, we describe applications on the basis of the following needed resources.

- CPU needed, basically the number and type of cores.
- RAM memory needed.
- Storage memory needed.
- Maximum latency tolerated to the user interface.
- Minimum throughput needed to the user interface, other applications or storage.

### 6.3.3 Model for Fog Node

Every Fog Node can be described, accordingly to the environment and applications description, on the basis of the following resources available on the node.

- CPU type, basically the number of cores and clock frequency.
- Total amount of RAM.
- Total amount of storage, basically the Hard Disk.

Other important parameters are related to the capability of interaction with other nodes, in particular we consider:

- latency;
- throughput.

We envision a situation where Fog Nodes are able to estimate latency and throughput of connections with other nodes, even if these parameters are particularly critical because they depends on the load of the network in a specific time, and because a direct measurement, of these parameters, requires the generation of traffic, increasing the workload on network and nodes.

## 6.4 Design and Optimization Problem

The design and optimization of a Fog-based access network are still open problems and, at this stage, they can be considered like the unique problem of finding the architecture able to optimize the usage of available resources on the Fog. In the following sections, we propose two different architectures and, on the basis of data collected on field, we identify which one is able to use available resources more efficiently, under specific conditions. Since we consider the optimization of resources usage, as the driver for the design of Fog-based access network, we need to carefully define the optimal solution for this kind of networks. The basic concept is to deploy the User Virtual Environment on the Fog-based infrastructure in an efficient way. In order to have a more formal approach, we introduce two possible definition of the optimization problem.

- Minimize the number of needed Fog Nodes in order to deploy a given number of User Virtual Environments.
- Maximize the number of User Virtual Environments that it is possible to deploy on a given number of Fog Nodes.

Even without an analytic approach, it is possible to consider that, under the assumption that a Fog Node is able to provide more resources than needed by a single User Virtual Environment, these two definitions are equivalent. Since, in general, the deployment of nodes is an harder constraint, in our analysis, we base the design problem, and in particular the evaluation of possible architectures of the Fog-based infrastructure, on the second criteria. This problem has multiple degrees of freedom and in particular it depends on the characteristics of User Virtual Environments and

of the Fog Nodes infrastructure. For this reason we prefer to approach this problem with basic considerations about possible architectures supported by data collected on field, instead of working on an analytical approach that would require some assumptions about environment and Fog Nodes size, that can not be correctly evaluated at this stage of the implementation.

## 6.5 Clustering

Fog-based access network is basically a layer of interconnected nodes, between clients devices and the Cloud. In order to control this infrastructure and, in particular, the deployment of User Virtual Environments, we divided the deployed Fog Nodes in subsets, called clusters.

### 6.5.1 Static Clusters with Hierarchy

In this case, a cluster is a statically defined set of Fog Nodes. Every node can join only one cluster and each cluster has a main node that collects information about the cluster, interacts with the cloud and manages the deployment of applications and SDN networks inside the cluster. We refer to this architecture as the one with *static clusters with hierarchy*. In this scenario, when a Fog Node discovers a specific client, which could be a connected user after the authentication or a smart object just discovered, contacts the main node of the cluster that retrieves, from the cloud, the User Virtual Environment related to that specific client. The main node also manages the deployment of the environment inside the cluster, on the virtualized and distributed infrastructure, provided by the Fog Nodes.

This architecture is easily manageable and highly reliable, because it can be implemented with technologies already used in data center, since a cluster of Fog Nodes serving multiple users, can be considered similar to a multi-tenant data center. The presence of a master node and the very idea of static cluster (including always the same nodes), simplifies the interaction with the cloud, the deployment of the environments and the collection of data related to the status of the cluster. Figure 6.2 shows the architecture based on static clusters with hierarchy.

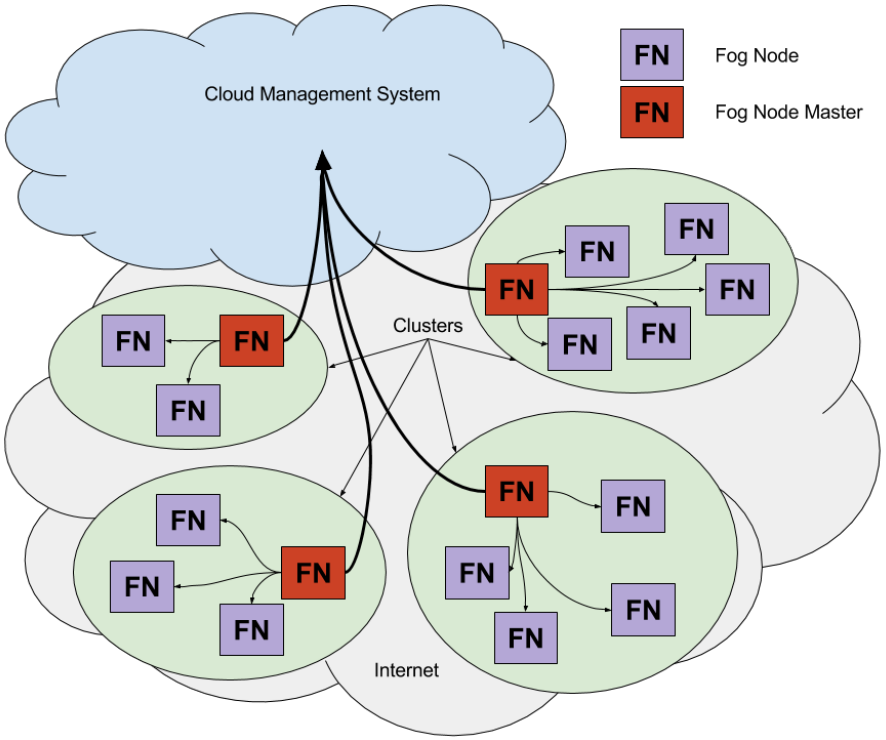



Figure 6.2: Representation of the concept of static cluster with hierarchy.

This approach is efficient if the cluster satisfies the requirements of all the environments related to the users connected to the cluster. If the cluster is not able to provide the resources needed by the environment, for example not enough RAM or too high latency between the nodes of the cluster, this approach appears to be not flexible and the environment can not be deployed. This issue is more critical because it not depends only on the balance between needed and available resources, but in some cases it also depends on which Fog Node the client is connected to. For example if a client is connected to a Fog Node on the border of the cluster, it is possible that some resources inside the cluster can not be used, because of the high latency, while other available resources with low latency, could be used, but they are on nodes inside other clusters, so the environment can not be deployed. Figure 6.3 show a graphic

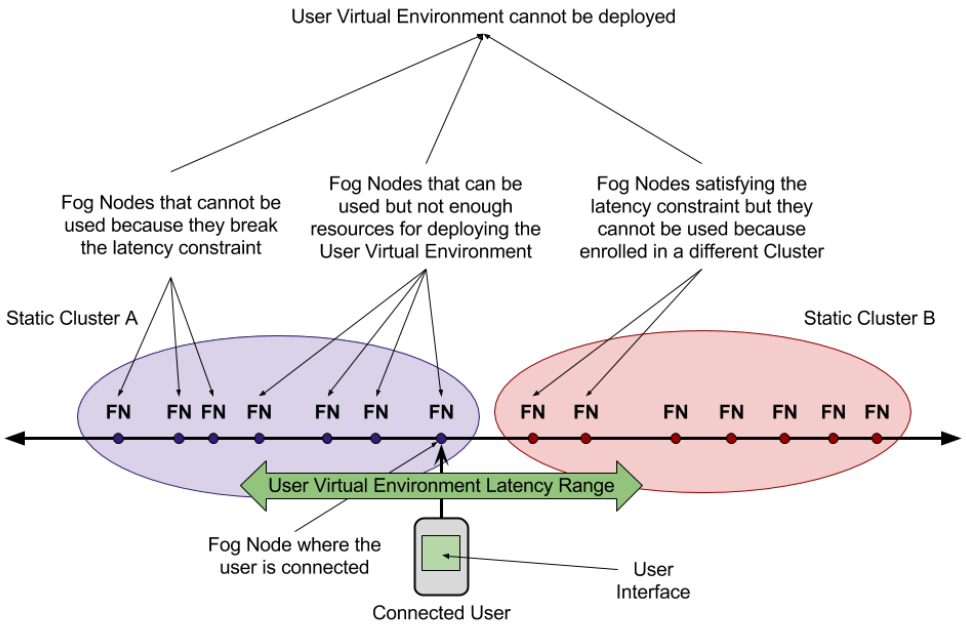



Figure 6.3: Representation of the impossibility to deploy a User Virtual Environment in a static cluster, with Fog Nodes available outside the cluster.

representation of this case. Another critical limitation of this approach is related to the capability of connecting extra application, not originally included in the environment, that are running on Fog Nodes outside the cluster. This situation requires inter-cluster environment deployment rules and protocols that weaken the benefit of this static architecture.

### 6.5.2 Dynamic Overlapping Clusters

In order to overcome the limitations of clusters with hierarchy, we introduce an alternative architecture, based on *dynamic overlapping clusters*. With this approach, clusters are not statically defined and a node can join more than one cluster at the same

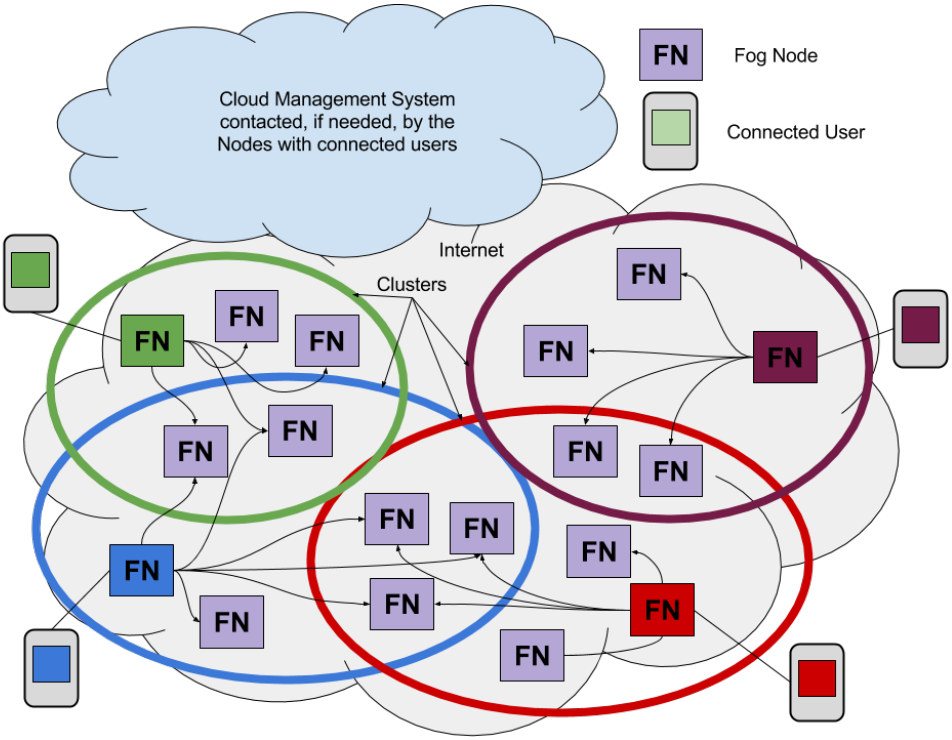



Figure 6.4: Representation of the concept of dynamic overlapping cluster.

time. Every node collects the descriptions of other nodes, defined in Section 6.3.3. When a client is discovered, the node retrieves the description of the User Virtual Environment associated to the client, including extra applications, and it selects the Fog Nodes to be involved in the deployment of the User Virtual Environment, on the basis of its knowledge of the Fog infrastructure. In other words, the Fog Node, where the client is connected, and User Virtual Environment, associated to the client, are paired and every pair has its own cluster, dynamically created by the node on the basis of the description of available Fog Nodes and the environment. The Fog Node directly manages the deployment of the environment on the cluster just created. Figure 6.4 is a graphic representation of the concept of dynamic overlapping cluster introduced in this section.

## 6.6 Architectures Comparison

We introduced two possible architectures for the infrastructure of Fog Nodes in access networks, now we investigate which one is able to provide the best solution for the optimization problem defined in Section 6.4.

In order to analyze both architectures, we rely on the modeling introduced in Section 6.3. Some of the parameters, that describe the Fog Node, and therefore also the cluster, such as CPU, RAM and storage, depend on the type of Fog Nodes more than on the architecture, while latency and throughput between nodes in the same cluster, or in different clusters, have a direct impact on the capability of the architecture of maximizing the number of deployable environments. In particular we define the latency between clusters: the lowest latency between one of the nodes in a cluster and one in the other cluster. This parameter makes sense only in case of static clusters and represents the minimum latency between clusters, while in the case of dynamic overlapping clusters, this parameter is always equal to 0 since the same node can be part of different clusters. This parameter is particularly important because, comparing its value to the maximum latency tolerated by all the application in a specific environment, it is possible to understand if an environment, that can not be deployed on a static cluster, because it is not possible to find enough computing or storage meeting the latency requirement inside the static cluster, could be deployed with an approach based on dynamic overlapping cluster. If the latency between static clusters is lower than the maximum tolerated latency inside the environment, then the static cluster becomes a limitation, excluding nodes that could be used in the environment deployment. This could make the difference if there are not enough nodes in the static cluster, to meet the requirements of the environment and satisfy the constraint on the latency. A similar approach and conclusions could be applied on the throughput requirements .

In order to make this consideration more clear and point out which conditions on latency or throughput eventually make an architecture more efficient than the other, we introduce a graphical representation of the problem in Figure 6.5. As it appears on this representation, if the static cluster includes nodes with mutual latency higher

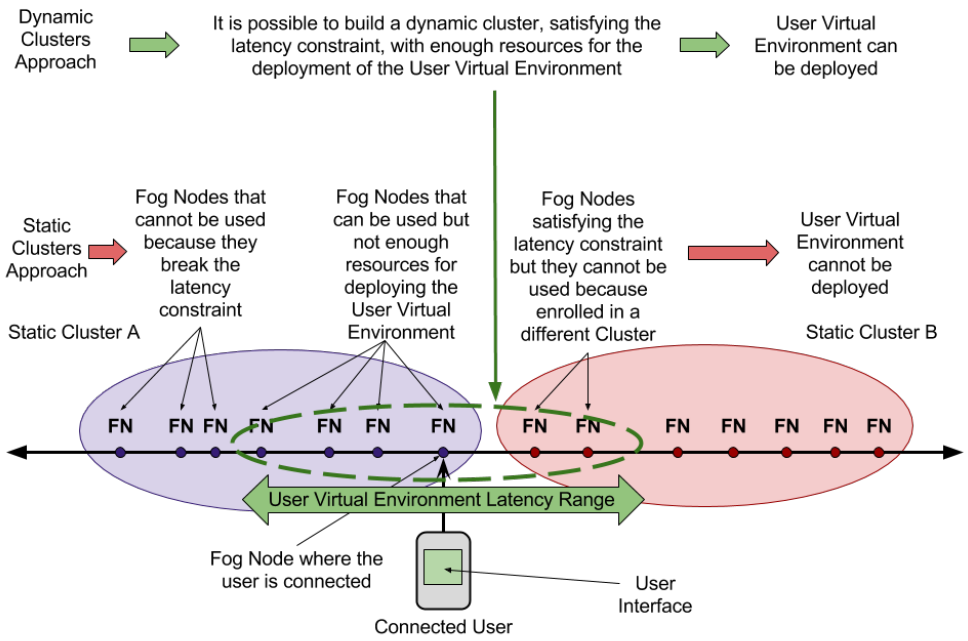



Figure 6.5: Graphical representation of the situation where a User Virtual Environment can not be deployed inside a static cluster, but there would be available Fog Nodes outside the cluster in order to make the deployment possible in an dynamic overlapping cluster.

than the one tolerated by the applications of the environment, then the possibility to find enough CPU, RAM and storage, inside the static cluster, satisfying constraint on latency and throughput imposed by the environment, depends on where the client is connected. If the static cluster, only includes nodes with mutual latency and throughput that meet the environment requirements, then these two parameters are no more a limitation, but a problem of resources saturation can arise in the cluster.

The dynamic overlapping clusters approach can make the difference if there are available Fog Nodes, which satisfy the latency and throughput constraints, but they

were not included in the static cluster. This key point is a direct consequence of the distribution of Fog Nodes on the basis of the mutual latency. In particular, considering the latency and starting from a full mesh network of Fog Nodes, if we define an upper limit for the latency and we don't consider all the links that exceed this limit, then we reduce the full mesh network of nodes to basically two possible different scenarios important for our analysis. We refer to this operation with the term *pruning* of the network. Once we eliminated, from the full mesh network topology, all the links with a latency higher than the limit, we can have a first scenario, where we can identify separated group of connected Fog Nodes, or a second scenario, if it is still possible to connect all the nodes. A dual approach can be based on throughput, not considering the links with a throughput lower than a defined threshold.

It is easy to prove that the reduction to one scenario or the other depends on the limits imposed on latency or throughput. In particular, the more the limit on latency is low, the more the second scenario is less likely. If we consider limits on latency and throughput equal to the values of latency and throughput included in User Virtual Environment modeling (see Section 6.3.1), and we prune the network not considering the links that violates these limits, then the remaining network topology gives us interesting information about the best approach to clustering. In particular, if we have the first scenario, with disconnected groups of nodes, then the approach based on dynamic overlapping cluster does not bring a valid improvement, because it is possible to define static clusters matching the groups of nodes still connected after the pruning, and it is not possible to involve in the environment deployment, nodes external to the cluster because there are not links to external nodes that meets the environment's requirements on latency or throughput. In this case the static clustering with hierarchy is preferred, leveraging an easier management. If, after the pruning of the full mesh network, we have the second scenario with a continuity of connectivity between nodes, then it is possible to leverage the higher flexibility of dynamic overlapping clusters, overcoming the limitation of static clusters on using available nodes for the deployment of User Virtual Environments.

In this approach, the effects of latency and throughput are mainly related to the connection between the user interface, in general running on an end device connected

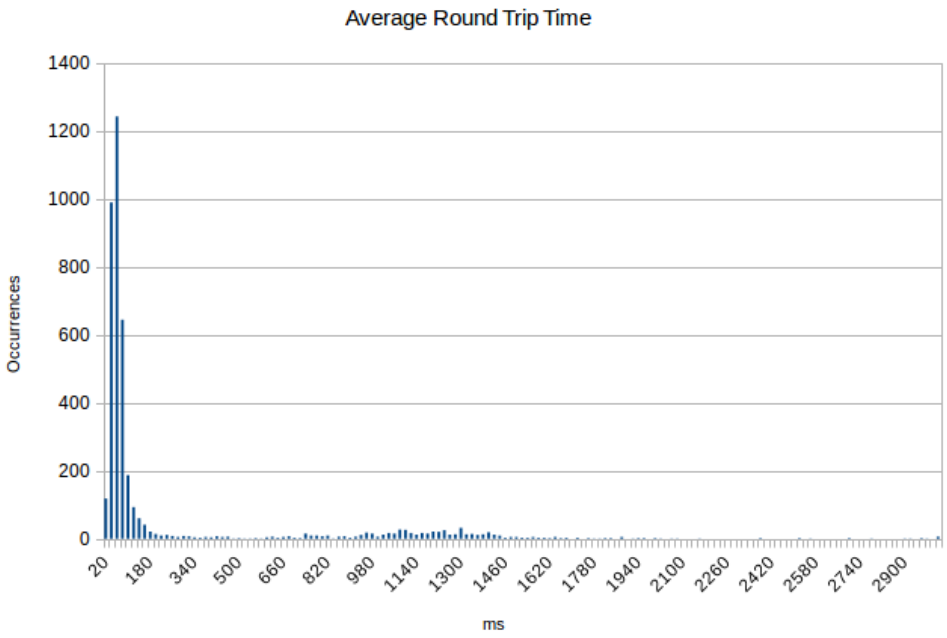



Figure 6.6: Representation of the distribution of average round trip time, on 89,500,000 measurements,on links connecting 4,116 nodes deployed on field

to a Fog Node, and the application running on a different Fog Node. In this scenario we expect a reduced effect of the constraint on the throughput, since, controlling the user interface and the application, it is possible to reduce the traffic between them. For this reason, in order to identify the best architecture for a nationwide network of Fog Nodes, we focused on latency and collected data from more than 89,500,000 measurements of round trip time, performed in last three years, between 4,116 nodes nodes deployed on field in order to manage Wi-Fi hotspots in Italy. We measured the round trip time in two ways: i) using a ping command and ii) opening a TCP session, the total distribution of the average collected values are shown in Figure 6.6. Then we compared these values, with the maximum latency acceptable in a User Virtual Environment. Since this parameter is related to the delay between an action on the

| | |
|---|---|
| Total number of measurements | 89,646,256 |
| Total number of Nodes involved in the measurements | 4,116 |
| Links with average round trip time < 100 ms | 77.25 % |
| Links with average round trim > 950 ms and < 1,400 ms | 10.03 % |

Table 6.1: Data gathered on field and deduced from the distribution of average round trip time of considered links.

user interface and the effect in the application, it involves the human perception of this delay. Even if a recent study [63] showed that human brain can identify images seen for as little as 13 ms, usually the acceptable delay threshold is considered at 100 ms, according to traditional studies [64][65][66]. Considered nodes are actually connected to Internet, that could be considered a full mesh network and, even if it was not possible to test all the links, the statistical distribution of latency clearly shows a very interesting situation. We observe a high concentration of links with round trip time below 100 ms, in particular we have a 77.25% of the measurement below this threshold. But the distribution appears to be shaped with a second local maximum around 1,200 ms involving the 10.03% of the links. Table 6.1 shows this data in details.

This particular shape of the distribution, with two local maximum, is probably due to the fact that nodes are in general connected with two different WAN technologies: optical fiber, with low latency, and xDSL where the latency is higher. This situation suggests that the Fog layer should not be designed with a unique clustering approach, but a hybrid architecture should be preferred. In particular a Dynamic Overlapping Cluster should be used with Fog Nodes with fiber Wide Area Network (WAN) links, in order to leverage the higher flexibility allowed by this approach. While the Fog Node connected with traditional xDSL links should be organized in Static Clusters.

## 6.7 Chapter Conclusions

Implementing Internet access network as flat platform where to deploy dynamically virtualized infrastructure, including applications and networking, introduces a change of vision involving all the network elements. In particular the scenario changes, from end devices connected to Cloud or Web applications, through the network, to user interfaces connected to dedicated environment, dynamically deployed directly on the network, able to provide services needed by users or Smart Objects. These environments can be created and destroyed as consequence of specific events, for example can be moved in order to follow a specific users, running on closest Fog Nodes, or different environment can be created depending on specific users' profiles.

This changes have impact on the architecture design and optimization of the Fog layer, access networks are made of. In this Chapter we provide a possible definition of the optimization problem for the Fog layer, a possible modeling of elements involved in the Fog layer, and two possible architecture: i) Static Clustering with Hierarchy, easily manageable and highly reliable, and ii) Dynamic Overlapping Clustering, more flexible and efficient.

We identified the latency between nodes as key parameter in order to choose one approach or the other. Performing more than 89,500,000 measurements on links connecting 4,116 nodes deployed on field, we found that, due to different WAN technologies, an hybrid approach should be preferred: a dynamic approach, which is more flexible and adaptable, is supported and should be preferred on Fog Nodes connected with low latency links (such as optical fiber), while a static approach is better for high latency Internet connection (such as xDSL), that are still a not negligible rate of all considered links.

# Chapter 7

# Conclusions

*The science of today*
*is the technology of tomorrow.*

– Edward Teller

## 7.1 Design

In this dissertation, we have introduced a new approach to Internet access networks in public spaces, such as Wi-Fi networks commonly referred to as Hotspot, based on Fog Computing (or Edge Computing), Software Defined Networking (SDN) and the deployment of Virtual Machines (VM) and Linux containers, on the edge of the network. According to this vision, we deploy specialized network elements, called Fog Nodes, on the edge of the network, able to virtualize the infrastructure and expose APIs to external applications. Calling these APIs, external applications are able to trigger and control the deployment of virtual environments, that include applications, in form of VMs or Containers, and SDN networking, directly on the access network. The main innovative points of this vision are:

- usage of technologies developed for different fields, such as data center and

IoT, in Internet access networks able to interconnect human users and smart objects;

- considering the access network as a flat virtualized infrastructure where it is possible to build the needed environments in real-time, including Virtual Machnies, Linux Containers and SDN networks;
- giving the control of the deployment of virtual environments to external application and in particular, to the clients of the platform, through mobile Apps running on connected devices. With this approach any event can be used as a trigger for the deployment of virtual environments.

## 7.2 Validation

In order to validate the proposed approach, we developed a preliminary study focused on bandwidth optimization. We analyzed data collected on field in existing highly crowded Wi-Fi Internet access network, in order to identify and measure the potential benefit of an infrastructure based on Fog Computing, compared to a traditional approach. Our results show that a significant portion (from 28% to 50%) of download data could be proactively managed by the Fog-based infrastructure.

## 7.3 Feasibility

In order to verify the feasibility of the envisioned platform and its compatibility with existing application, we deployed in laboratory, two testbeds: one related to the Wi-Fi Internet access system, and another related to IoT applications.

### 7.3.1 Wi-Fi Internet Access

We have developed the capability to trigger and control the deployment of full virtual environments, including networking and applications, directly at the edge of access networks, as a consequence of the interaction with a real authentication system, used

in Wi-Fi public Internet access locations. After the authentication process, we are able to deploy applications on Linux Containers and connect containers to user's devices and smart objects, according to the user's profile retrieved during the authentication, deploying SDN networks. We measured that it is possible to add these new capabilities without a significant degradation of the network performance, considering in particular the duration of the deployment of the virtual environment compared to the authentication process only.

### 7.3.2 IoT Hub and the Interaction between Fog and Cloud

In order to support and extend the concept of Fog-based network infrastructure, we introduce a specific use case, related to IoT, where it is possible to design advanced services based on the envisioned approach. Leveraging the virtualization capabilities of the Fog Nodes and the interaction between the Fog and the Cloud, we introduced the concept of *IoT Hub*, hosted on the Fog, with replicas in the Cloud, proving the feasibility and the advantage brought by the proposed architecture. In particular we show that the deployment of this Fog-based application is feasible, since it can run on devices usually used to manage Internet access networks, and that the interaction between Fog and Cloud improve security and reliability, without introducing delay that could affect performance.

## 7.4 Benefits Evaluation in Real Deployments

Redesigning provider's core applications, in order to leverage the presence of a Fog layer, it is possible to have great benefits. We investigated in particular the following aspects.

- **Authentication**: a system based on the interaction between clients and the Fog Node is able to improve the users' quality of experience, measurable in a three times larger number of connections and users.
- **Location analytics**: hosting on the fog node a virtual infrastructure able to locally process data provided by the Access Point infrastructure, we are able

to collect and elaborate locally, more than 500,000 events every day, in a real location, saving 300 MBytes of traffic every day from the Node on site to the Cloud, in order to produce real time analytics and daily report.

- **Computing resources optimization**: considering a set of 3,500 locations with Internet access service for guests, we measured how the nodes deployed on field are under-utilized, due to the highly variable level of workload. The nodes, already deployed for service management, have to be sized to manage the peak of workload, but they actually spend a big portion of their life time in idle. Redesigning the provider's core application for statistics, in oder to make it able to run on Fog Nodes, we show that it is possible to perform these task without involving the Cloud platform, using the computing and memory resources already deployed, for hotspot management, and underused in the traditional approach.

The last point is particularly interesting, since we measured that, redesigning the provider's internal applications, it is possible to use the resources, already deployed on field, in order to perform the task usually performed at the data center. This analysis introduces the possibility to offer services in a Cloud-less approach leveraging the capability and flexibility of Fog.

## 7.5 Architecture

After the validation analysis, the description of the proposed platform, the description of specific use cases and real deployment, we conclude our dissertation, focusing on the design and optimization of the architecture for the Fog-based Internet access network. Starting from the evolution of the role of access networks, from connecting end devices to hosting *dynamic user virtual environments*, allowed by Fog and SDN, we investigate the problem of the design and optimization of the Fog layer. We introduce the concept of *static* and *dynamic overlapping clusters*, and identify the latency between nodes as a key element that makes one approach better than the other. Analyzing more than 89,500,000 latency measurements between 4,116 Nodes

of a nationwide network of Wi-Fi hotspots, we found that modern Internet connections provide performance able to support Dynamic Clustering of Fog Nodes, which is more flexible and adaptable. However, collected results also show that a not negligible rate of high latency Internet links are still active, forcing to organize some Fog Nodes in Static Clusters. For this reason, on a nationwide network, an hybrid approach should be preferred, using Dynamic or Static Clusters, depending on the type of the Internet connections.

# Bibliography


[1] Here's why 'the internet of things' will be huge, and drive tremendous value for people and businesses. URL: http://www.businessinsider.com/growth-in-the-internet-of-things-2013-10?IR=T.

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things Characterization of Fog Computing. In *Proc. of the Mobile Cloud Computing, (MCC'12)*, pages 13–15, Helsinki, Finland, August 2012. ACM Press.

[3] Openfog consortium. URL: https://www.openfogconsortium.org/.

[4] Software-defined networking: The new norm for networks. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[5] Lxc. URL: https://en.wikipedia.org/wiki/LXCf.

[6] M. Aazam and H. Eui-Nam. Fog Computing and Smart Gateway Based Communication for Cloud of Things. In *Proc. of Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 464–470, Barcelona, Spain, August 2014. IEEE.

[7] S. Cirani, G. Ferrari, N. Iotti, and M. Picone. In *Fog Networking for 5G and IoT Workshop, in conjunction with 12th Annual IEEE International Conference*

*on Sensing, Communication and Networking (SECON 2015*, pages 464–470, Seattle WA, USA, June 2015.

[8] S. Sae Lor, Luis M. Vaquero, and P. Murray. The cloud in core networks. *IEEE Communications Letters*, (October 2012):1703–1706.

[9] Ivan Stojmenovic and Sheng Wen. The Fog Computing Paradigm: Scenarios and Security Issues. In *Proc. of the 2014 Federated Conference on Computer Science and Information Systems*, pages 1–8, Warsaw, Poland, September 2014. IEEE.

[10] H. Madsen, G. Albeanu, B. Burtschy, and F.L. Popentiu-Vladicescu. Reliability in the utility computing era: Towards reliable Fog computing. In *Proc. of Systems, Signals and Image Processing (IWSSIP), 2013 20th International Conference on*, pages 43–46, Bucharest, Romania, July 2013. IEEE.

[11] J. Zhu, D.S. Chan, M.S. Prabhu, P. Natarajan, H. Hao, and F. Bonomi. Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture. In *Proc. of Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 320–323, Redwood City CA, USA, March 2013. IEEE.

[12] Mugen Peng, Shi Yan, Kecheng Zhang, and Chonggang Wang. Fog Computing based Radio Access Networks: Issues and Challenges. (61222103), 2015. URL: `http://arxiv.org/abs/1506.04233`, `arXiv:1506.04233`.

[13] Swati Agarwal, Shashank Yadav, and Arun Kumar Yadav. An architecture for elastic resource allocation in Fog Computing. pages 201–207, 2015. `doi:10.090592/IJCSC.2015.615`.

[14] B. Ottenwalder, K. Koldehofe, K. Rothermel, and U. Ramachandran. Migcep: Operator migration for mobility driven distributed complex event processing. *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, pages 183–194, 2013.

[15] K. Hong, D. Lillethum, U. Ramachandran, B. Ottenwalder, and K. Koldehofe. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, pages 15–20, 2013.

[16] Rakesh Suryawanshi and Ganesh Mandlik. Focusing on Mobile Users at Edge and Internet of Things using Fog Computing. *International Journal of Scientific Engineering and Technology Research*, 04(17):3225–3231, 2015.

[17] Van-giang Nguyen, Kyoungjae Sun, and Younghan Kim. Fog Networking Architecture in Virtualized EPC Networks. In *Korean Institute of Communications and Information Sciences (KICS) Summer Symposium on ICT*, pages 428–429, 2015.

[18] Y Navaneeth Krishnan, Chandan N Bhagwat, and Aparajit P Utpat. Fog Computing- Network Based Cloud Computing. (Icecs):250–251, 2015.

[19] Luis M. Vaquero and Luis Rodero-Merino. Finding your Way in the Fog. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, 2014. URL: `http://dl.acm.org/citation.cfm?id=2677046.2677052`, `doi:10.1145/2677046.2677052`.

[20] M. Aazam, I. Khan, A.A. Alsaffar, and Eui-Nam Huh. Cloud of things: Integrating internet of things and cloud computing and the issues involved. In *Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on*, pages 414–419, Jan 2014. `doi:10.1109/IBCAST.2014.6778179`.

[21] M. Aazam, Pham Phuoc Hung, and Eui-Nam Huh. Smart gateway based communication for cloud of things. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6, April 2014. `doi:10.1109/ISSNIP.2014.6827673`.

[22] M. Yannuzzi, R. Milito, R. Serral-GraciÃă, D. Montero, and M. Nemirovsky. Key ingredients in an iot recipe: Fog computing, cloud computing, and more

fog computing. In *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 325–329, Dec 2014. doi:10.1109/CAMAD.2014.7033259.

[23] Docker. URL: https://www.docker.com/.

[24] B. I. Ismail, E. Mostajeran Goortani, M. B. Ab Karim, W. Ming Tat, S. Setapa, J. Y. Luke, and O. Hong Hoe. Evaluation of docker as edge computing platform. In *Open Systems (ICOS), 2015 IEEE Confernece on*, pages 130–135, 2015.

[25] Y. Xu, V. Mahendran, and S. Radhakrishnan. Sdn docker: Enabling application auto-docking/undocking in edge switch. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 864–869, April 2016.

[26] F. Ramalho and A. Neto. Virtualization at the network edge: A performance comparison. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, June 2016. doi:10.1109/WoWMoM.2016.7523584.

[27] R. Morabito. A performance evaluation of container technologies on internet of things devices. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 999–1000, April 2016. doi:10.1109/INFCOMW.2016.7562228.

[28] Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Broadband internet performance: a view from the gateway. In *ACM SIGCOMM computer communication review*, volume 41, pages 134–145. ACM, 2011.

[29] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki, and Akira Kato. The impact and implications of the growth in residential user-to-user traffic. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 207–218. ACM, 2006.

[30] Matti Siekkinen, Denis Collange, Guillaume Urvoy-Keller, and Ernst W Biersack. Performance limitations of adsl users: A case study. In *International Conference on Passive and Active Network Measurement*, pages 145–154. Springer, 2007.

[31] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman. On dominant characteristics of residential broadband internet traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 90–102. ACM, 2009.

[32] Yatendra Sahu, RK Pateriya, and Rajeev Kumar Gupta. Cloud server optimization with load balancing and green computing techniques using dynamic compare and balance algorithm. In *Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on*, pages 527–531. IEEE, 2013.

[33] Jiang Zhu, Douglas S Chan, Mythili Suryanarayana Prabhu, Preethi Natarajan, Hao Hu, and Flavio Bonomi. Improving web sites performance using edge servers in fog computing architecture. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 320–323. IEEE, 2013.

[34] Lorenzo Maggi, Lazaros Gkatzikis, Georgios Paschos, and Jérémie Leguay. Adapting caching to audience retention rate: Which video chunk to store? *arXiv preprint arXiv:1512.03274*, 2015.

[35] Kamran Riaz Khan, Zaafar Ahmed, Shabbir Ahmed, Affan Syed, and Syed Ali Khayam. Rapid and scalable isp service delivery through a programmable middlebox. *ACM SIGCOMM Computer Communication Review*, 44(3):31–37, 2014.

[36] Mung Chiang, Steven H Low, A Robert Calderbank, and John C Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[37] Plumgrid. URL: http://www.plumgrid.com/.

[38] Openstack. URL: https://www.openstack.org/.

[39] Io visor. URL: https://www.iovisor.org/.

[40] Zaafar Ahmed, Alizai Muhammad Hamad, and Affan A. Syed. InKeV: In-Kernel Distributed Network Virtualization for DCN. *ACM SIGCOMM Computer Communication Review*, (July), 2016. URL: https://ccronline.sigcomm.org/wp-content/uploads/2016/07/sigcomm-ccr-paper4.pdf.

[41] Global site speed overview: How fast are websites around the world? URL: https://analytics.googleblog.com/2012/04/global-site-speed-overview-how-fast-are.html.

[42] J. Falk and M. Kucherawy. Rpl: Ipv6 routing protocol for low-power and lossy networks. RFC 6650, June 2012. URL: https://tools.ietf.org/rfc/rfc6550.txt.

[43] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of ipv6 packets over ieee 802.15.4 networks. RFC 4944 (Proposed Standard), September 2007. Updated by RFCs 6282, 6775. URL: http://www.ietf.org/rfc/rfc4944.txt.

[44] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014. URL: http://www.ietf.org/rfc/rfc7252.txt.

[45] S. Cheshire and M. Krochmal. Multicast dns. RFC 6762 (Proposed Standard), February 2013. URL: http://www.ietf.org/rfc/rfc6762.txt.

[46] S. Cheshire and M. Krochmal. Dns-based service discovery. RFC 6763 (Proposed Standard), February 2013. URL: http://www.ietf.org/rfc/rfc6763.txt.

[47] Simone Cirani, Luca Davoli, Gianluigi Ferrari, Rémy Léone, Paolo Medagliani, Marco Picone, and Luca Veltri. A scalable and self-configuring architecture for service discovery in the internet of things. *IEEE Internet of Things Journal*, 1(5):508–521, 2014.

[48] Z. Shelby. Constrained restful environments (core) link format. RFC 6690 (Proposed Standard), August 2012. URL: `http://www.ietf.org/rfc/rfc6690.txt`.

[49] M. Nottingham. Web linking. RFC 5988 (Proposed Standard), October 2010. URL: `http://www.ietf.org/rfc/rfc5988.txt`.

[50] Z. Shelby, C. Bormann, and S. Krco. CoRE Resource Directory. Internet-Draft draft-ietf-core-resource-directory-01, Internet Engineering Task Force, December 2013. Proposed standard. URL: `http://tools.ietf.org/id/draft-ietf-core-resource-directory-01.txt`.

[51] A. Castellani, S. Loreto, A. Rahman, and T. Fossati. Guidelines for HTTP-CoAP Mapping Implementations. Internet-Draft draft-ietf-core-http-mapping-04, Internet Engineering Task Force, July 2014. URL: `http://tools.ietf.org/html/draft-ietf-core-http-mapping-04.txt`.

[52] Z. Shelby and M. Vial. CoRE Interfaces. Internet-Draft draft-ietf-core-interfaces-02, Internet Engineering Task Force, November 2014. URL: `https://tools.ietf.org/html/draft-ietf-core-interfaces-02.txt`.

[53] Matthias Kovatsch, Martin Lanter, and Zach Shelby. Californium: Scalable Cloud Services for the Internet of Things with CoAP. In *Proceedings of the 4th International Conference on the Internet of Things (IoT 2014)*, Cambridge, MA, USA, October 2014.

[54] C. Bormann, M. Ersue, and A. Keranen. Terminology for constrained-node networks. RFC 7228 (Informational), May 2014. URL: `http://www.ietf.org/rfc/rfc7228.txt`.

[55] The Contiki Operating System. URL http://www.contiki-os.org. URL: http://www.contiki-os.org.

[56] Raspberry Pi Foundation. URL: http://www.raspberrypi.org/.

[57] Microsoft. Microsoft azure - cloud platform. URL http://azure.microsoft.com/it-it/. URL: http://azure.microsoft.com/it-it/.

[58] Amazon. Amazon ec2. URL http://aws.amazon.com/ec2/. URL: http://aws.amazon.com/ec2/.

[59] Oracle Java VM. Java. URL https://www.oracle.com/java/index.html. URL: https://www.oracle.com/java/index.html.

[60] Vikram Adve Dinakar Dhurjati, Sumant Kowshik and Chris Lattner. Memory Safety Without Runtime Checks or Garbage Collection. In *Proc. Languages Compilers and Tools for Embedded Systems 2003*, San Diego, CA, June 2003. URL: http://llvm.cs.uiuc.edu/pubs/2003-05-05-LCTES03-CodeSafety.html.

[61] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote authentication dial in user service (radius). RFC 2865 (Draft Standard), June 2000. Updated by RFCs 2868, 3575, 5080, 6929. URL: http://www.ietf.org/rfc/rfc2865.txt.

[62] Cisco virtual wireless controller. URL: http://www.cisco.com/c/en/us/products/wireless/virtual-wireless-controller/index.html.

[63] Mary C Potter, Brad Wyble, Carl Erick Hagmann, and Emily S McCourt. Detecting meaning in rsvp at 13 ms per picture. *Attention, Perception, & Psychophysics*, 76(2):270–279, 2014.

[64] Stuart K Card, George G Robertson, and Jock D Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 181–186. ACM, 1991.

[65] Robert B Miller. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 267–277. ACM, 1968.

[66] Brad A Myers. The importance of percent-done progress indicators for computer-human interfaces. In *ACM SIGCHI Bulletin*, volume 16, pages 11–17. ACM, 1985.

# Acknowledgements

First, I would like to thank my advisor Prof. Gianluigi Ferrari for his unreserved support, guidance, patience and encouragement during this challenging period. He gave me all the freedom to act as an independent researcher, and at the same time, continuing to provide help when it was needed.

I am also deeply grateful to my colleagues and friends at Guglielmo and Caligoo, two Italian companies that made this research possible, supporting me and working with me in order to collect data from systems in production.

I would like to express my gratitude to Flavio Bonomi, one of the pioneer of Fog Computing, that provided me a clear vision of this change in network architecture. He introduced me amazing people, and brought me to the very edge of this evolution.

I would also thank all the people at Nebbolo Technology, that kindly welcomed me like a friend, at the beginning of their exciting adventure, and allowed me to spend time with them learning a lot, professionally and personally.

I also owe much to the staff at PLUMgrid, where I was so fortunate to collaborate with so many wonderful people, always ready to answer and provide support. I would like to thank Pere Monclus, one of the most brilliant people I ever met, he has the exceptional capability to make clear, complex problems, thank you to his clear vision of technology and the incredible rapidity in understanding complicated concepts.

Finally, I owe my deepest thanks to my wife, my daughters and my family, that made this adventure possible.