



# Integrating SDN and NFV with QoS-Aware Service Composition

Valeria Cardellini<sup>1</sup> , Tihana Galinac Grbac<sup>2</sup> , Andreas Kassler<sup>3</sup> ,  
Pradeeban Kathiravelu<sup>4</sup> , Francesco Lo Presti<sup>1</sup> , Antonio Marotta<sup>3</sup> ,  
Matteo Nardelli<sup>1</sup> , and Luís Veiga<sup>4</sup>

<sup>1</sup> University of Rome Tor Vergata, Rome, Italy  
{cardellini,nardelli}@ing.uniroma2.it, lopresti@info.uniroma2.it

<sup>2</sup> Faculty of Engineering, University of Rijeka, Rijeka, Croatia  
tihana.galinac@riteh.hr

<sup>3</sup> Karlstad University, Karlstad, Sweden  
andreas.kassler@kau.se, antonio.marotta@live.it

<sup>4</sup> INESC-ID Lisboa/Instituto Superior Técnico,  
Universidade de Lisboa, Lisbon, Portugal  
pradeeban.kathiravelu@tecnico.ulisboa.pt, luis.veiga@inesc-id.pt

**Abstract.** Traditional networks are transformed to enable full integration of heterogeneous hardware and software functions, that are configured at runtime, with minimal time to market, and are provided to their end users on “as a service” principle. Therefore, a countless number of possibilities for further innovation and exploitation opens up. Network Function Virtualization (NFV) and Software-Defined Networking (SDN) are two key enablers for such a new flexible, scalable, and service-oriented network architecture. This chapter provides an overview of QoS-aware strategies that can be used over the levels of the network abstraction aiming to fully exploit the new network opportunities. Specifically, we present three use cases of integrating SDN and NFV with QoS-aware service composition, ranging from the energy efficient placement of virtual network functions inside modern data centers, to the deployment of data stream processing applications using SDN to control the network paths, to exploiting SDN for context-aware service compositions.

## 1 Introduction

Software-Defined Networking (SDN) is a new paradigm that provides programmability in configuring network resources. It introduces an abstraction layer on the network control layer that allows runtime and ad-hoc network reconfiguration. Therefore, it enables to adapt at runtime not only physical network resources but also software services that compose complex services delivered to end users. Such a new network feature thus provides a valuable mechanism to be exploited in the modeling of QoS-aware service compositions integrating services from various networks. This paradigm has been successfully incorporated into the virtualization of the telecommunication network and an architecture concept

called Network Function Virtualization (NFV), where virtual network functions are interconnected into service compositions to create communication services.

Traditional networks that have been designed for yesterday peak requirements are inefficient to cope with nowadays massive communication traffic injected by a large number of users (e.g., billions of devices in the Internet of Things). The main obstacle of traditional networks to provide full exploitation of their resources and accelerate innovation is caused by the lack of integration of the variety of hardware and software appliances. Moreover, the lack of standardized interfaces make network management costly and slow adapting to modern trends, and user demands [14, 20, 27].

Within the 5G network, SDN and NFV are the two key technologies introduced as enablers [33]. In future networks, the optimal cost is achieved through dynamic and self-adaptive deployment on a network infrastructure which is continuously controlling its performances and autonomously managing its resources. The primary goal of such a dynamic and autonomous deployment is to accomplish and maintain the quality of service (QoS) requirements of complex services. By adopting SDN and NFV for the composition of complex services, Software-Defined Service Composition (SDSC) [21] separates the execution of service compositions from the data plane of the overall system.

SDSC facilitates the integration and interoperability of more diverse implementations and adaptations of the services. A reliable execution of service composition can be guaranteed through the network management capabilities offered by SDN, in finding the best alternative among various service implementations and deployments among the multiple potential services deployments for the service composition execution. SDSC thus offers an increased control over the underlying network, while supporting the execution from various traditional web service engines and distributed frameworks.

There are various modeling approaches for QoS-aware service composition which have been proposed so far. With the introduction of a programmable approach to implement and use network resources, we should investigate performance modeling approaches that jointly consider all network layers and their composite behavior and outputs. Therefore, the contribution of this chapter is to analyze the integration of SDN and NFV in modeling the performance of service compositions and investigate possible side effects that can arise from their composite interactions. To this end, we present three different use cases of integrating SDN and NFV with QoS-aware service composition, ranging from the energy efficient placement of virtual network functions inside modern data centers, to the deployment of data stream processing (DSP) applications using SDN to control the network paths, to exploiting SDN for context-aware service compositions.

In the upcoming sections of this chapter, we continue to discuss the benefits and use cases of integrating SDN and NFV with QoS-aware service composition. Section 2 provides an overview of the basic concepts: SDN, NFV, and service compositions. Section 3 discusses the energy-efficient green strategies enabled by the integration of SDN and NFV with service compositions. Section 4 focuses

on a specific example of composite service - represented by DSP applications - and elaborates on the integration of a DSP framework with an SDN controller, showing a full vertical integration of the application and network layers. Section 5 discusses how SDN can offer context-aware service compositions. Finally, we discuss the benefits and open research issues in QoS-aware service compositions in Sect. 6 and conclude the chapter by identifying future research directions in Sect. 7.

## 2 Overview of Basic Concepts

A traditional network architecture divides Telco/Network operators from Internet Service Providers (ISPs) and Content Providers. Services are provided over highly specialized technologies which limit their full exploitation by end users. A new network architecture that is proposed for future networks introduces new abstraction layers with standardized interfaces that would enable Telco/Network Providers, ISPs, and Content Providers to provide their services over the web, independently from the underlying network. The vision of future networks is to provide their users with complex services that result from the autonomous composition of simple, possibly legacy, elementary services. Such a service orientation has also been recently reaffirmed for the next decade in the Service Computing manifesto [6], that call for the widespread adoption of service computing.

### 2.1 Introduction to NFV

The basic concept of NFV is to apply Cloud computing technologies to realize telecommunication applications. NFV revolves around the concept of virtualization, which enables to run multiple systems in isolation on a single hardware system. The exploitation of virtualization allows to decouple network functions from the related (dedicated) hardware [17]. In other words, a software implementation of different network functions (e.g., modulation, coding, multiple access, firewall, deep packet inspection, evolved packet core components) can be deployed on top of a so-called hypervisor, which runs on commercial off-the-shelf servers instead of dedicated hardware equipment. The hypervisor provides for virtualization and resource management (e.g., scheduling access to CPU, memory, and disk for the network functions). In addition, an orchestration framework needs to be in place, so to combine different virtual functions to obtain higher layer service chains implementing the end-to-end service. Moreover, the orchestration framework manages the deployment (e.g., which virtual function to place on what physical server) and the life cycle of the virtual network functions, including the management of their scalability. The latter comprises several tasks, among which monitoring performance, scaling either vertically or horizontally resources (i.e., either acquiring more powerful computing resources or spawning more replicas of the same virtual network function and load balancing among them).

Consequently, Virtual Network Functions (VNFs) are different from classical server virtualization technologies because VNF may form service chains composed of multiple virtual network functions, that exchange traffic which may be

deployed on one or multiple virtual machines running different network functions and replacing thus a variety of hardware appliances [33]. Such software implementation of network functions is easily portable among different vendors and may coexist with hardware-based platforms. Thus, the main benefits provided are a reduction of capital and operational expenditures, offering a reduced time-to-market as well as scalability to different resource demands.

However, with the introduction of VNFs, additional problems may arise, such as increased complexity. Additional interfaces need to be defined and maintained (e.g., between the hypervisor and the orchestration system), which leads to more complex system design. In addition, as applications can have strict requirements in terms of latency, performance guarantees are more difficult to be satisfied. This is because a given implementation of a VNF may perform differently when deployed on different hardware. For example, the deployment of I/O intensive VNF (e.g., a home subscriber service) on a server equipped with a standard HDD may lead to lower performance than the one resulting from a deployment on a server equipped with an SSD or NV-RAM. Consequently, new benchmarking tools are required that allow correlating the performance of a given VNF when deployed on a given hardware with a certain configuration.

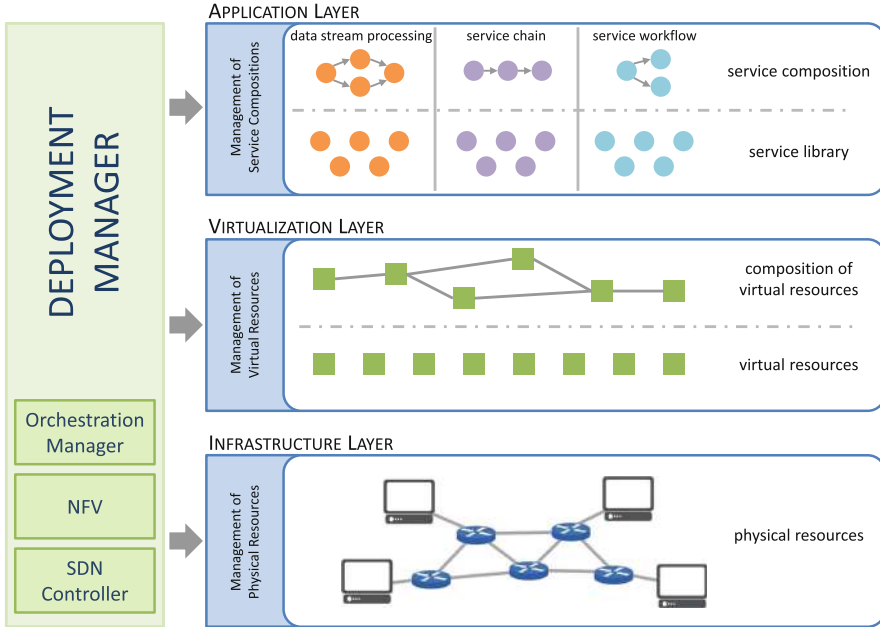
## 2.2 Introduction to Service Composition Using SDN

The second enabling technology is SDN, which separates the network control plane from the infrastructure (data) plane [31]. It involves logical centralization of network intelligence and introduces abstraction of physical networks from the applications and services via standardized interfaces. SDN is considered an enabling technology for high volumes of traffic flows and responds “at runtime” on dynamic demand for network resources by avoiding time-consuming and costly manual reconfiguration of the network. Thus, it increases network resource exploitation and decreases time to market. Furthermore, service-orientation is introduced to enable the runtime discovery and deployment of services. When combined with NFV and SDN technologies, this feature can significantly improve the efficiency of network operations.

Figure 1 presents a high-level architecture, emphasizing three distinct management layers that are coordinated by a vertical deployment manager to provide possibly coordinated QoS-aware decisions about service deployment.

At the *infrastructure layer*, routers and switches are distributed over the network topology. These devices have their logical representation that is used for control and management purposes. Decisions of centralized network control are transferred over the standardized physical interfaces to operate over devices in this layer.

Network resources are virtualized in the *virtualization layer*. Each virtual resource has its logical representation that enables efficient management. The virtual resources may be interconnected into a graph-like topology. Again, autonomous decisions about their interconnection and placement are subject of the management entity at this layer.



**Fig. 1.** High-level network overview.

Finally, at the *application layer* a number of basic component services are available in distributed data centers and exposed in service libraries. The complex services may be composed of many basic services that are accessible through service registries and can be composed on the basis of different goals. In the three use cases, we present later in this chapter, we consider network service chains, Web service and eScience workflows, and data stream processing (DSP) applications. A network service chain allows assembling services out of multiple service functions typically using basic patterns for service composition, e.g., a sequence of VNFs, with one or multiple instances needed for each VNF. Web services and eScience workflows usually organize their component services using more complex workflow patterns, e.g., conditional choice, loops, and fork-and-join. Finally, a DSP application is represented as a directed acyclic graph, that can be seen as a workflow diagram.

A service composition deployment on top of SDN allows cross-layer optimizations, as the services interact with the SDN controller through its northbound Application Programming Interface (API) protocols and using REpresentational State Transfer (REST) [39], Advanced Message Queuing Protocol (AMQP) [42], and Message Queue Telemetry Transport (MQTT) [32] transport protocols. On the other hand, the SDN controller orchestrates the data center network that the services are deployed on, through its southbound API protocols such as OpenFlow [28] and NetConf [16]. Such a cross-layer optimization supported by SDN allows QoS guarantees at the service and network levels.

NFV and SDN do not rely on each other. NFV is providing flexible infrastructure, while SDN software can run and can provide flow-based configuration of network functions. Both technologies, when used in cooperation, can offer enhanced QoS guarantees. In such new network architecture, the network logic is abstracted on several layers of abstraction. The management decisions of each layer may have reflections on the QoS provided by the network. Thus, the selection of collected management decisions within *deployment manager* should balance between flexibility provided at each level of network abstraction and optimal QoS.

An ongoing standardization endeavor is Next Generation Service Overlay Network (NGSON), aiming to establish a collaborative framework among the stakeholders from various networks and technology paradigms in order to unify their vision on common service delivery platform. Thus, the end-user need for complex service delivery across the network borders would be satisfied. The standard aims to identify self-organizing management capabilities of NGSON including self-configuration, self-recovery, and self-optimization of NGSON nodes and functional entities.

### 2.3 Overview of Use Cases

In this chapter, we will look into three illustrative use cases of integrating SDN and NFV with QoS-aware service composition.

Section 3 presents an overview on green strategies for VNF embedding, supported by SDN and NFV. Here, the key idea is to manage the NFV infrastructure, namely the composition of compute and networking resources including servers and networking equipment in an energy efficient way. By powering down unused servers and switches, the total energy of the infrastructure can be minimized. Important questions to ask are then what is the minimum number of servers, switches, and links that are necessary in order to provide the SLA desired for the service chains that need to be embedded into the physical network and compute infrastructure, where to place the functions and how to route the service chain traffic in order to find a balance between energy efficiency, performance and SLA.

Section 4 presents how the integration of an SDN controller with a DSP framework allows to adjust the network paths as per-application needs in the QoS-aware deployment of DSP applications on the computing and network resources. In the proposed integrated framework, SDN is used to expose to the DSP framework the network topology and network-related QoS metrics. Such information is exploited in a general formulation of the optimal placement problem for DSP applications, which jointly addresses the selection of computing nodes and of network paths between each pair of selected computing nodes.

We define services that access, process, and manage Big Data as big services. They pose computation and communication challenges due to their complexity, volume, variety, and velocity of Big Data they deal with. Moreover, they are often deadline-bound and mission-critical. Each big service is composed

of multiple services to be able to execute it in the Internet-scale at the distributed clouds. Such a componentization of big service improves its resilience and latency-awareness. For example, consider a big service for weather forecast. It consists of various services including sensor data retrieval, data analysis services, and prediction. These component services are inherently distributed, including the ones that manage the actuators and the sensors in land, sea, and satellites. By leveraging the SDN and NFV paradigms, SDSC ensures an efficient service composition from the replicated and globally distributed services. Section 5 discusses how SDSC leverages SDN to build and efficiently execute complex scientific workflows and business processes as service compositions.

### 3 Green Strategies for VNF Embedding

Next generation 5G networks will rely on distributed virtualized datacenters to host virtualized network functions on commodity servers. Such NFV will lead to significant savings in terms of infrastructure cost and reduced management complexity. Virtualization inside modern datacenters is a key enabler for resources consolidation, leading towards green strategies to manage both compute and network infrastructures where VNFs are hosted. However, green strategies for networking and computing inside data centers, such as server consolidation or energy aware flow routing, should not negatively impact on the quality and service level agreements expected from network operators, given that enough resources exist. For example, given two different resource allocation strategies, one focusing on performance while the other focusing on energy efficiency, while both strategies may lead to a resource allocation that satisfies user demands and SLAs, a green strategy does so by minimizing the energy consumption. Once fewer resources are available than requested, green strategies should guide the resource allocation processes towards operational points that are more energy friendly.

Important tools available for Cloud Operators are server consolidation strategies that migrate Virtual Machines (VMs) towards the fewest number of servers and power down unused ones to save energy. As VNFs are composed of a set of VNF Components (VNFC) that need to exchange data over the network under capacity and latency constraints, the networking also plays an important part. By using SDN, one can dynamically adjust the network topology and available capacity by powering down unused switch ports or routers that are not needed to carry a certain traffic volume [19], thus consuming the least amount of energy at a potential expense of higher latency. Green strategies try to place the VNFC onto the fewest amount of servers and to adjust the network topology and capacity to match the demands of the VNFCs while consuming the least amount of energy for operating the VNF Infrastructure. Such design of the VNF placement and virtual network embedding can be formulated as a mathematical optimization problem, and efficient heuristics can be designed to quickly solve the problem.

We can consider the Virtualized Compute and Network Infrastructure as the set of hardware resources (which is comprised of the compute and network

infrastructure) that is hosting a certain number of VNFs inside a virtualized data center. The virtualized data center can be geo-distributed to serve different users at different locations using the lowest cost in terms of energy, network, etc. We assume that each VNF is made of a set of service chains, which is a group of VNFC which have a set of traffic demands and a maximum tolerable latency allocated towards them. More precisely, the traffic demands specify how much traffic, between two adjacent services in a chain, the first sends to the second one. A service needs resources, e.g., in terms of CPU, memory, disk, and so on, to process packets and then forward the processing results to the next component of the chain.

The latency of a service chain is the sum of the experienced delays on the used paths, on which all the demands of the service chain are forwarded. It also includes the host internal processing related latency, which may be different for different architectural setups. For example, using standard Linux networking approach leads to much higher latency and less available capacity compared to using the recently developed approaches for user-mode packet forwarding and processing based on proprietary techniques, such as Intel's Data Plane Development Kit (DPDK).<sup>1</sup> Similarly, Single Root Input/Output Virtualization (SR-IOV<sup>2</sup>) is an extension to the PCI-express standard that allows different virtual machines (VMs) hosting the VNFs in a virtual environment to share a single network card over fast PCI-express lanes. Consequently, the additional latency for VNF packet processing depends on the virtualization technology used in the servers, which may be different for different server types. In addition, when two VNFC are placed on the same server, there is also a not negligible overhead when forwarding the packets from one component to another (after proper processing) and this overhead (and thus the additional latency and capacity limits) also depends on the virtualization technology used.

In the following, we assume that we have available a set  $J$  of servers and a network graph  $G(N, E)$ , where  $N$  represents the set of network nodes and  $E$  denotes the links among them. Given the family of service chains, which are defined as a specific number of traffic demands between couples of a subset  $\bar{V} \subset V$  out of all VNFC, the objective of the problem is to allocate all the VNFCs on the servers and to find the network routes that satisfy the traffic demands while minimizing the overall power consumption  $P_{VNI}$  of the Virtual Network Infrastructure, which is the sum of the power consumption of the compute ( $P_{\text{servers}}$ ) and network infrastructure ( $P_{\text{switches}}$ ), given the latency, resource and bandwidth capacity budgets:

$$\min f = P_{VNI} = P_{\text{servers}} + P_{\text{switches}} \quad (1)$$

The key idea for developing green strategies is to place the network functions on the minimum number of servers and use the minimum number of highly energy

<sup>1</sup> <https://software.intel.com/en-us/networking/dpdk>.

<sup>2</sup> <https://www.intel.com/content/dam/doc/white-paper/pci-sig-single-root-io-virtualization-support-in-virtualization-technology-for-connectivity-paper.pdf>.



efficient network nodes that can serve the required capacity. Consequently, all unused servers and switches can be powered down to reduce energy consumption.

### 3.1 Power Model Examples for Compute and Network Infrastructure

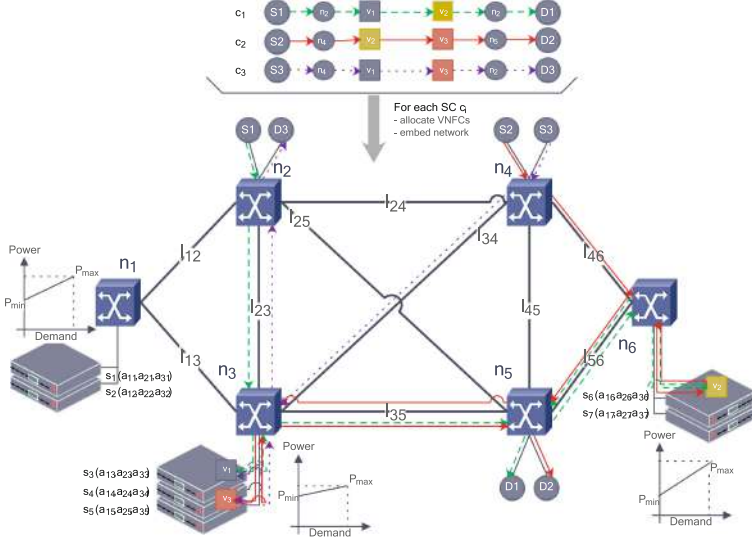
Several power models have been proposed for the compute infrastructure. Typically, they assume that the CPU of a server is the most power hungry component [35], and consequently most models just consider the power consumption due to CPU load. In general, the relationship between server power consumption and CPU utilization is linear [24,36] with some small deviations that are due to processor architecture, CPU cache related aspects and compiler optimizations leading to a different CPU execution. For performance modeling of green server and network consolidation strategies, we can simplify that for each server  $j$  there is a unique idle power consumption  $P_{idle,j}$ , which denotes the energy required by the server when it is just powered on and does not run any compute (except the basic Operating System and management services). The maximum power consumption  $P_{max,j}$  denotes the power consumed by the given server when all the CPU cores are under full load. In between the two extreme cases, the power consumption follows a linear model dependent on the CPU utilization.

The network related power consumption can also be simplified to make it tractable in numerical models. For example, the work in [5] assumes that for network switches there are two main components that impact the total power consumption. A static and constant power is required to power the chassis and the line cards, which is independent of the traffic that the switch serves and the number of ports used. In addition, depending on the number of ports per line rate are powered on, there is a dynamic power consumption, which also depends on the link speed the port is using (e.g., 1 Gbps or 10 Gbps) and the dynamic utilization of the ports. The power consumption also depends on the switch manufacturers: the work by Heller et al. [19] provides an overview on the power consumption of three different 48-port switch models. For example, one switch has a power consumption of 151 W when the switch is idle and all the ports are powered down, while it increases to 184 W when all the ports are enabled and to 195 W when all the ports serve traffic at 1 Gbps. As one can see, just powering on a switch requires the highest amount of power, while powering on additional ports does not add much to the total power consumption while the traffic dependent power consumption is almost negligible. Consequently, many green strategies try to conserve energy by powering down unused switches and power down unused ports.

### 3.2 Illustrative Example

In this section, we provide a simple example to illustrate the problem in Fig. 2. We assume there are seven servers (labeled from  $s_1$  to  $s_7$ ), each one with its own dedicated power profile specified by a given idle power  $P_s^{min}$  and maximum power consumption  $P_s^{max}$ . Each server has limited resources in terms of,

e.g., CPU, memory and disk capacities. To be more specific, server  $s_i$  has available  $a_{1i}$  CPU,  $a_{2i}$  RAM and  $a_{3i}$  DISK. Each server is connected to a specific router (e.g., the Top of Rack Switch in case of a Data Center).



**Fig. 2.** The joint VNF placement and network embedding problem [26].

Each link that connects the servers to the switch or the switches with each other has a dedicated capacity and latency. In the example, the latency for the link between  $n_1$  and  $n_2$  is denoted as  $l_{12}$ . The latency has typically several components. The first one is the latency due to the capacity that the links operate, which is constant. There is also latency due to the virtualization technique applied, which depends on the load of the servers and other configurations (e.g., CPU cache misses). Furthermore, there is a load-dependent latency due to queuing, which is typically non-linear. However, under low load, such latency can be assumed to be linearly increasing, while under higher load, we can use a piecewise approximation to model the latency due to traffic being routed over the interface. In addition, each link has a dedicated capacity (omitted from Fig. 2 due to complexity).

In the given example, we should embed into this NFV Infrastructure three service chains ( $c_1$ ,  $c_2$  and  $c_3$ ). Each service chain has its unique latency bound, a dedicated traffic source  $S_1$ ,  $S_2$  and  $S_3$  and sink  $D_1$ ,  $D_2$  and  $D_3$ . For example, in 5G for machine-to-machine traffic low latency should be enforced while for multimedia traffic latency bounds could be more relaxed. Also, the model can be specified flexibly to model also control plane related service chains, with more stringent delay requirements. In the example, we have three different VNFCs ( $v_1$ ,  $v_2$  and  $v_3$ ) and we assume that the traffic source for  $c_1$  is the Sender  $S_1$ ,

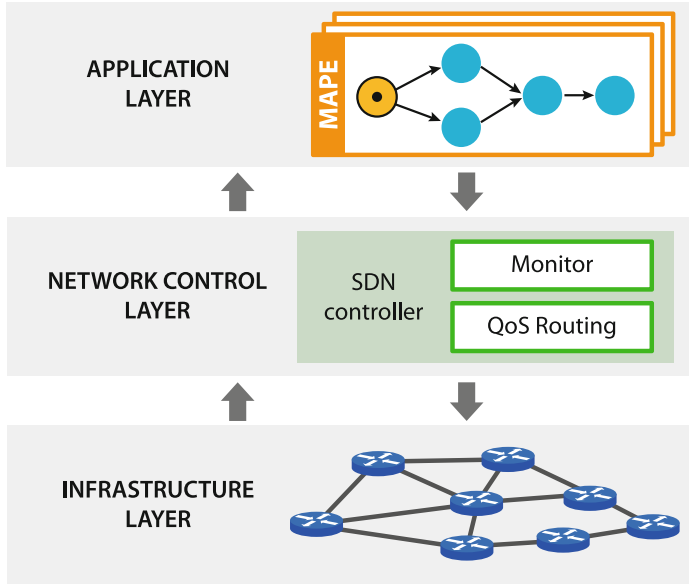
which is connected to router  $n_2$  and injects a certain volume of traffic into the service chain towards  $v_1$ . Then,  $v_1$  processes the packets (for which it needs resources such as CPU, memory, and disk) and forwards the processed traffic (which may have a different volume than the one injected) towards VNFC  $v_2$ , which again processes it and forwards a certain volume to the destination  $D_1$  that is connected to router  $n_2$ .

Note that Fig. 2 assumes additional source/sink nodes where traffic for a service chain is created/terminated. The figure shows an example of joint VNF placement and network embedding into the physical substrate network. VNFC  $v_1$  would be placed onto server  $s_3$ ,  $v_3$  onto server  $s_4$ , and so on. Servers hosting no VNFC would be powered down ( $s_1$ ,  $s_2$ ,  $s_5$ ,  $s_7$ ) together with all the nodes not carrying any traffic ( $n_1$ ).

## 4 Integrating SDN into the Optimal Deployment of DSP Applications

In the section, we present a use case of integrating SDN with QoS-aware service composition that focuses on Data Stream Processing (DSP) applications. The advent of the Big Data era and the diffusion of the Cloud computing paradigm have renewed the interest in DSP applications, which can continuously collect and process data generated by an increasing number of sensing devices, to timely extract valuable information. This emerging scenario pushes DSP systems to a whole new performance level. Strict QoS requirements, large volumes of data, and high production rate exacerbate the need for an efficient usage of the underlying infrastructure. The distinguishing feature of DSP applications is their ability to processing data on-the-fly (i.e., without storing them), moving them from an operator to the next one, before reaching the final consumers of the information. A DSP application can be regarded as a composition of services [1] with real-time processing issues to address. It is usually modeled as a directed acyclic graph (DAG), where the vertexes represent the processing components (called application *operators*, e.g., correlation, aggregation, or filtering) and the edges represent the logical links between operators, through which the data streams flow.

To date, DSP applications are typically deployed on large-scale and centralized (Cloud) data centers that are often distant from data sources [18]. However, as data increase in size, pushing them towards the Internet core could cause excessive stress on the network infrastructure and also introduce high delays. A solution to improve scalability and reduce network latency lies in taking advantage of the ever-increasing presence of near-edge/Fog computing resources [4] and decentralizing the DSP application, by moving the computation to the edges of the network close to data sources. Nevertheless, the use of a diffused infrastructure poses new challenges that include network and system heterogeneity, geographic distribution as well as non-negligible network latencies among distinct nodes processing different parts of a DSP application. In particular, this latter aspect could have a strong impact on DSP applications running in latency-sensitive domains.



**Fig. 3.** DSP framework with SDN controller integration.

To address these challenges, we have proposed the solution depicted in Fig. 3 and named *SDN-integrated DSP Framework* (for short, SIFD), which combines and integrates a DSP application framework with an SDN controller. To this end, we have:

- extended the architecture of Apache Storm, a well known open-source DSP framework, by designing, developing, and integrating few key modules that enable a distributed QoS-aware scheduler architected according to the MAPE (Monitor, Analyze, Plan, and Execute) reference model for autonomic systems [7, 8];
- designed, developed and implemented the controller logic for standard SDN controller and the associated API to provide network monitoring and dedicated stream routing configuration in an SDN network.

The proposed solution represents a full vertical integration of the application and network layers. The resulting architecture is highly modular and capable of taking full advantage of the SDN paradigm in modeling and optimizing the performance of Fog-based distributed DSP applications. In particular, SIFD enables the cross-layer optimization of the Fog/Cloud and SDN layers, whereby the SDN layer exposes to the upper layer the network topology and QoS metrics. This allows the optimal deployment of DSP applications by exploiting full knowledge of the computational and network resources availability and status. In this setting, an optimal deployment algorithm determines not only the application components placement on the underlying infrastructure but also the network paths between them.

For the sake of comparison with a non-SDN based solution, the proposed solution is backward compatible with legacy IP network, whereby network paths are solely determined by the underlying routing protocol and cannot be adjusted as per-application needs, thus providing no control by the DSP framework. In this setting, the DSP manager can at most monitor the network performance between candidate endpoints (see, e.g., [13] for a scalable network monitoring service) and determine operator placement on the underlying infrastructure by taking account the observed network delays.

#### 4.1 The SIDF Architecture

SIDF uses a layered architecture to combine a DSP framework with an SDN controller (Fig. 3). The layered infrastructure enforces separation of concerns and allows to obtain a loosely coupled system. Each layer realizes new functionalities on top of lower-level services and exposes them as a service to the higher layer. SIDF comprises three main layers: infrastructure layer, network control layer, and application layer.

At the lowest level, the infrastructure layer and the network control layer represent the classical SDN network. Specifically, the *infrastructure layer* comprises network equipment, such as SDN devices and legacy IP devices. The former enables to monitor and dedicate communication paths, whereas the latter only exposes paths as black-boxes, resulting from their routing protocol.

The *network control layer* manages the heterogeneity of network devices and controls their working conditions. SIDF includes a network controller that realizes two functionalities: monitor and QoS routing. The monitoring components periodically observe the network so to extract metrics of interest; to limit the footprint of monitoring operations, we only retrieve network delays among network devices and computing nodes. Observe that these monitoring operations can be realized in an SDN controller assisted manner as proposed in [41], where the SDN controller periodically sends probes on links to measure their transferring delays, or in a distributed manner, where neighbor SDN devices autonomously compute latencies. As a result, the network control layer can expose a view of the infrastructure as a connected graph (or *network graph*), where network devices and computing nodes are interconnected by network links; the latter are labeled with monitoring information (e.g., network latency). Observe that, with legacy IP devices, the link between two network nodes represents the logical connectivity resulting from the routing protocols. As regards the QoS routing functionalities, the SDN controller allows installing dedicated stream routing configurations in the underlying infrastructure. Leveraging on the exposed network graph, the application layer of SIDF can instruct the network to route streams on specific paths, according to application needs. For example, the application might require to route data using either a best-effort path, the path that minimizes the number of hops, or the one that minimizes the end-to-end delay between two computing nodes.

The *application layer* includes the DSP framework, which abstracts the computing and network infrastructure and exposes to users simple APIs to execute DSP applications. Many DSP frameworks have been developed so far. Nevertheless, most of them have been designed to run in a clustered environment, where network delays are (almost) zero [9]. Since in an infrastructure with distributed computing resources (like in the Fog computing environment) network delays cannot be neglected, SIDF includes a custom distributed DSP framework that conveniently optimizes the execution of DSP applications. This framework, named Distributed Storm [8], has been implemented as an extension of Apache Storm [40], one of the mostly adopted open-source DSP frameworks. Distributed Storm oversees the deployment of DSP applications, which can be reconfigured at runtime so to satisfy QoS requirements (e.g., maximum application response time). To this end, the framework includes few key modules that realize the MAPE (Monitor, Analyze, Plan, and Execute) control cycle, which represents the reference model for autonomic systems [7, 8]. During the execution of MAPE phases, Distributed Storm cooperates with the other layers of SIDF so to jointly optimize the application deployment and the QoS-aware stream routing. Specifically, during the Monitor phase, the framework retrieves the resource and network conditions (e.g., utilization, delay) together with relevant application metrics (e.g., response time). Network conditions are exposed by the network control layer. During the Analyze phase, all the collected data are analyzed to determine whether a reconfiguration of the application deployment should be planned. If it is worth to reconfigure the application as to improve performance (or more generally, to satisfy QoS requirements), in the Plan and Execute phases the framework first plans and then executes the corresponding adaptation actions (e.g., relocate the application operators, change the replication degree of operators). The Plan phase determines the optimal deployment problem, whose general formulation is presented in the next section. If a reconfiguration involves changing the stream routing strategy, the Execute phase also interacts with the network control layer, so to enforce new forwarding rules.

## 4.2 DSP Deployment Problem

We now illustrate the optimal deployment problem for DSP applications with QoS requirements. We provide a general formulation of the optimal placement problem for DSP applications which jointly addresses the operator placement and the data stream routing by modeling both the computational and networking resources. A detailed description of the system model can be found in [9].

For a DSP application, solving the deployment problem consists in determining for each operator  $i$ :

1. the operator placement, that is the computational node where to deploy the operator  $i$ ;
2. the network paths that the data streams have to traverse from an operator  $i$  to each of the downstream operator  $j$ .

For the sake of simplicity, here we do not consider the operator replication problem, that is the determination of the number of parallel replicas for each operator to deploy in order to sustain the expected application workload. Nevertheless, the following arguments can be easily extended to the general case, e.g., using the approach presented in [10].

A deployment strategy can be modeled by associating to each operator  $i$  a vector  $\mathbf{x}^i = (x_1^i, \dots, x_R^i)$ , where  $x_u^i = 1$ , with  $u \in \{1, \dots, R\}$  representing a computing resource, if the operator  $i$  is placed on the node  $u$  and 0 otherwise. Similarly, for each stream  $(i, j)$  from operator  $i$  to operator  $j$ , the vector  $\mathbf{y}^{(i,j)} = (y_1^{(i,j)}, \dots, y_\Pi^{(i,j)})$ , where  $y_\pi^{(i,j)} = 1$ , with  $\pi \in \{1, \dots, \Pi\}$  representing a network path, if the data stream from operator  $i$  to operator  $j$  follows the path  $\pi_h$  and 0 otherwise.

The Operator Placement and Stream Routing (OPSR) problem takes the following general form:

$$\begin{aligned}
 & \min F(\mathbf{x}, \mathbf{y}) \\
 & \text{subject to: } Q^\alpha(\mathbf{x}, \mathbf{y}) \leq Q_{\max}^\alpha \\
 & \quad Q^\beta(\mathbf{x}, \mathbf{y}) \geq Q_{\min}^\beta \\
 & \quad \mathbf{x}, \mathbf{y} \in A
 \end{aligned} \tag{2}$$

where  $\mathbf{x} = (\mathbf{x}^{i_1}, \dots, \mathbf{x}^{i_n})$  is the vector of the operator deployment binary variables and  $\mathbf{y} = (\mathbf{y}^{(i_1, j_1)}, \dots, \mathbf{y}^{(i_n, j_n)})$  is the vector of the network path variables.

Here,  $F(\mathbf{x}, \mathbf{y})$  is a suitable objective function to be optimized which can conveniently represent application QoS metrics, e.g., response time, system and/or network related metrics, e.g., amount of resources, network traffic, or a combination thereof.  $Q^\alpha(\mathbf{x}, \mathbf{y})$  and  $Q^\beta(\mathbf{x}, \mathbf{y})$  are, respectively, those QoS attributes whose values are settled as a maximum and a minimum, and  $\mathbf{x} \in A$  is a set of functional constraints (e.g., this latter set includes the constraint  $\sum_u x_u^i = 1$ , which requires that a correct placement deploys an operator on one and only one computing node, and  $\sum_\pi y_\pi^{(i,j)} = 1$ , which requires that, in a correct routing, a stream flows on a single path).

The formulation above represents the most general problem formulation whereby we jointly optimize the application deployment  $\mathbf{x}$ , by placing the operator on suitable nodes in the network, while at the same time determining the network paths  $\mathbf{y}$  to carry the stream between operators.

Using standard arguments, see, e.g., [9] for a similar problem, it can be proved that the resulting OPSR problem is NP-hard. As a consequence, efficient heuristics are required to deal with large problem instances in practice. Nevertheless, the proposed formulation can supply useful information for designing heuristics that, not only reduce the resolution time, but guarantee provable approximation bounds on the computed solution.

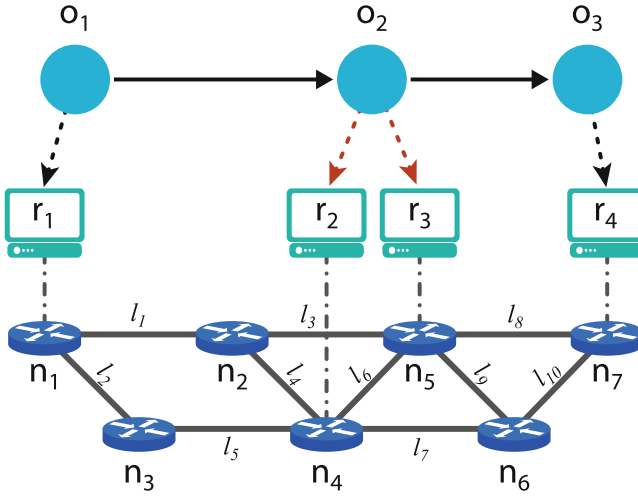


Fig. 4. SDN-supported placement of a DSP application.

### 4.3 Illustrative Example

OPSR determines how the computing and network resources should be utilized so to execute a DSP application with QoS requirements (e.g., response time, cost, availability). We observe that the application performance depends not only on computing resources, but also on network links that realize the communication among the computing nodes. This is especially true in geo-distributed environment (like Fog computing) and when Big Data have to be efficiently transmitted and processed. The strength of OPSR is the ability to jointly optimize (i.e., in a single stage) the selection of computing nodes and of network paths between each pair of selected computing nodes.

We exemplify the problem using Fig. 4. We consider a simple DSP application that filters and forwards important events to a notification service within a limited time interval (i.e., it has QoS requirements on response time). The application comprises a pipeline of three operators: a data source  $o_1$ , a filter  $o_2$ , and a connector to the external notification service  $o_3$ . For the execution, OPSR has to identify computing and network resources from the available infrastructure that, in our example, comprises 4 processing nodes ( $r_i$  with  $i \in \{1, \dots, 4\}$ ), 7 network devices ( $n_i$  with  $i \in \{1, \dots, 7\}$ ), and 10 network links ( $l_i$ , with  $i \in \{1, \dots, 10\}$ —observe that the network is not fully connected). To better show the problem at hand and reduce its complexity, we assume that each computing node  $r_i$  can host at most one operator and that  $o_1$  and  $o_3$  have been already placed on  $r_1$  and  $r_4$ , respectively. Therefore, OPSR has to deploy only the filtering operator  $o_2$  selecting between two possible choices:  $r_2$  and  $r_3$ .

Interestingly, the network control layer can expose different views of the network, so that the upper application layer can select the most suitable network characteristics for running its applications. In our example, we consider that the



network control layer exposes paths with different QoS attributes in terms of communication latency and available bandwidth.

- In case  $o_2$  is deployed on  $r_2$ , OPSR has to further select the network paths for streams  $(o_1, o_2)$  and  $(o_2, o_3)$ , which should flow between  $(r_1, r_2)$  and  $(r_2, r_4)$ , respectively. For the first stream  $(o_1, o_2)$ , the network controller exposes  $\pi_1 = \{l_1, l_4\}$ , with 10 ms latency and 100 Mb/s bandwidth, and  $\pi_2 = \{l_2, l_5\}$ , with 25 ms latency and 1 Gb/s bandwidth. Similarly, for the second stream  $(o_2, o_3)$ , the network controller exposes  $\pi_3 = \{l_6, l_8\}$ , with 10 ms latency and 300 Mb/s bandwidth, and  $\pi_4 = \{l_7, l_{10}\}$ , with 15 ms latency and 850 Mb/s bandwidth.
- In case  $o_2$  is deployed on  $r_3$ , OPSR can determine the network paths for streams  $(o_1, o_2)$  and  $(o_2, o_3)$ , which should flow between  $(r_1, r_3)$  and  $(r_3, r_4)$ , respectively. For the first stream  $(o_1, o_2)$ , the network controller exposes  $\pi_5 = \{l_1, l_3\}$ , with 10 ms latency and 100 Mb/s bandwidth, and  $\pi_6 = \{l_2, l_5, l_6\}$ , with 30 ms latency and 600 Mb/s bandwidth. For the second stream  $(o_2, o_3)$ , the network controller exposes  $\pi_7 = \{l_8\}$ , with 5 ms latency and 100 Mb/s bandwidth, and  $\pi_8 = \{l_9, l_{10}\}$ , with 15 ms latency and 600 Mb/s bandwidth.

The utilization of any of these paths is upon request, because the SDN controller has to allocate resources so to guarantee that QoS performance does not degrade over time (e.g., due to link over-utilization). Since selecting one path or another deeply changes the application performance, OPSR picks the most suitable one driven by the DSP application QoS requirements, which are captured by the objective function  $F(\mathbf{x}, \mathbf{y})$ . Our DSP application needs to forward event notifications with bounds on delay, therefore it prefers to transfer data using the paths with minimum communication latency. Hence, OPSR maps  $o_2$  on  $r_3$  and selects the paths  $\pi_5$  and  $\pi_7$ , which introduce a limited communication latency of 15 ms. Observe that, in case the DSP application aimed to optimize the amount available bandwidth (as in case of media streaming applications), OPSR would have mapped  $o_2$  on  $r_2$  and selected the paths  $\pi_2$  and  $\pi_4$ , which provide a bandwidth of 1 Gb/s and 850 Mb/s, respectively.

Although this is a toy example, it gives a flavor of the potentialities coming from the cooperation between SDN and distributed DSP applications. At the same time, the example shows the combinatorial nature of the OPSR problem, which calls for the development of new efficient heuristics.

#### 4.4 Related Work on Big Data and SDN

With the renewed interest in DSP applications, in the last years many research works have focused on the placement and runtime reconfiguration of DSP applications (e.g., [2, 9, 10, 25, 45] and therein cited works). However, some of these works [2, 45] do only consider the deployment of the DSP application in a clustered and locally distributed environment. Moreover, to the best of our knowledge, none of them exploits the support for the flexible and fine-grained programmable network control offered by SDN.

Enlarging the focus to Big Data applications, of which DSP applications represent the real-time or near-real-time constituent, SDN is considered as a promising paradigm that can help to address issues that are prevailing with such a kind of applications [11, 37]. These issues comprise data processing and resource allocation in locally and geographically distributed data centers, including micro data centers in Fog and edge computing, data delivery to end users, a joint optimization that addresses the tight coupling between data movement and computation, and application scheduling and deployment.

So far, in the Big Data scenario, most works have leveraged SDN to optimize the communication-intensive phase of Hadoop MapReduce [15] by placing MapReduce tasks close to their data, thus reducing the amount of data that must be transferred and therefore the MapReduce job completion time [29, 38, 43, 44]. A first work that explores the tight integration of application and network control utilizing SDN has been presented by Wang et al. [43], which explores the idea of application-aware networking through the design of an SDN controller using a cross-layer approach that configures the network based on MapReduce job dynamics at runtime. The Pythia system proposed by Neves et al. [29] employs communication intent prediction for Hadoop and uses this predictive knowledge to optimize at runtime the network resource allocation. The Pythia network scheduling component computes an optimized allocation of flows to network paths and, similarly to the QoS routing in our SIDF architecture, maps the logical flow allocation to the physical topology and installs the proper sequence of forwarding rules on the network switches. Xiong et al. propose Cormorant [44], which is a Hadoop-based query processing system built on top of SDN, where MapReduce optimizes task schedules based on the network state provided by SDN and SDN guarantees the exact schedule to be executed. Specifically, SDN is exploited to provide the current snapshot of the network status and to install the network path having the best available bandwidth. Their experimental results show a 14–38% improvement in query execution time over a traditional approach that optimizes task and flow scheduling without SDN collaboration. Qin et al. in [38] propose a heuristic bandwidth-aware task scheduler that combines Hadoop with the bandwidth control capability offered by SDN with the goal to minimize the completion time of MapReduce jobs.

The integration of SDN into the control loop of self-adaptive applications has been studied by Beigi-Mohammadi et al. [3] with the goal of exploiting network programmability to meet application requirements. This is a new trend in the design of self-adaptive systems. We also explore it with the SIDF architecture: the integration of SDN allows us to adapt at runtime the stream routing so that the QoS requirements of the DSP application can still be guaranteed when network operating conditions change. Besides the SDN appealing features, the strict cooperation between adaptive systems and the SDN controller might easily become a scalability bottleneck. Indeed, SDN controller are often implemented as a single centralized entity, whereas adaptive systems can span over geographically distributed infrastructures. Further research investigations are needed to enable the exploitation of SDN features in a scalable manner.

## 5 Context-Aware Composition of Big Services

Big services are typically composed of smaller web services or microservices, each with multiple alternative deployments to ensure performance, scalability, and fault-tolerance. Such service compositions enable the design and implementation of complex business processes, eScience workflows, and Big data applications, by aggregating the services. Services are often implemented using several approaches, languages, and frameworks still offering the same API, standardized as RESTful or Service Oriented Architecture (SOA) [30] web services.

As the demand for QoS and data quality is on the rise, along with the ever-increasing scale of Big data, service compositions execute in computational nodes that are geographically distributed in the Internet-scale. SDN can be extended and leveraged to manage the underlying network that interconnects the building blocks of such complex workflows, to enhance the scalability and potential use cases in services computing. An integration of SDN and NFV into service composition facilitates efficient context-aware distribution of service execution closer to the data, minimizing latency and communication overhead.

### 5.1 Software-Defined Service Composition (SDSC)

SDSC is an approach to a distributed and decentralized service composition, which leverages SDN for an efficient service placement on the service nodes. Following the SDSC approach, a typical eScience workflow is mapped onto a geographically distributed service composition. SDSC exploits both the data-as-a-service layer and network layer for the resource allocation. System administrators can monitor the health of the service compositions, through the web service engines that host the services, by observing the runtime parameters such as the executed requests and the requests on the fly can be monitored. The list of multiple web service deployments can be retrieved from the web service registry. In addition to these, SDSC leverages the global network knowledge of the SDN controller to find the network parameters such as bandwidth utilization to fine tune the services placement, offering features such as congestion control and load balancing, which can better be achieved in the network layer.

By separating the execution from the data plane of the overall system, SDSC facilitates integration and interoperability of more diverse implementations and adaptations of the services. A resilient execution of service composition can be guaranteed through the network management capabilities offered by SDN, in finding the best alternative among various service implementations and deployments among the multiple potential services deployments for the service composition execution. SDSC thus facilitates an increased control over the underlying network, while supporting the execution from various traditional web services engines and the distributed execution frameworks.

The core of SDSC is constituted by the communication between inter-domain SDN controllers, facilitated by various Message-Oriented Middleware (MOM) [12] protocols such as AMQP and MQTT. The service requests are mapped to the network through SDN, and the resource provisioning is managed

with the assistance of the SDN controller. Hence, each domain is aware of the services that are served by the services hosted in them. By offering communication between inter-domain controllers, resources are allocated efficiently for each service request.

There is an increased demand for configurability to service composition. Context-aware service composition is enabled by exploiting SDN in deploying service compositions. The Next Generation Service Overlay Network (NGSON) specification offers context-aware service compositions by leveraging virtualization [20]. SDN and NFV support context-awareness and traffic engineering capabilities [34], to manage and compose services. Research efforts focus on efficient resource utilization as well as enabling pervasive services [23] motivated by the standardization effort of NGSON.

## 5.2 Componentizing Data-Centric Big Services on the Internet

Workflows of mission-critical applications consist of redundancy in links and alternative implementations and deployments in place, either due to parallel independent developments or developed such to handle failures, congestion, and overload in the nodes. Distributed cloud computing and volunteer computing are two examples that permit multi-tenant computation-intensive complex workflows to be executed in parallel, leveraging distributed resources.

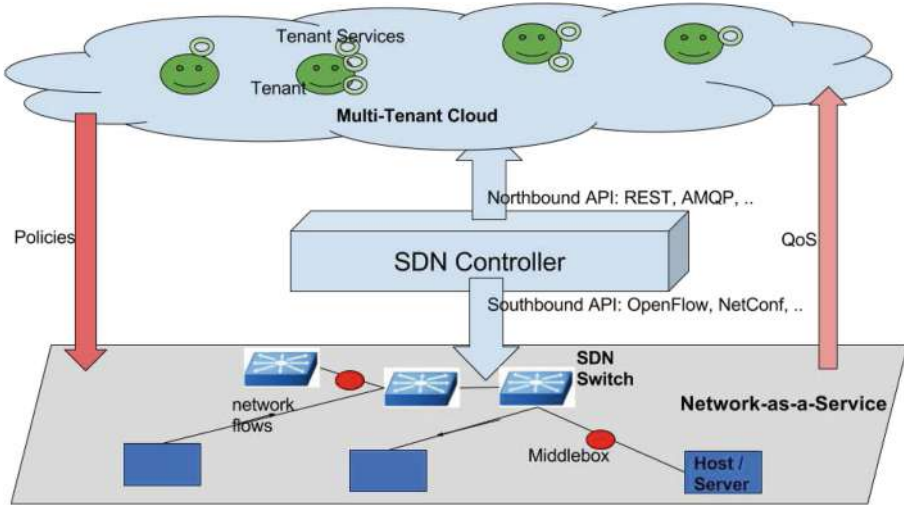
Figure 5 represents a multi-tenant cloud environment with various tenants. The tenants execute several big services. Many aspects such as locality of the executing cloud data center and policies must be considered for an efficient execution of the service workflow. An SDN controller deployment can ensure QoS to the cloud, by facilitating an efficient management of the network-as-a-service consisting of SDN switches, middleboxes, and hosts or servers. The controller communicates with the cloud applications through its northbound API, while controlling the SDN switches through its southbound API. Thus, SDN facilitates an efficient execution of big services.

In practice, no complex big service is built and deployed as a singleton or a tightly coupled single cohesive unit. Mayan [21], which is a distributed execution model and framework for SDSC, defines the services that compose a big services workflow as the “building blocks” of the workflow. SDSC aims to extend the SDN-enabled service execution further to the Internet-scale.

**Representation of the Model.** We need to consider and analyze the potential execution alternatives of the services, to support a context-aware execution of service compositions. In this section, we formally model the big services as service compositions and consider the potential execution alternatives for their context-aware execution. Services are implemented by various developers following different programming languages and paradigms.

$\forall n \in \mathbb{Z}^+; \forall \alpha \in \{A, B, \dots, N\}: s_\alpha^n$  represents the  $\alpha^{th}$  implementation of service  $s^n$ .

Each implementation of a service can have multiple deployments, distributed throughout the globe, either as replicated deployments or independent



**Fig. 5.** Network- and service-level views of a multi-tenant cloud.

deployments by different edge data centers. These multiple deployments facilitate a bandwidth-efficient execution of the services.

$\forall m \in \mathbb{Z}^+ : s_{\alpha m}^n$  represents the  $m^{th}$  deployment of  $s_{\alpha}^n$ .

Each service can be considered a function of a varying number of input parameters. Any given big service  $S$  can be represented as a composite function or a service composition. These service compositions are composed of a subset of globally available services.

$\forall x \in \mathbb{Z}^+, x \leq n; S = s^1 \circ s^2 \circ \dots \circ s^x$ .

The minimum number of execution alternatives for any service can be represented by  $\kappa_x$ , where:

$$\forall s \in S : \kappa_x = \sum_{\alpha=A}^N m_{\alpha}.$$

Here,  $N$  different service implementations and a varying number  $m_{\alpha}$  of deployments for each implementation of  $s$  are considered.

**Minimum and Maximum Execution Alternatives.** Now we will formalize the maximum and minimum execution alternatives for any service composition, considering the multiple implementations or deployed replicas of the same service. More execution alternatives will offer more resilience and scalability to the service composition.

$\eta_S$  represents the number of alternative execution paths for each big service  $S$ . The service that has the minimum alternatives limits the minimum number of potential alternatives for a service composition.

$$\eta_S \geq \min(\kappa_x : x \leq n) \geq 1.$$

Taking into account the alternatives due to various service combinations in the big service, the maximum alternatives is limited by a product of alternatives for each service.

$$\eta_S \leq \prod_{x=1}^n \kappa_x.$$

Hence,

$$\min(\kappa_x: x \leq n) \leq \eta_S \leq \prod_{x=1}^n \kappa_x.$$

Various protocols and web services standards unify the message passing between the services, and enable seamless migration among the alternatives, in a best-effort and best-fit strategy. SOA and RESTful web services support common message formats through standardizations. These efforts unify and revolutionize the way services are built on the Internet.

### 5.3 Illustrative Example

Figure 6 illustrates a sample workflow that represents a service composition. This workflow can be an eScience workflow or a complex business process. The workflow represents multiple possible execution paths when the service composition is decomposed or componentized into services (Services 1, 2, ..., n). A, B, C, ..., Z represents the alternative implementations for each of the services. Thus, service implementations such as 1A, 1B, and 1Z can function as an alternative to each other (here, each of these is an implementation of service 1).

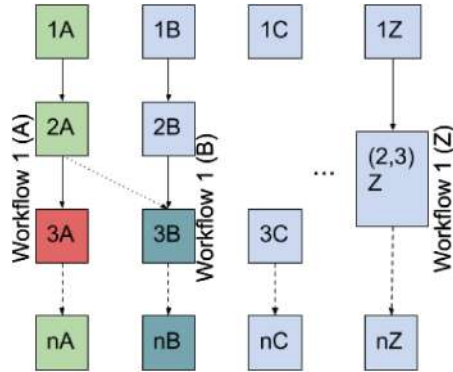
As illustrated by Fig. 6, if service 3A is either congested or crashed, the service execution can be migrated to the next best-fit (chosen based on locality or some other policy) deployment 3B. (2,3)Z represents a service that is equal to the service composition of 3A(2A), the output of 2A as an input to 3A. Hence, it is not an alternative to 2A or 3A. It is also possible that not all the services have alternative deployments in considered environments (as indicated by the lack of Service 2 as in 2C). Service deployment details need to be specified in the service registry to be able to compose and execute the service workflows seamlessly.

### 5.4 eScience Workflows as Service Compositions

The Internet consists of various data-centric big services. Complex eScience workflows leverage multiple big services for their execution and can be decomposed into various geo-distributed web services and microservices. eScience workflows can, therefore, be represented by service compositions. Thus, these big services, centered around big data, can be expressed into simpler web services, which can be executed in a distributed manner.

Mayan seeks to find the best fit among the alternatives of available service execution options, considering various constraints of network and service level resource availability and requirements, while respecting the locality of the service requests. Mayan proposes a scalable and resilient execution approach to offer a multi-tenant distributed cloud computing platform to execute these services beyond data center scale.

Mayan enables an adaptive execution of scientific workflows through federated SDN controllers deployed in a wide area network. Hence, Mayan leverages



**Fig. 6.** Simple representation of multiple alternative workflow executions.

the various potential alternative execution paths existing between the service compositions, while exploiting the network knowledge of the SDN controller. Furthermore, Mayan utilizes the local workload information available at the web service engine and web services registry. The information received from this services layer includes web service requests on the fly and web services served at any time by the web service deployment. As an implementation of an SDSC, Mayan exploits both the control plane and services plane in offering a load-balanced, scalable, and resilient execution of service compositions. Mayan leverages OpenDaylight’s data tree as an efficient control plane data store while using an AMQP-based messaging framework to communicate across multiple network domains in service resource allocation.

### 5.5 Inter-domain SDN Deployments

The SDN architecture needs to be extended for an Internet-wide service composition. A global view of the entire network hierarchy may not even be feasible to achieve for a single central controller due to the organizational policies. An inter-domain SDN deployment is necessary to cater for this scale and segregation of the network. Here, each domain (that can represent a cloud, organization, or a data center) is orchestrated by an SDN controller cluster.

The clustered deployment prevents the controller from becoming a single point of failure or a bottleneck. As eScience workflows are deployed on a global scale, a federated deployment of controller clusters is leveraged to enable communications between inter-domain controller clusters, without sharing a global network view. The federated deployment allows network level heuristics to be considered beyond data center scale, using MOM protocols in conjunction with SDN. Inter-domain controllers communicate through MOM messages between one another. Hence, SDN controllers of different domains have protected access to data orchestrated by one another, based on a subscription-based configuration rather than a static topology.



Some research work has previously leveraged federated SDN controller deployments for various use cases. CHIEF [22] presents a scalable inter-domain federated SDN controller deployment for wide area networks, as a “controller farm”. It builds a large-scale community cloud orchestrated by various independent controller clusters sharing data through a protected MOM API. Such controller farm may support collaboration between multiple organization networks, otherwise limited from network-level coordination. SDSC can be extended to create a Service Function Chaining (SFC), that is an ordered sequence of middlebox actions or VNFs such as load balancing and firewall.

## 6 Benefits and Open Issues

Network virtualization and programmability of network resources enable dynamic creation of service chains that satisfy QoS demands of complex services at runtime. Runtime control of traffic and usage of network resources is provided from infrastructure to control layer thus enabling runtime management decisions. Abstracting the network infrastructure plane is a movement similar as introducing higher levels of abstraction into programming languages. The key benefit of such abstraction is enabling less experienced developers to easier program new applications, using abstract objects of network resources, with the help of formal programming frameworks and environments. The risks of programmer faults are minimized through formalisms implemented in programming languages. The main benefit is in offloading new application developers of very complex network skills, thus opening application development even to not skilled people and innovation opportunities to the wider community. Abstraction of network resources will benefit with opening innovation opportunities based on the use of unlimited network resources.

A direct consequence of opening network resources to wider developers community is in accelerating the process of offering new features to end users and minimizing development costs. Another result of abstraction is the introduction of standard interfaces that enable evolution and change of each layer independently. Contrary to traditional networks where there is a dominant vendor lock-in solutions, in new network architecture, with introduced standard application platform interfaces between network layers, the independence to provider equipment has opened numerous opportunities for innovation by using an unlimited pool of network resources and services offered by various networks.

Furthermore, the programmable network enables numerous possibilities for network automation. New service management models may be developed at each network layer independently with runtime control of network resources. These may be used to autonomous control efficiency of network resource use while addressing specific QoS requirements of the particular application.

Nowadays, service compositions and Big Data applications must deal with changing environments and variable loads. Therefore, to guarantee acceptable performance, these applications require frequent reconfigurations, such as adjustments of application component placement or selection of new services. In this



respect, SDN capability of programming, the network at runtime allows a cross-layer management of computational and networking resources, thus enabling a joint optimization of application placement (or service composition) and data stream routing. The cross-layer management can be beneficial especially in geodistributed environments, where network resources are heterogeneous, subject to changing working conditions (e.g., congestion), and characterized by non-negligible communication delays. In an SDN environment, the application control layer (e.g., service composition broker, DSP framework) can regard the network as a logical resource, which can be managed as a computing resource in a virtualized computing environment. Specifically, the programmability allows to automate and control the network so to adjust its behavior as to fulfill the application needs. For example, multiple paths or paths with specific QoS attributes can be reserved for transmitting data, data streams can be redirected during application components downtime, or network devices can be programmed to carry out new functions. Moreover, the use of standardized interfaces between the application layer and network controller (i.e., Northbound APIs) allows simplifying the implementation and utilization of new network services (e.g., QoS-based routing).

With respect to the integration of SDN and Big Data and specifically to the SIFD architecture presented in Sect. 4, we observe that when the network controller in SDN is used for Big Data applications, its performance could be degraded due to the rapid and frequent flow table update requests which might not be sustained by today SDN controllers. The problem is exacerbated if the controller serves multiple applications/frameworks as it can easily become the performance bottleneck of the entire architecture. To this end, we need to define solutions which cater for the presence of multiple applications, with possible diverse and conflicting QoS requirements by defining policies which ensure fair usage of network resources in the face of competing resources requests. The problem becomes relevant in large-scale distributed environments, where a centralized approach might not scale, and distributed solution becomes preferable.

New service development formalisms may be required to standardize processes at the network management level. In the future use of such a programmable network environment, a network is seen as an unlimited pool of resources. So, it is expected a significantly increase in the network use with a number of new and innovative services. Such increase in diversity of network services and a number of new application interfaces would need to redefine service development and management models. New design principles would be needed, and this need would be recognized with increased diversity at network application layer. For such purposes, there is a need for new developments in formal methods for introducing the controlled behavior in programming network. Development of network compilers is ongoing research activity for these purposes. Furthermore, new mathematical models are needed that would be able to describe network behavior. There is a need for some generative models that can predict the parameters from the internal properties of the processes we are controlling. Such models would not only bring efficiency in processing network control algorithms, but would also be stimulating phenomena in network behavior.

## 7 Conclusions

In this chapter, we looked into how SDN and NFV enable QoS-aware service compositions, and how SDN can be leveraged to facilitate cross-layer optimizations between the various network and service layers. So far, SDN has been largely and separately exploited mainly in telecommunication environments. For example, NFV placement and SDN routing for network embedding have been used to achieve energy efficiency as explained in Sect. 3. However, there is an increasing interest in exploring the network control opportunities offered by SDN in the Big Data context, as discussed for the deployment of DSP applications on the underlying computing and networking resources. In the use case presented in Sect. 4, SDN is used to expose to the service management layer the network topology and network-related QoS metrics. The service management layer determines both the application components placement on the underlying computing resources and the network paths between them. In this way, SDN allows autonomous adjustment of the network paths as per-application needs. Furthermore, in Sect. 5 we provided an example of using SDN for the design and implementation of complex scientific and business processes.

Through these three examples, we presented different deployment management decisions for service compositions over the layers of a network architecture that integrates SDN and NFV. As future research direction, we identify the need for the development of an autonomous management framework that can coordinate cross-layer decisions taken by different management layers while deploying service compositions that satisfy QoS guarantees in an Internet-scale distributed network. Future work is also needed to investigate the side effects that may arise from the coordination among management decisions at different layers.

## References

1. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., et al.: Aurora: a new model and architecture for data stream management. *VLDB J.* **12**(2), 120–139 (2003)
2. Aniello, L., Baldoni, R., Querzoni, L.: Adaptive online scheduling in Storm. In: *Proceedings of 7th ACM International Conference on Distributed Event-Based Systems, DEBS 2013*, pp. 207–218 (2013)
3. Beigi-Mohammadi, N., Khazaei, H., Shtern, M., Barna, C., Litoiu, M.: On efficiency and scalability of software-defined infrastructure for adaptive applications. In: *Proceedings of 2016 IEEE International Conference on Autonomic Computing, ICAC 2016*, pp. 25–34 (2016)
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: *Proceedings of 1st Workshop on Mobile Cloud Computing, MCC 2012*, pp. 13–16 (2012)
5. Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A.: Energy-efficient data replication in cloud computing datacenters. *Cluster Comput.* **18**(1), 385–402 (2015)
6. Bouguettaya, A., Singh, M., Huhns, M., Sheng, Q.Z., et al.: A service computing manifesto: the next 10 years. *Commun. ACM* **60**(4), 64–72 (2017)
7. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: On QoS-aware scheduling of data stream applications over fog computing infrastructures. In: *Proceedings of IEEE ISCC 2015*, pp. 271–276, July 2015

8. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Distributed QoS-aware scheduling in Storm. In: Proceedings of 9th ACM International Conference on Distributed Event-Based Systems, DEBS 2015, pp. 344–347 (2015)
9. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Optimal operator placement for distributed stream processing applications. In: Proceedings of 10th ACM International Conference on Distributed and Event-Based Systems, DEBS 2016, pp. 69–80 (2016)
10. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Optimal operator replication and placement for distributed stream processing systems. *ACM SIGMETRICS Perform. Eval. Rev.* **44**(4), 11–22 (2017)
11. Cui, L., Yu, F.R., Yan, Q.: When big data meets software-defined networking: SDN for big data and big data for SDN. *IEEE Netw.* **30**(1), 58–65 (2016)
12. Curry, E.: Message-oriented middleware. In: *Middleware for Communications*, pp. 1–28. Wiley, Hoboken (2005)
13. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.* **34**(4), 15–26 (2004)
14. Davy, S., Famaey, J., Serrat, J., Gorricho, J.L., Miron, A., Dramitinos, M., Neves, P.M., Latre, S., Goshen, E.: Challenges to support edge-as-a-service. *IEEE Commun.* **52**(1), 132–139 (2014)
15. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113 (2008)
16. Enns, R., Bjorklund, M., Bierman, A., Schönwälder, J.: Network Configuration Protocol (NETCONF). RFC 6241, June 2011
17. Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: challenges and opportunities for innovations. *IEEE Commun.* **53**(2), 90–97 (2015)
18. Heinze, T., Aniello, L., Querzoni, L., Jerzak, Z.: Cloud-based data stream processing. In: Proceedings of 8th ACM International Conference on Distributed Event-Based Systems, DEBS 2014, pp. 238–245 (2014)
19. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: ElasticTree: saving energy in data center networks. In: Proceedings of 7th USENIX Conference on Networked Systems Design and Implementation, NSDI 2010 (2010)
20. John, W., Pentikousis, K., Agapiou, G., Jacob, E., Kind, M., Manzalini, A., Risso, F., Staessens, D., Steinert, R., Meirosu, C.: Research directions in network service chaining. In: 2013 IEEE SDN for Future Networks and Services. SDN4FNS (2013)
21. Kathiravelu, P., Galinac Grbac, T., Veiga, L.: Building blocks of Mayan: Componentizing the escience workflows through software-defined service composition. In: Proceedings of 2016 IEEE International Conference on Web Services, ICWS 2016, pp. 372–379 (2016)
22. Kathiravelu, P., Veiga, L.: CHIEF: controller farm for clouds of software-defined community networks. In: Proceedings of 2016 IEEE International Conference on Cloud Engineering Workshop, IC2EW 2016 (2016)
23. Liao, J., Wang, J., Wu, B., Wu, W.: Toward a multiplane framework of NGSON: a required guideline to achieve pervasive services and efficient resource utilization. *IEEE Commun.* **50**(1) (2012)
24. Lim, S.H., Sharma, B., Nam, G., Kim, E.K., Das, C.R.: MDCSim: a multi-tier data center simulation platform. In: Proceedings of 2009 IEEE International Conference on Cluster Computing and Workshops, August 2009
25. Lohrmann, B., Janacik, P., Kao, O.: Elastic stream processing with latency guarantees. In: Proceedings of IEEE ICDCS 2015, pp. 399–410 (2015)

26. Marotta, A., D'Andreagiovanni, F., Kassler, A., Zola, E.: On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures. *Comput. Netw.* **125**, 64–75 (2017)
27. Matsubara, D., Egawa, T., Nishinaga, N., Kaffle, V.P., Shin, M.K., Galis, A.: Toward future networks: a viewpoint from ITU-T. *IEEE Commun.* **51**(3), 112–118 (2013)
28. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **38**(2), 69–74 (2008)
29. Neves, M.V., De Rose, C.A.F., Katrinis, K., Franke, H.: Pythia: faster big data in motion through predictive software-defined network optimization at runtime. In: *Proceedings of IEEE 28th International Parallel and Distributed Processing Symposium, IPDPS 2014*, pp. 82–90 (2014)
30. Newcomer, E., Lomow, G.: *Understanding SOA with Web Services*. Addison-Wesley, Upper Saddle River (2005)
31. Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T.: A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun. Surv. Tutorials* **16**(3), 1617–1634 (2014)
32. OASIS: MQTT version 3.1.1 (2014)
33. Osseiran, A., Monserrat, J.F., Marsch, P.: *5G Mobile and Wireless Communications Technology*, 1st edn. Cambridge University Press, New York (2016)
34. Paganelli, F., Ulema, M., Martini, B.: Context-aware service composition and delivery in NGSONs over SDN. *IEEE Commun.* **52**(8), 97–105 (2014)
35. Panda, P.R., Silpa, B.V.N., Shrivastava, A., Gummidipudi, K.: *Power-Efficient System Design*, 1st edn. Springer, Boston (2010). <https://doi.org/10.1007/978-1-4419-6388-8>
36. Pedram, M., Hwang, I.: Power and performance modeling in a virtualized server system. In: *Proceedings of 39th International Conference on Parallel Processing Workshops, ICPPW 2010*, pp. 520–526 (2010)
37. Qadir, J., Ahad, N., Mushtaq, E., Bilal, M.: SDNs, clouds, and big data: new opportunities. In: *Proceedings of 12th International Conference on Frontiers of Information Technology*, pp. 28–33 (2014)
38. Qin, P., Dai, B., Huang, B., Xu, G.: Bandwidth-aware scheduling with SDN in Hadoop: a new trend for big data. *IEEE Syst. J.* **11**(4), 2337–2344 (2015)
39. Richardson, L., Ruby, S.: *RESTful Web Services*. O'Reilly Media, Inc., Sebastopol (2008)
40. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., et al.: Storm@Twitter. In: *Proceedings of ACM SIGMOD 2014*, pp. 147–156 (2014)
41. Van Adrichem, N.L., Doerr, C., Kuipers, F.A.: OpenNetMon: network monitoring in OpenFlow software-defined networks. In: *Proceedings of 2014 IEEE Network Operations and Management Symposium, NOMS 2014* (2014)
42. Vinoski, S.: Advanced message queuing protocol. *IEEE Internet Comput.* **10**(6) (2006)
43. Wang, G., Ng, T.E., Shaikh, A.: Programming your network at run-time for big data applications. In: *Proceedings of 1st Workshop on Hot Topics in Software Defined Networks, HotSDN 2012*, pp. 103–108. ACM (2012)
44. Xiong, P., He, X., Hacigumus, H., Shenoy, P.: Cormorant: running analytic queries on MapReduce with collaborative software-defined networking. In: *Proceedings of 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2015*, pp. 54–59 (2015)
45. Xu, J., Chen, Z., Tang, J., Su, S.: T-Storm: traffic-aware online scheduling in Storm. In: *Proceedings of IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014*, pp. 535–544 (2014)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Energy vs. QoX Network- and Cloud Services Management

Bego Blanco<sup>1</sup>✉, Fidel Liberal<sup>1</sup>, Pasi Lassila<sup>2</sup>, Samuli Aalto<sup>2</sup>,  
Javier Sainz<sup>3</sup>, Marco Griboaud<sup>4</sup>, and Barbara Pernici<sup>4</sup>

<sup>1</sup> University of the Basque Country, Leioa, Spain  
{`begona.blanco,fidel.liberal`}@ehu.eus

<sup>2</sup> Aalto University, Espoo, Finland  
{`Pasi.Lassila,samuli.aalto`}@aalto.fi

<sup>3</sup> Innovati Group, Madrid, Spain  
jsg@grupoinnovati.com

<sup>4</sup> Politecnico di Milano - DEIB, Milan, Italy  
{`marco.griboaud,barbara.pernici`}@polimi.it

**Abstract.** Network Performance (NP)- and more recently Quality of Service/Experience/anything (QoS/QoE/QoX)-based network management techniques focus on the maximization of associated Key Performance Indicators (KPIs). Such mechanisms are usually constrained by certain thresholds of other system design parameters. e.g., typically, cost. When applied to the current competitive heterogeneous Cloud Services scenario, this approach may have become obsolete due to its static nature. In fact, energy awareness and the capability of modern technologies to deliver multimedia content at different possible combinations of quality (and prize) demand a complex optimization framework.

It is therefore necessary to define more flexible paradigms that make it possible to consider cost, energy and even other currently unforeseen design parameters not as simple constraints, but as tunable variables that play a role in the adaptation mechanisms.

In this chapter we will briefly introduce most commonly used frameworks for multi-criteria optimization and evaluate them in different Energy vs. QoX sample scenarios. Finally, the current status of related network management tools will be described, so as to identify possible application areas.

## 1 Introduction

Network Performance- and more recently Quality of Service/Experience/X-based network management techniques (where “X” can represent “S” service, “P” perception, “E” experience or “F” flow, just to give a few examples), focus on the maximization of associated KPIs. Such mechanisms are usually constrained by certain thresholds of other system design parameters, e.g., typically, cost. When applied to the current competitive heterogeneous Internet of Services scenario, this approach may have become obsolete due to its static nature. In fact,

energy awareness and the capability of modern technologies to deliver multimedia content at different possible combinations of quality (and prize) demand a complex optimization framework.

It is therefore necessary to define a more flexible paradigm that makes it possible to consider cost, energy and even other currently unforeseen design parameters not as simple constraints, but as tunable variables that play a role in the adaptation mechanisms. As a result, for example, the service supply will then search for the maximum QoE at the minimum cost and/or energy consumption. In consequence, a certain service will not be offered at a single and specific guaranteed price, but will vary with the objective of obtaining the best (QoE, cost, energy, etc.) combination at a given time.

Unfortunately, most considered design parameters are conflicting, and therefore the improvement of one of them entails some deterioration of the others. In these circumstances, it is necessary to find a trade-off solution that optimizes the antagonistic criteria in the most efficient way. Therefore, the resource allocation problem becomes a multi-criteria optimization problem and the relevance of each criteria gains uttermost importance.

This chapter analyzes the existing optimization frameworks and tools and studies the complexity of introducing utility functions into network/management mechanisms, including fairness considerations. Then, we present cost/energy/\*-aware network and cloud services management scenarios. Finally, we address the challenge of introducing energy-awareness in network controlling mechanism and provide a general view of current technologies and solutions.

## 2 Dealing with Multi-criteria Optimization: Frameworks and Optimization Tools

Regardless the mathematical or heuristic tools applied in order to find (near) optimal solutions in the scope of Internet of Services management mechanisms, all of them share common issues due to the extension of the original definition of the problem to a multi-criteria one. This section provides a summarized compilation of those issues, especially those related to how the decision maker (DM) will take into consideration different antagonistic criteria.

### 2.1 Generic Definition of the Problem

The classical constrained single criteria problem deals with finding the combination of design parameters (normally represented by a vector  $x^*$ ) in the feasible space ( $S$ ) that minimize a single function (1).

$$\exists x^* \in S / \min f(x^*) = z \quad (1)$$

Then the multi-criteria or multi-objective optimization problem, defined as an extension of the mono-criteria one, aims at simultaneously minimizing a collection of requirements keeping the equality and inequality constraints of the feasible space (2).

$$\exists x^* \in S / \min f_i(x^*) = z_i \forall i = 1, 2, \dots, k \quad (2)$$

The optimal solution that minimizes simultaneously all the criteria is most of the times hardly achievable, and is known as utopian solution [5]. Therefore, the actual best solution of the problem should be as close as possible to this utopian solution. The optimization problem must then be redefined to extract from the whole feasible space of solutions, those closer to the utopian solution. That set of solutions characterizes the Pareto-optimal front. The goal of a good multi-criteria optimization problem is the search of a set of solutions that properly represents that Pareto front, i.e., uniformly distributed along that Pareto front.

However, due to the trade-offs among different parameters, in most of the cases there will not exist such a solution which minimizes all the criteria simultaneously. So, the nature of the problem is usually re-defined by introducing the concept of Utility Function, responsible for quantifying the relevance and composite articulation of different criteria. Then, the real formulation of the problem can be expressed mathematically as follows (3).

$$\exists x^* \in S / \min U(z_1, z_2, \dots, z_k) \quad (3)$$

## 2.2 Incorporating Multiple Criteria in General Optimization Methods

Multiple Objective Optimization (MOO) has been a field of intensive research in different engineering areas. This activity has led to the development of a lot of MOO methods ranging from exact methods to meta-heuristics and including several different nature algorithms.

In this section, we propose a comprehensive taxonomy of the optimization problem synthesized from the works in [13, 14, 21, 30, 31, 36]. The presented taxonomy categorizes the optimization problems according to different perspectives where the main goal is to determine how the multiple criteria are considered by the DM. Table 1 summarizes the characterization of the optimization criteria that are defined as follows:

- **Qualitative vs. quantitative criteria:** refers to how the analyzed criteria are measured. If the DM is able to represent the preference degree of one option against the others by a numerical value, then the criteria are quantitative. Otherwise, the criteria are qualitative, meaning that preference can not be numerically measured or compared and, in consequence, a descriptive value is assigned.
- **Preference articulation:** refers to the point in time the DM establishes its preferences:
  - **A priori preference articulation:** the preferences are defined at problem modeling stage, adding supplementary constraints to the problem (i.e., weighted sum and lexicographic methods).
  - **A posteriori preference articulation:** once the optimization problem provides the set of results from the optimization process, DM's preferences are used to refine the final solution (i.e., in evolutive and genetic methods).



- **Progressive preference articulation:** DM's preferences are gradually incorporated in an interactive way during the optimization process.
  - **Without preference articulation:** when there is no preference definition for the problem (i.e., max-min formulation, global criterion method).
- **Continuous vs. discrete:** refers to the variable type used to describe the optimization criteria. When the optimization problem handle discrete variables, such as integers, binary values or other abstract objects, the objective of the problem is to select the optimum solution from a finite, but usually huge, set. On the contrary, continuous optimization problems handle infinite variable values. In consequence, continuous problems are usually easier to solve due to their predictability, because the solution can be achieved with an approximate iterative process. Since cost/energy aware network and services management must deal with both discrete (i.e., number of servers, route lengths, radio bearers, etc.) and continuous design parameters (i.e., coding bit rate, transmission power, etc.) both techniques should be considered.
  - **Constrained vs. not constrained:** refers to the possibility of attaching a set of requirements expressed through (in)equality equations to the optimization problem. In this case, besides finding a solution that optimizes a collection of criteria, it must also meet a set of constraints. Non constrained methods can be used to solve constrained methods, replacing restrictions for penalizations on objective functions to prevent possible constraint violations. As aforementioned, classical network management approached involved considering a single criteria only and establishing Cost and Energy constraints. The proposal in ACROSS to move to a multi-criteria optimization analysis does not necessarily imply getting rid of all the possible constraints.

Those classifications do not result into disjoint categories. In fact, multi-criteria optimization problems in the considered heterogeneous network and services management scenario may fall into one or several of the categories listed above.

Summarizing, before beginning with the process of multi-criteria optimization problem there is a crucial previous step: the definition of the criteria to be optimized, i.e., the preferences of the DM about the suitability of the obtained solution.

Regardless the decision maker being the Cloud/Network/SOA designer or service operator the adaptation algorithm must incorporate the impact of different criteria on their perception of the goodness of any solution. A key factor in the analysis for decision making is indeed the fact that the functions that model decision maker's preferences (criteria or objective functions) are not usually known a priori.

### 2.3 Complexity of Defining Multi-criteria Utility Functions to be Incorporated in Network/Management Mechanisms

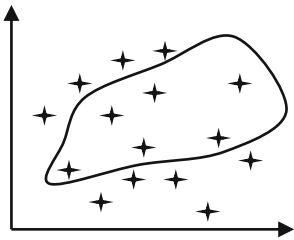
Considering the relevance of the choice of a multi-criteria utility function, different tools aiding at this task will be reviewed in this section.

**Table 1.** Characterization of the optimization criteria for DM.

Description type	Qualitative
	Quantitative
Preference articulation	A priori
	A posteriori
	Progressive
	None
Type of variables	Continuous
	Discrete
Constraint definition	Constrained
	Not constrained

- Goal attainment
- MAUT (Multi-Attribute Utility Theory)
- Preference relations
- Fuzzy logic
- Valuation scale

**Goal Attainment.** This basic format restricts the feasible space with the most relevant set of alternatives according to the DM’s preferences (Fig. 1). Such preferences if represented mathematically usually result in a n-dimensional shape or contour in the decision space limiting those solutions acceptable by the DM (similar to that imposed by the constraints in the design space). It is a simple and direct format, that just splits alternatives into relevant/non-relevant groups. However, it only offers little information about preferences, not providing any hint about the predilections of the DM.



**Fig. 1.** Selection set within feasible space.

The work in [8,9] describe the use of goal attainment preference modeling in multi-criteria algorithms.

**Multi-Attribute Utility Function.** In this case, the utility function is build by describing the repercussion of an action regarding a specific criterion. Each action is assigned a numerical value, so that the higher the value, the more preferable the action. Then, the assessment of an action becomes the weighted sum of the numerical values related to each considered criterion. This representation format is capable of modeling DM's preference more precisely than the Selection Set. Nonetheless, it also means that the DM must evaluate its inclinations globally, comparing each criterion against all the others, which is not always possible. Therefore, this type of utility function is suitable just for the cases in which a perfect global rationality can be assumed [46]. For example, this classic model is commonly employed in economics and welfare field.

Besides, utility has an ordinal nature, in the sense that the preference relation between the possible choices is more significant than the specific numerical values [4]. So, this leaves the door open to discarding the numerical value of the utility, as it is shown next.

**Preference Relations.** This representation format models the inclination over a set of possible choices using a binary relation that describes the qualitative preference among alternatives. Then, a numerical value is linked to that relation, defining the preference degree of alternative  $x_i$  against alternative  $x_k$  in a quantitative way [37].

This format of preference modeling provides an alternative to the assignment of a numerical value to different utility levels, allowing the comparison of alternatives pairwise, providing the DM higher expressiveness to enunciate his preferences (i.e., similar to Analytic Hierarchy Process – AHP [39]). Outranking methods employ this format of preference representation.

**Fuzzy Logic.** This format allows the introduction of uncertainty over the preferences under analysis. In order to avoid ambiguity in the definition process of the preferences, each “ $x_i$  is not worse than  $x_k$ ” is attached a credibility index. In this sense, fuzzy logic becomes a useful tool [46], as a general framework for preference modeling where certain sentences are a particular case.

The obstacle of using fuzzy logic with credibility indexes is the weakening of the concept of truth. The infinite possible values of truth between absolute truth and falseness have an intuitive meaning that does not correspond to their formal semantics. In addition, there are other problems, such as the formulation of the credibility index itself.

**Valuation Scale.** This preference formulation defines a formal representation of the comparison between possible choices that expresses both the structure of the described situation and the variety of manipulations that can be made on it [37]. This type of sentences are appropriately expressed in logical language. But classical logic can be too inflexible to acceptably define expressive models. In consequence, other formalisms must be taken into account to provide the model with the required flexibility.

**Conclusion.** Preference or criteria description format plays a crucial role in the definition of the nature and structure of the information the DM employs to set his predilections up towards the different possibilities. The selection of the best representation format will rely on the characteristics of the specific area of expertise. Sometimes, inclinations will be better expressed using numerical values, and in other cases using more natural descriptions, such as words or linguistic terms.

The final goal is to contrast the impact of the potential actions with the purpose of making a decision. Therefore, it is necessary to establish a scale for every considered criterion. The elements of the scale are denoted degrees, levels or ranks.

**Table 2.** Summary of methods to define multi-criteria utility functions ordered by complexity.

Goal attainment	Simple, just relevant/non-relevant categorization of preferences
Multi-Attribute Utility Theory (MAUT)	Utility function as a weighted sum of numeric values assigned to criteria, needs perfect global rationality
Preference relations	Modeled through binary relations to define preferences pairwise
Fuzzy logic	Introduces uncertainty through a credibility index
Valuation scale	Establishes a formal representation of the preference between alternatives

Table 2 summarizes the aforementioned methods to define multi-criteria utility functions represent the preferences of the decision maker related to the multiple criteria to be optimized. This table also orders them according to its complexity, starting with the simpler Goal Attainment method and ending with the completer Valuation Scale.

## 2.4 Multi-criteria Problems Solving Mechanisms

Once the optimization problem is modeled or formulated, the solution is found after the application of an optimization method.

Most optimization algorithms frequently imply an iterative searching process. Beginning with an initial approach to the solution, the algorithm performs consecutive steps towards the termination point. The search strategy states the difference among the diverse methods and there is no universal method applicable to any kind of problem. Table 3 shows a classification of the main optimization solving families.

**Table 3.** Classification of optimization solving methods.

Weighted sum [21]	The multiple objective functions are aggregated in a single function by the assignment of weights
Random search [13]	Generate random numbers to explore the search (feasible) space
Tabu search [14]	Iteratively make movements around the current solution constrained by a group of forbidden or tabu movements
Physical programming [31]	Incorporate preferences without the need of weight assignment. Address both design metrics and constraints in the same way, integrating them into the utility function
Lexicographic [6]	Objective functions are processed in a hierarchical basis
Genetic and evolutionary [12, 15]	Imitate the optimization process of the natural selection. Employ techniques such as heredity, mutation, natural selection or factor recombination to explore the feasible space and select the current solution
Simulated annealing [14]	Imitate the iterative process of cold and heat application for metal annealing by increasing or decreasing the difference between the ideal solution and the current approach
Ant colony optimization (ACO) and swarm optimization [40]	Imitate animal behavior related to their intra-group communication or their search for the optimal ways towards the food
Outranking methods [46]	Build an ordered relation of the feasible alternatives based on the defined preferences over a set of criteria to eventually complete a recommendation

## 2.5 Fairness Consideration

Traditionally, the goal of any optimization problem has been the search for the optimum solution for a given situation among all the possible ones in the feasible solution space. This optimality meaning has often been understood as a Pareto Optimum, i.e., the result of the maximization/minimization of the objective functions (or criteria), where the result of none of the objective functions can be improved, but at the expense of worsening another one. Finding a Pareto-optimal solution means finding the technically most efficient solution. And applying this concept to the field of networking, this optimality results on the optimum distribution of resources among the flows traveling through the network.

Obviously, an optimal distribution of resources not always implies an equitable use of them. Indeed, in some cases it may lead to absolutely unfair situations that entail the exhaustion of some resources. In that sense, the efficient assignment of resources derived from the direct application of optimization

algorithms may leave without service some customers or final users, due to the provision of all the benefit to others (see examples in [9,38]). Obviously, the global utility of the system is the maximum, but the result is clearly unfair, and the situation worsens as the heterogeneity of the final users increases.

The conflict between the maximization of the benefit, the optimal resource allocation and the fairness of the distribution is a field that has been widely analyzed in Economy, as part of microeconomics or public finances. The conclusion is that the incompatibility between fairness and efficiency is not a design problem of the optimization algorithms, but of the formulation of the problem to be optimized, where the fairness concept must be included. The difficulty rises up since efficiency is an objective or technical goal that, in consequence, can be measured and assessed quantitatively. This has nothing to do with the concept of fairness, a subjective concept whose assessment is not trivial.

Although fairness may initially seem to be easy to define, it has a variety of aspects that complicate its proper delimitation. Taking the sense of equanimity, an equitable distribution of resources could be defined as an evenly split available resource assignment among the flows competing for them. The disadvantage of this distribution is that it does not take into account the specific necessities of each flow. If all the flows obtain the same portion of resources, those with lower requirements benefit from a proportionally higher resource quantity.

Changing the definition of fair distribution to that assigning the resources proportionally on the basis of flow requirements is neither the ideal solution. In this case, the most consuming items are benefited, i.e., those which contribute more to the network congestion, to the detriment of lighter transmissions and consequently, of the global performance of the network.

In addition, other aspects such as cooperation must also be considered. There may be some nodes in the network not willing to give up their resources to other transmissions, and so, this kind of behavior should be punished. But, what happens when a node doesn't give up resources to the network due to the lack of them? It would be the case of a node with low battery or low capacity links. Would these be reason enough to reduce the transmission resources that have been assigned? In this case, would the distribution be fair? This conflict remains unsolved, although some approaches have been formulated and are discussed next.

The work in [8] presents several interpretations of the concept of fairness. In one hand, there is the widely accepted *max-min fairness* definition [38], usually employed in social science. It is based in the search for consecutive approaches to the optimum solution in a way that no individual or criterion can improve its state or utility if it means a loss for a weaker individual or criterion.

Translating this concept to communications, the distribution of network resources is considered max-min fair when all the minimum transmission rates of the data flows are maximized and all the maximum transmission rates are minimized. It is proven that this fairness interpretation is Pareto-efficient.

Another interpretation of fairness that also searches for the trade-off between efficiency and equity is the *proportional fairness* [26]. A resource distribution

among the network flows is considered proportional when the planned priority of a flow is inversely proportional to the estimated resource consumption of this flow. It can also be proven that the proportional fairness is Pareto-efficient.

Both aforementioned interpretations the bandwidth is shared to maximize some utility function for instantaneous flows. This means that the optimality of the resource assignment is measured for a static combination of flows. Taking into account the real random nature of the network traffic, it is necessary to define the utility in terms of the performance of individual flows with finite duration. And in this case, it is not so clear that the max-min or proportional fairness concepts reach an optimum result. With random traffic, the performance and, in consequence, the utility depend on precise statistics of the offered traffic and are hard, if not impossible, to be analytically assessed.

Sharing flows under a *balanced fairness* criterion [9], the performance becomes indifferent to the specific traffic characteristics, simplifying its formulation. The term balanced fairness comes from the necessary and sufficient relations that must be fulfilled to guarantee the insensitiveness in stochastic networks. This insensitiveness entails that the distribution of the active flow number and, in consequence, the estimated throughput, depends just on the main traffic offered in each route.

Balanced fairness makes it possible to approach the behavior of the elastic traffic over the network and, in addition to the insensitiveness property, it also makes it possible to find the exact probability of the distribution of concurrent flows in different routes and then evaluate the performance metrics.

The balanced fairness is not always Pareto-efficient, but in the case that existing one, it will be one of a kind.

### 3 Cost/Energy/\*-Aware Network and Cloud Services Management Scenarios

Once most well known multi-criteria optimization techniques are introduced, the next step is to analyze the application scenarios. This section overviews several research scenarios where energy-aware control of different systems has been considered as part of the ACROSS project. The scenarios include the following: modeling and analysis of performance-energy trade-off in data centers, characterization and energy-efficiency of applications in cloud computing, energy-aware load balancing in 5G HetNets and finally incorporating energy and cost to opportunistic QoE-aware scheduling.

#### 3.1 Modeling and Analysis of Performance-Energy Trade-Off in Data Centers

An increasing demand for green ICT has inspired the queueing community to consider energy-aware queueing systems. In many cases, it is no longer enough to optimize just the performance costs, but one should also take into account the energy costs. An idle server (waiting for an arriving job to be processed) in the

server farm of a typical data center may consume as much as 60% of the peak power. From the energy point of view, such an idle server should be switched off until a new job arrives. However, from the performance point of view, this is suboptimal since it typically takes a rather long time to wake the server up. Thus, there is a clear trade-off between the performance and energy aspects.

The two main metrics used in the literature to analyze the performance-energy trade-off in energy-aware queueing systems are ERWS and ERP. Both of them are based on the expected response time,  $E[T]$ , and the expected power consumption per time unit,  $E[P]$ . The former one, ERWS, is defined as their weighted sum,  $w_1 E[T] + w_2 E[P]$  and the latter one, ERP, as their product,  $E[T] \cdot E[P]$ . Also, generalized versions of these can be easily derived.

Here we model data centers as queueing systems and develop policies for the optimal control of the performance-energy trade-off. For a single machine the system is modeled as an M/G/1 queue. When considering a whole data-center, then a natural abstraction of the problem is provided by the dispatching problem in a system of parallel queues.

*Optimal Sleep State Control in M/G/1 Queue:* Modern processors support many sleep states to enable energy saving and the deeper the sleep state the longer is the setup delay to wake up from the sleep state. An additional feature in the control is to consider if it helps to wait for a random time (idling time) after busy period before going to sleep. Possible approaches for the sleep state selection policy include: randomized policy, where processor selects the sleep state from a given (optimized) distribution, or sequential policy, where sleep states are traversed sequentially starting from the lightest sleep state to the deepest one. Analysis of such a queueing system resembles that of classical vacation models.

Gandhi et al. see [17], considered the M/M/1 FIFO queue with deterministic setup delay and randomized sleep state selection policy but without the possibility of the idle timer, i.e., the timer is either zero or infinite, and they showed for the ERP metric that the optimal sleep state selection policy is deterministic, i.e., after busy period the system goes to some sleep state with probability 1 (which depends on the parameters). Maccio and Down [29] added the possibility of an exponential idle timer in the server before going to sleep, and showed for the ERWS cost metrics and for exponential setup delays that the optimal idle timer control still sets the idle timer equal to zero or infinite, i.e., the idle timer control remains the same. Gebrehiwot et al. considered the more general M/G/1 model with generally distributed service times, idle timer distributions and setup delays, both ERP and ERWS cost metrics (and even slightly more generalized ones) and randomized/sequential sleep state selection policies. Assuming the FIFO service discipline, it was shown in [20] that even after all the generalizations the optimal control finally remains the same: the optimal policy (a) either never uses any sleep states or (b) it will directly go to some deterministic sleep state and wake up from there. This result was shown to hold for the Processor Sharing (PS) discipline in [19] and for the Shortest Remaining Processing Time (SRPT) discipline in [18]. Thus, it is plausible that the result holds for any work-conserving discipline.



*Energy-Aware Dispatching with Parallel Queues:* The data center can be modeled as a system of parallel single-server queues with setup delays. The system receives randomly arriving jobs with random service requirements. The problem is then to identify for each arrival where to dispatch arriving new jobs based on state information available about the system, e.g., the number of jobs in the other queues. Another modeling approach is to consider a centralized queue with multiple servers, i.e., the models are then variants of the multiserver  $M/M/n$  model.

In the parallel queue setting and without any energy-aware considerations, the optimality of the JSQ policy for minimizing the mean delay with homogeneous servers is one classical result, see [48]. However, in an energy-aware setting the task is to find a balance for using enough servers to provide reasonably low job delay while taking into account the additional setup delay costs, and to let other servers sleep to save energy. Achieving this is not at all clear. For the centralized queue approach, Gandhi et al. proposed the delayed-off scheme, where servers upon a job completion use an idle timer, wait in the idle state for this time before going to sleep, and new jobs are sent to idle servers if one is available or otherwise some sleeping server is activated. An exact analysis under Markovian assumptions was done in [16], and it was shown that by appropriately selecting the mean idle timer value, the system keeps a sufficient number of servers in busy/idle state and allows the rest to sleep. An important result has been only recently obtained by Mukherjee et al. in [33], which considers the delayed-off scheme in a distributed parallel queue setting; it was shown that asymptotically delayed-off can achieve the same delay scaling as JSQ, i.e., is asymptotically delay optimal, and at the same time leaves a certain fraction of servers in a sleep state, independent of the value of the idle timer and the setup delay. This result holds asymptotically when the server farm is large with thousands of servers.

However, in a small/moderate sized data center there is still scope for optimization. In this setting the use of MDP (Markov Decision Process) and Policy Iteration has been recently considered by Gebrehiwot et al. in [28], where the data center is assumed to consist of two kinds of servers: normal always-on servers and instant-off servers, which go to sleep immediately after queue empties, i.e., there are no idle timers, and an explicit near optimal policy is obtained for minimizing the ERWS metric that uses as state the number of jobs in the queues and the busy/sleep status. Also, size-aware approaches with MDP have been recently applied by Hyytiä et al. in [24,25].

### 3.2 Characterization and Energy-Efficiency of Applications in Cloud Computing

**Modeling Applications.** With the goal of improving energy efficiency in cloud computing, several authors have studied the different factors that are causing energy loss and energy waste in data centers. In [32], the different aspects are discussed in detail, and idle runs are discussed as one of the causes for energy waste, as already mentioned earlier in this chapter. Low power modes have been proposed in the literature both for servers and storage components, however

their benefits are often limited due to their transition costs and inefficiencies. To improve energy efficiency and reduce the environmental impact of federated clouds, in the EU project ECO<sub>2</sub>Clouds [47] an adaptive approach to resource allocation is proposed, based on monitoring the use and energy consumption of resources, and associating it to running applications. The demand for resources can therefore associated to applications requesting resources, rather than only to the scheduling of resources and tasks in the underlying cloud environment.

Along this line, we have studied within ACROSS how different types of applications make use of resources, with the goal of improving energy efficiency.

As mentioned in Sect. 3.1, to compare different solutions in terms of response time and power consumption, the two main approaches are ERWS and ERP. An alternative, which allows evaluating energy efficiency at application level, is the *energy per job* indicator. This indicator allows comparing different solutions in terms of work performed, rather than on performance parameters, and to discuss ways of improving energy efficiency of applications in terms of application-level parameters.

Another aspect which has been considered is that increasing resources is not always beneficial in terms of performances, as the systems may present bottlenecks in their execution which may cause inefficiencies in the system: in some cases, the additional resources will worsen energy efficiency, as the new resources are not solving the problem and are themselves underutilized. As a consequence, in considering energy efficiency in applications in clouds, some aspects can better characterize the use of resources:

- *Shared access to resources*: during their execution application can request access to shared resources with an impact on energy consumption due to synchronization and waiting times.
- *The characterization of the application execution patterns*: batch applications and transactional applications present different execution patterns: in batch applications the execution times are usually longer with larger use of resources, but response time constraints are not critical; in transactional applications, response times are often subject to constraints and the allocated resources must guarantee they are satisfied.

These application-level aspects have an impact on the resource allocation criteria in different cases. In the following, we discuss how to model batch and transactional applications considering these aspects with the goal of choosing the number of resources to be associated to an application in terms of VMs with the goal of minimizing the energy-per-job parameter.

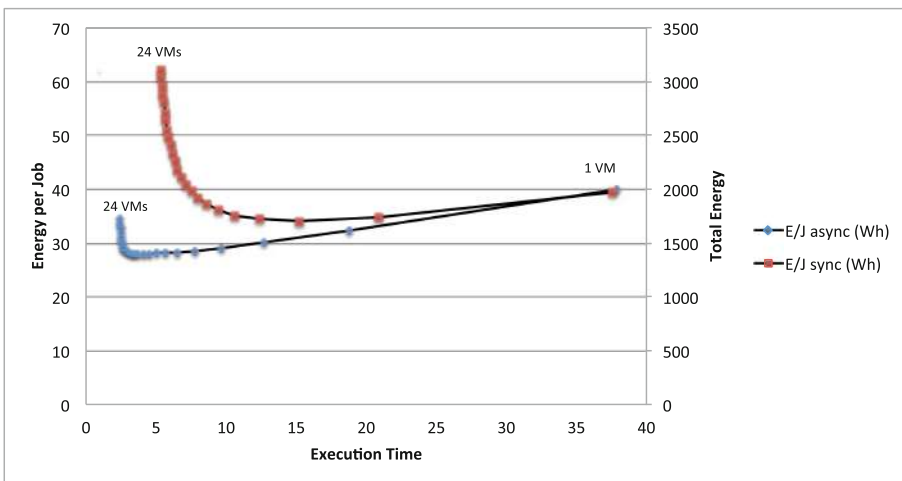
*Batch Applications.* Batch applications have been studied in detail in [22] to consider the following aspects: number of VMs allocated for executing a batch of similar applications, shared resources (in particular shared storage access and access synchronization), heterogeneous deployments environments for VMs, with servers with different capacity.

While for the details we refer to [22], we summarize here the main characteristics of the approach. The general goal is to minimize idle time to improve

energy efficiency, while avoiding to increase execution time for each application in the batch, which would result in an increase of the total energy. We assume that in computing the energy per job, idle time is distributed to all applications being run on the system in an equal basis. Queuing models have been developed to represent applications, in terms of computing nodes to execute the application and storage nodes for data access, which is assumed to be shared, with the possibility of choosing between asynchronous access and synchronous access (with synchronization points). In both cases the critical point is represented by the ratio between the service time for computing nodes and the service time for storage access: going beyond this point the energy per job is increasing without significant benefit in execution times.

An example is shown in Fig. 2, where it is clear that increasing the number of VMs for an application after the critical point is mainly resulting in a loss of energy efficiency, both with synchronous and asynchronous storage access.

*Transactional Workloads.* For transactional workloads, the main application-level parameter affecting energy consumption is the arrival rate. In fact, assuming an exponential distribution of arrivals, if the arrival rate  $\lambda$  is much lower than the service time, the idle times will be significant. On the other hand, getting closer to service time, the response time will increase, as shown in Fig. 3. The details of the computations can be found in [23]. The paper also describes how different load distribution policies for VMs can influence energy-per-job. Assuming again that idle power is uniformly distributed to all VMs running on the same host, three policies have been evaluated: (1) distributing the load equally; (2) allocating larger loads to VMs with lower idle power; (3) allocating larger loads to VMs with higher idles power. Initial simulation results result in Policy 2 being the worst, while Policy 1 and 3 are almost equivalent, with Policy 1 resulting in better energy-per-job and Policy 3 in better response times [23].



**Fig. 2.** Energy per job in batch applications

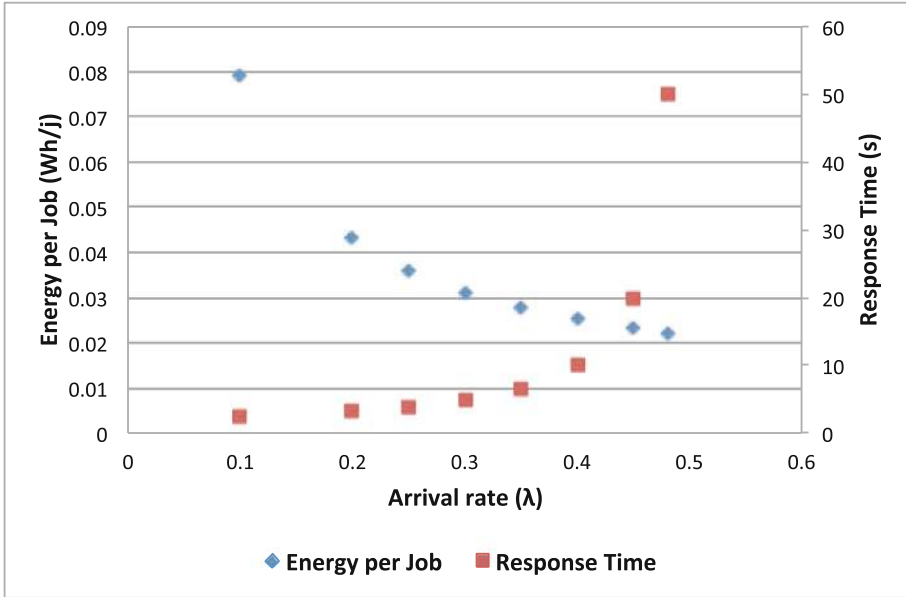


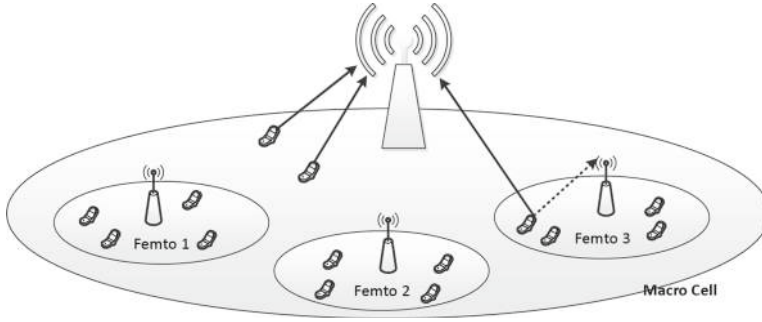
Fig. 3. Energy-per-job in transactional applications [23]

### 3.3 Energy-Aware Load Balancing in 5G HetNets

The exponential growth of mobile data still continues and heterogeneous networks have been introduced as a vital part of the network architecture of future 5G networks. Heterogeneous networks (HetNet) especially alleviate the problem that the user data intensity may have spatially large variations. These are network architectures with small cells (e.g., pico and femtocells) overlaying the macrocell network. The macrocells are high power base stations providing the basic coverage to the whole cell area, while the small cells are low power base stations used for data traffic hotspot areas within a macrocell to improve spectral efficiency per unit area or for areas that the macrocell cannot cover efficiently.

In HetNets, when a user arrives in the coverage area of a small cell it can typically connect to either the local small cell or to the macrocell, as illustrated in Fig. 4. Typically, the small cells offer in its coverage area a possibility for achieving high transmission rates. However, depending on the congestion level at the small cell it may be better from the system point of view to utilize the resources of the macrocell instead. This raises the need to design dynamic load balancing algorithms. In 5G networks the energy consumption of the system will also be an important factor. Thus, the load balancing algorithms must be designed so that they take into account both the performance of the system, as well as the energy used by the whole system.

Consider a single macrocell with several small cells inside its coverage area. The small cells are assumed to have a wired backhaul connection to the Internet.



**Fig. 4.** User inside a femtocell may connect either to the local small cell (femto) or the macrocell to achieve better load balancing.

They typically also operate on a different frequency than the macrocell and hence do not interfere with the transmissions of the macrocell. From the traffic point of view, each cell can be considered, whether it is the macrocell or a small cell, as a server with its own queue each having its own characteristics. The traffic itself may consist, for example of elastic data flows. The load balancing problem then corresponds to a problem of assigning arriving jobs or users to parallel queues. The difference to a classical dispatching problem, where an arrival can be routed to any queue, is that in this case the arrival can only select between two queues: its own local queue or the queue representing the macrocell.

In order to include the energy aspects in the model, the macrocell must be assumed to be operating at full power continuously. This is because the macrocell provides the control infrastructure and the basic coverage in the whole macrocell region and it can not be switched off. However, depending on the traffic situation it may be reasonable to switch off a low power small cell since the small cells typically have power consumption at least an order of magnitude lower than the macrocell. The cost of switching off a base station is that there may be a significant delay, the so-called set up delay, when turning the base station back on again. The queueing models used for the small cells must then be generalized to take this into account.

The resulting load balancing problem that optimizes for example the overall weighted sum of the performance and the energy parts of the whole system is difficult. However, it can be approached under certain assumptions by using the theory of Markov Decision Processes. This has been done recently by Taboada et al. in [42], where the results indicate that a dynamic policy that knows the sleep state of the small cells and the number of flows when compared with an optimized randomized routing policy is better able to keep the small cells sleeping and it thus avoids the harmful effect of setup delays leading to gains for both the performance and energy parts, while at high loads the energy gain vanishes but the dynamic policy still gives a good improvement in the performance.

### 3.4 Incorporating Energy and Cost to Opportunistic QoE-Aware Scheduling

One of the fundamental challenges that network providers nowadays face is the management for sharing network resources among users' traffic flows so that most of traditional scheduling strategies for resource allocation have been oriented to the maximization of objective quality parameters. Nevertheless, considering the importance and the necessity of network resource allocation for maximizing subjective quality, scheduling algorithms aimed at maximizing users' perception of quality become essential.

Thus, to overcome the lacks found in the field of traffic flow scheduling optimization, during the last years we have analyzed the following three stochastic and dynamic resource allocation problems:

1. Subjective quality maximization when channel capacity is constant [44],
2. Subjective quality maximization in channels with time-varying capacity [43],
3. Mean delay minimization for general size distributions in channels with time-varying capacity [41, 45].

Since these problems are analytically and computationally unfeasible for finding an optimal solution, we focus on designing simple, tractable and implementable well-performing heuristic priority scheduling rules.

For this aim, our research is focused on the Markovian Decision Processes (MDP) framework and on Gittins and Whittle methods [41, 43–45] to obtain scheduling index rule solutions. In this way, first of all, the above scheduling problems are modeled in the framework of MDPs. Later, using methodologies based on Gittins or/and Whittle approaches for their resolution, we have proposed scheduling index rules with closed-form expression.

The idea of Gittins consists in allocating resources to jobs with the current highest productivity of using the resource. The Gittins index is the value of the charge that provides that the expected serving-cost to the scheduler is in balance with the expected reward obtained when serving a job in  $r$  consecutive time slots, which results in the ratio between the expected total reward earned and the expected time spent in the system when serving a job in  $r$  consecutive time slots.

On the other hand, the Whittle approach consists in obtaining a function that measures the dynamic service priority. For that purpose, the optimization problem formulated as a Markov Decision Process (MDP) can be relaxed by requiring to serve a job per slot on average, which may allow introducing the constraint inside the objective function. Then, it is further approached by Lagrangian methods and can be decomposed into a single-job price-based parametrized optimization problem. Since the Whittle index is the break-even value of the Lagrangian parameter, it can be interpreted as the per cost of serving. In such a way, the Whittle index represents the rate between marginal reward and marginal work, where marginal reward (work) is the difference between the expected total reward earned (work done) by serving and not serving at an initial state and then employing a certain optimal policy.

As a first step towards ACROSS targeted multi-criteria optimization, it is worth mentioning the utility-based MDP employed in [43, 44] for QoE maximization. This function depended on delay only but we plan to extend it to a generic problem aimed at maximizing a multivariate objective function. Considering the meaning of work and reward in Whittle related modeling, such extension could demand the modification of the structure of the problem itself (i.e., alternative MDP) or just considering different criteria in the work/reward assignments.

Although we carried out some very preliminary tests with LP and AHP based articulation of preferences for QoE vs. energy optimization in [27] we plan to further analyze index rules techniques in the multi-criteria problem.

## 4 Current Technologies and Solutions

Research on energy-aware control has been actively pursued in the academia already for a long time, and Sect. 3 introduced several scenarios that have analyzed and given valuable insights to the fundamental tradeoff between energy efficiency and QoS/QoE. Due to the rising costs of energy, the industry is also actively developing solutions that would enable more energy efficient networks. Next we review industry efforts towards such architectures and finally we introduce a framework for energy-aware network management systems.

### 4.1 Industry Efforts for Integrating Energy Consumption in Network Controlling Mechanisms

New network technologies have been recently started to consider cost/energy issues in the early stages of the design and deployment process. Besides the infrastructure upgrade, the incorporation of such technologies requires the network managers must handle a number of real-time parameters to optimize Network energy /cost profile. These parameters include, among others, the sleep status of networks elements or the activation of mobile resources to provide extra coverage or change in performance status of some of the processors in the network.

The fact is that energy consumption in networks is rising. Therefore, network equipment requires more power and greater amounts of cooling. According to [27]. By 2017 more than 5 zettabytes of data will pass through the network every year. The period 2010–2020 will see an important increase in ICT equipment to provide and serve this traffic. Smartphones and tablets will drive the mobile traffic to grow up to 89 times by 2020, causing energy use to grow exponentially. For example, mobile video traffic is expected to grow 870%, M2M (IoT) 990% and Applications 129%. As a consequence, ICT will consume 6% of Total of Global Energy consumption: in 2013 it was 109,1 GW according to the energy use models at different network levels shown in Table 4.

**Table 4.** Energy use models.

Devices	Networks
PC's 36,9 GW	Home & Enterprise 9,5 GW
Printer 0,9 GW	Access 21,2 GW
Smartphones 0,6 GW	Metro 0,6 GW Aggregation and transport
Mobile 0,6 GW	Edge 0,7 GW
Tablets 0,2 GW	Core 0,3 GW
	Service Provider & Data Center 37,1 GW

One of the challenges the industry faces is how to support that growth in a sustainable and economically viable way. However, there is an opportunity for important reductions in the energy consumption because the networks are dimensioned in excess of current demand and even when the network is low in traffic the power used is very important and most of it is wasted [34]. The introduction of new technologies will provide a solution to improve the energy efficiency at the different scenarios (see Table 5).

**Table 5.** Scenarios for energy efficiency increase.

Home: Sleep mode	
Office: Cloud	
Access: VDL2, Vectoring, VoIP	Wireless Access: LTE Femto, Small, HetNet IP: MPLS Backhaul Fixed Wireless: Microwave Backhaul for Wireless 2G 3G, Fiber Copper: VDL2, Vectoring, VoIP, PON
Metro: IP/MPLS Transport, Packet Optical	
Edge: IP Edge	
IP Core: Next Gen IP Router and Transport (10 Gb)	
Service Provider & Data Center	

Current forecasts estimate that the trend will be to manage energy consumption and efficiency policies based on different types of traffic. Two organizations pursuing this goal are introduced next.

**GeSI Global e-Sustainability Initiative (GeSI)** [2]. Building a sustainable world In collaboration with members from major Information and Communication Technology (ICT) companies and organisations around the globe, the Global e-Sustainability Initiative (GeSI) is a leading source of impartial information, resources and best practices for achieving integrated social and environmental sustainability through ICT.



In a rapidly growing information society, technology presents both challenges and opportunities. GeSI facilitates real world solutions to real world issues both within the ICT industry and the greater sustainability community. We contribute to a sustainable future, communicate the industry's corporate responsibility efforts, and increasingly drive the sustainability agenda.

Members and Partners: ATT, Telecom Italia, Ericsson, KPN, Microsoft, Nokia, Nokia Siemens.

**Green Touch** [3]. GreenTouch is a consortium of leading Information and Communications Technology (ICT) industry, academic and non-governmental research experts dedicated to fundamentally transforming communications and data networks, including the Internet, and significantly reducing the carbon footprint of ICT devices, platforms and networks.

## 4.2 C-RAN: Access Network Architecture of Future 5G Networks

Cloud computing represents a paradigm shift in the evolution of ICT and has quickly become a key technology for offering new and improved services to consumers and businesses. Massive data centers, consisting of thousands of connected servers, are fundamental functional building blocks in the implementation of cloud services. With the rapidly increasing adoption of cloud computing, the technology has faced many new challenges related to scalability, high capacity/reliability demands and energy efficiency. At the same time, the huge increase in the processing capacity enables the use of more accurate information that the control decision may be based on. This justifies the development of much more advanced control methods and algorithms, which is the objective of the work as described earlier in Sect. 3.

To address the growing challenges, the research community has proposed several architectures for data centers, including FatTree, DCell, FiConn, Scafida and JellyFish [7]. On the other hand, vendors, such as, Google, Amazon, Apple, Google etc., have been developing their own proprietary solutions for the data centers which has created interoperability problems between service providers. To push forward the development of architectures addressing the challenges and to enable better interoperability between cloud service providers, IEEE has launched the IEEE Cloud Computing Initiative which is developing presently two standards in the area: IEEE P2301 Draft Guide for Cloud Portability and Interoperability Profiles and IEEE P2302 Draft Standard for Intercloud Interoperability and Federation.

Cloud-based approaches are also considered as part of the development of the future 5G networks. Namely, in the C-RAN (Cloud-Radio Access Network) architecture [35] the radio access network functionality is moved to the cloud. This means that all the radio resource management and cell coordination related functionality requiring complex computations are implemented in the cloud. This makes the functionality of the base stations simpler and hence also cheaper to manufacture. However, this places tough requirements on the computing capacity and efficiency of the centralized processing unit, essentially a data center, and

the interconnection network between the base stations and the data center. Several projects based on the C-RAN architecture have been initiated in the Next Generation Mobile Networks (NGMN) consortium and EU FP7 [10], and the C-RAN architecture will most likely be considered also in the standardization by 3GPP.

### 4.3 A Framework for Energy-Aware Network Management Systems

Considering the problem modeling and the existing optimization frameworks described in the previous sections, the challenge now is the integration of energy consumption in network controlling mechanisms. The networks in the data centers and in the operators world are showing a fast evolution with growing size and complexity that should be tackled by increased flexibility with softwarization techniques.

Emerging 5G Networks now exhibit extensive softwarization of all network elements: IoT, Mobile, and fiber optics-based transport core. This functions should be integrated in a network management environment with autonomous or semi-autonomous control response capabilities based on defined SLA's and applying policies and using simulated scenarios and past history learning.

By monitoring the energy parameters of radio access networks, fixed networks, front haul and backhaul elements, with the VNFs supporting the internal network processes, and by estimating energy consumption and triggering reactions, the energy footprint of the network (especially backhaul and fronthaul) can

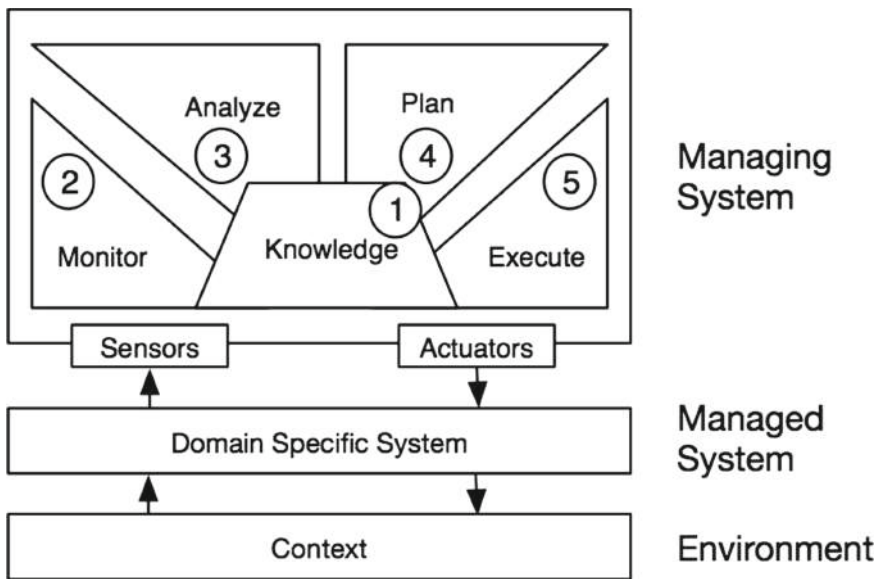
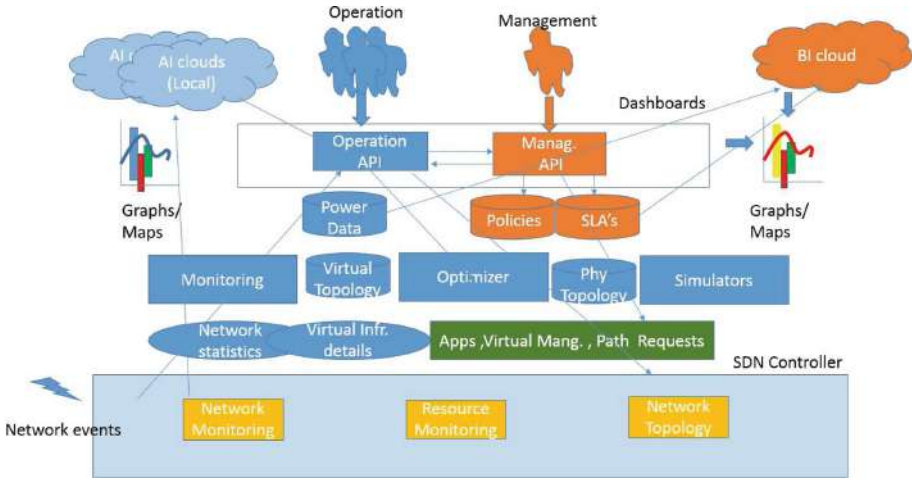


Fig. 5. MAPE-K diagram.



**Fig. 6.** Functional description of an energy management and monitoring application.

be reduced while maintaining QoS for each VNO or end user. An Energy Management and Monitoring Application can be conveniently deployed along a standard ETSI MANO and collect energy-specific parameters like power consumption and CPU loads (see Figs. 5 and 6). Such an Energy Management and Monitoring Application can also collect information about several network aspects such as traffic routing paths, traffic load levels, user throughput and number of sessions, radio coverage, interference of radio resources, and equipment activation intervals. All these data can be used to compute a virtual infrastructure energy budget to be used for subsequent analyses and reactions using machine learning and optimization techniques [11].

The application can optimally schedule the power operational states and the levels of power consumption of network nodes, jointly performing load balancing and frequency bandwidth assignment, in a highly heterogeneous environment. Also the re-allocation of virtual functions across backhaul and front haul will be done as part of the optimization actions, in order to cover virtual network functions to less power-consuming or less-loaded servers, thus reducing the overall energy demand from the network.

Designing software systems that have to deal with dynamic operating conditions, such as changing availability of resources and faults that are difficult to predict, is complex. A promising approach to handle such dynamics is self-adaptation that can be realized by a Monitor-Analyze-Plan-Execute plus Knowledge (MAPE-K) feedback loop. To provide evidence that the system goals are satisfied, regarding the changing conditions, state of the art advocates the use of formal methods.

Research in progress [1] tries to reinforce the approach of consolidating design knowledge of self-adaptive systems with the traditional tools of SLA's and policy modules and in particular with the necessity of defining the decision criteria

using formalized templates and making it understandable for a human operator or manager via the human interfaces and dashboards as shown in Fig. 6. This figure shows the proposed architecture of an advanced Network Monitoring and Management System that includes energy management. At the top are the two agents responsible for the management of the network: On one side those responsible for the negotiating the SLA's with the customers and of establishing the policies of the operation. On the other side those responsible for the detailed technical operation. These roles are supported by a set of applications and reside in the corresponding specialized cloud environments. The Business Intelligent cloud helps the Management API to generate dashboards for the optimization of the operation business results, issuing recommendations to the managers or autonomously implementing decisions. Those decisions will be based dynamically on contractual commitments, market conditions and customer's needs. The operational cloud supports the technical operations with specialized technical AI dashboards using available information from many sources: Network monitoring information including real and historical performance data from the network, power data and network statistics. Simulated data can be used to support the operation by providing hypothetical failure scenarios, possible solutions and the impact of applying those solutions. This helps together with the historical data with the analysis of the consequences of possible decisions when trying to solve specific incidents. As in the Business application the operational cloud will analyse the scenarios and select the optimal configuration autonomously or mediated by the operator interaction via the corresponding dashboards reducing the total energy footprint of the network. At the bottom of Fig. 6 is the SND network Controller with access to Network and Resource Monitoring and Topology that reacts to real Network events implementing the required network solution as directed by the layers above.

## 5 Conclusions and Foreseen Future Research Lines

### 5.1 Conclusions

This chapter has addressed the challenges of combining energy and QoS/QoE issues in the management mechanisms of network and cloud services. Unfortunately, these design parameters are usually conflicting and it is necessary to introduce multi-criteria optimization techniques in order to achieve the required trade-off solution.

So, as a first step, the common issues related to multi-objective optimization problems and mechanism have been depicted. These issues include typical preference articulation mechanisms, typical optimization methods and fairness considerations. Then, most well-known optimization methods have been briefly summarized in order to provide Internet of Services research community with a broad set of tools for properly addressing the inherent multi-criteria problems.

Finally, in the multiuser/multiservice environments considered in ACROSS, how resources are distributed and the impact into different kind of users must be carefully tackled. As analyzed, fairness is most of the times considered once

the algorithm has selected the most efficient (i.e., optimal) solution. However, the incompatibility between fairness and efficiency is not a design problem of the optimization algorithms, but of the formulation of the problem to be optimized, where the fairness concept must be included.

The next step is to model and analyze the problem of including the performance/energy trade-off into different scenarios in the scope of the ACROSS project. We start this analysis studying the use case of data centers modeled as queuing systems to develop policies for the optimal control of the QoS/QoE-energy balance. The trend in this area is to focus in small/moderate size data centers.

Then, the second scenario focuses on the way different applications use the resources available in cloud environments and its impact in terms of energetic cost. Considering that increasing resources does not always benefit the performance of the system, we analyze two application-level approaches in order to improve energy efficiency: the characterization of the application execution patterns and the shared access to resources.

Next, we show an example of energy-aware load balancing in 5G HetNets where cells of different sizes are used to adapt the coverage to the variations of user data traffic. We discuss the challenge of designing a load-balancing algorithm that considers both the performance of the system and the energy consumption of the whole system. The discussion suggests a MDP approach for the multi-criteria optimization problem.

The last analyzed scenario presents a network services provider that shares resources among different traffic flows. The goal here is to introduce energy and cost into opportunistic QoE-aware scheduling. The research focuses on the use of MDP framework to model the scheduling problem and the application of Gittins and Whittle methods to obtain scheduling index rule solutions.

Finally, the chapter compiles the current state of emerging technologies and foreseen solutions to the energy/performance trade-off issue in network and cloud management systems addressed in the ACROSS project. Based on the expected huge increase of network traffic and, in consequence, of energy consumption, the design of upcoming network management systems must face the challenge of addressing power efficiency while still meeting the KPIs of the offered services. Industry is already fostering innovative initiatives to integrate energy issues into network controlling mechanisms.

In this direction, we present C-RAN architecture as the cloud-based solution for the future 5G access network. This approach moves all the radio resource management and cell coordination functionality to the cloud. The increasing complexity of the service management and orchestration in the cloud requires advanced network control methods and algorithms. Therefore, as final conclusion, we suggest a framework to include energy awareness in network management systems that implements a MAPE-K feedback loop.

## 5.2 Future Work

The joint research accomplished in the scope of the COST ACROSS action has allowed the identification of common interests to develop in future collaborations. Remaining under the umbrella of Energy/Cost-aware network management, this future work will strongly rely on the application of multi-criteria optimization techniques in order to cope with conflicting performance objectives.

As previously concluded, the consideration of fairness in a optimization process does not fall to the multi-criteria optimization algorithm. On the contrary, it must be considered in the formulation of the design problem itself. Therefore one of the issues that will be addressed in future work grounded in the result of the COST ACROSS action is the inclusion of fairness among users/services/resource allocation in the definition network and services management optimization.

Besides, analyzing the problem of the introduction of energy-awareness in load balancing processes in 5G HetNets, another of the proposed future research lines is to use MDP and Policy Iteration in order to optimize the dispatching problem focusing in small/moderate size data centers. Similarly, we also found common interests in the development of further analysis of index rules techniques in the multi-criteria problem of opportunistic QoE-aware scheduling.

Finally, research in progress envisages innovative initiatives to integrate energy issues into network controlling mechanisms and interactive management approaches including self-adaption features.

**Acknowledgment.** The research leading to these results has been supported by the European Commission under the COST ACROSS action, supported by COST (European Cooperation in Science and Technology), and by Spanish MINECO under the project 5RANVIR (no. TEC2016-80090-C2-2-R).

## References

1. Decide Project ICT H2020 ID: 731533: ICT-10. Software Technologies. Technical report
2. GeSI home: thought leadership on social and environmental ICT sustainability. <http://gesi.org/>. Accessed 09 Oct 2017
3. GreenTouch. <https://s3-us-west-2.amazonaws.com/belllabs-microsite-greentouch/index.html>. Accessed 09 Oct 2017
4. Aleskerov, F.T.: Threshold utility, choice, and binary relations. *Autom. Remote Control* **64**(3), 350–367 (2003)
5. Andersson, J.: A survey of multiobjective optimization in engineering design. University, Linköping, Sweden, Technical Report No: LiTH-IKP-R-1097 (2000)
6. Belton, V., Stewart, T.J.: *Multiple Criteria Decision Analysis: An Integrated Approach*. Kluwer Academic Publishers, New York (2002)
7. Bilal, K., Malik, S.U.R., Khan, S.U., Zomaya, A.Y.: Trends and challenges in cloud datacenters. *IEEE Cloud Comput.* **1**(1), 10–20 (2014)
8. Bonald, T., Massoulié, L., Proutière, A., Virtamo, J.: A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Syst.* **53**(1–2), 65–84 (2006)

9. Bonald, T., Proutière, A.: On performance bounds for balanced fairness. *Perform. Eval.* **55**(1–2), 25–50 (2004)
10. Chih-Lin, I., Huang, J., Duan, R., Cui, C., Jiang, J.X., Li, L.: Recent progress on C-RAN centralization and cloudification. *IEEE Access* **2**, 1030–1039 (2014)
11. Casetti, C., Costa, L.C., Felix, K., Perez, G.M., Robert, M., Pedro, M., Pérez-Romero, J., Weigold, H., Al-Dulaimi, A., Christos, B.J., Gerry, F., Giovanni, G., Leguay, J., Mascolo, S., Papazois, A., Rodriguez, J.: Cognitive network management for 5G by 5GPPP working group on network management and QoS the path towards the development and deployment of cognitive networking list of contributors. Technical report, 5GPPP Network Management & Quality of Service Working Group (2017). <https://bscw.5g-ppp.eu/pub/bscw.cgi/d154625/NetworkManagement.WhitePaper.1.pdf>
12. Coello, C.: Handling preferences in evolutionary multiobjective optimization: a survey. In: *Proceedings of the 2000 Congress on Evolutionary Computation, CEC00* (Cat. No.00TH8512), vol. 1, pp. 30–37. IEEE (2000)
13. Ehrgott, M., Gandibleux, X.: *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Kluwer Academic Publishers, New York (2002)
14. Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Trans. Evol. Comput.* **8**(5), 425–442 (2004)
15. Fonseca, C., Fleming, P.: Genetic algorithms for multiobjective optimization: formulation discussion and generalization. In: *International Conference on Genetic Algorithms*, vol. 93, pp. 416–423, San Mateo, California (1993)
16. Gandhi, A., Doroudi, S., Harchol-Balter, M., Scheller-Wolf, A.: Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. *Queueing Syst.* **77**(2), 177–209 (2014)
17. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.A.: Optimality analysis of energy-performance trade-off for server farm management. *Perform. Eval.* **67**(11), 1155–1171 (2010)
18. Gebrehiwot, M.E., Aalto, S., Lassila, P.: Energy-aware server with SRPT scheduling: analysis and optimization. In: Agha, G., Van Houdt, B. (eds.) *QEST 2016*. LNCS, vol. 9826, pp. 107–122. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-43425-4\\_7](https://doi.org/10.1007/978-3-319-43425-4_7)
19. Gebrehiwot, M.E., Aalto, S., Lassila, P.: Energy-performance trade-off for processor sharing queues with setup delay. *Oper. Res. Lett.* **44**(1), 101–106 (2016)
20. Gebrehiwot, M.E., Aalto, S., Lassila, P.: Optimal energy-aware control policies for FIFO servers. *Perform. Eval.* **103**, 41–59 (2016)
21. Hamacher, H.W., Pedersen, C.R., Ruzika, S.: Multiple objective minimum cost flow problems: a review. *Eur. J. Oper. Res.* **176**(3), 1404–1422 (2007)
22. Ho, T.T.N., Gribaudo, M., Pernici, B.: Characterizing energy per job in cloud applications. *Electronics* **5**(4), 90 (2016)
23. Ho, T.T.N., Gribaudo, M., Pernici, B.: Improving energy efficiency for transactional workloads in cloud environments. In: *Proceedings of Energy-Efficiency for Data Centers (E2DC)*, Hong Kong, May 2017
24. Hyttia, E., Righter, R., Aalto, S.: Energy-aware job assignment in server farms with setup delays under LCFS and PS. In: *2014 26th International Teletraffic Congress (ITC)*, pp. 1–9. IEEE, September 2014
25. Hyttiä, E., Righter, R., Aalto, S.: Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure. *Perform. Eval.* **75–76**, 17–35 (2014)

26. Kelly, F.P., Maulloo, A.K., Tan, D.K.H.: Rate control for communication networks: shadow prices, proportional fairness and stability. *J. Oper. Res. Soc.* **49**(3), 237–252 (1998)
27. Liberal, F., Taboada, I., Fajardo, J.O.: Dealing with energy-QoE trade-offs in mobile video. *J. Comput. Netw. Commun.* **2013**, 1–12 (2013)
28. Gebrehiwot, M.E., Aalto, S., Lassila, P.: Near-optimal policies for energy-aware task assignment in server farms. In: 2nd International Workshop on Theoretical Approaches to Performance Evaluation, Modeling and Simulation (2017)
29. Maccio, V., Down, D.: On optimal policies for energy-aware servers. *Perform. Eval.* **90**, 36–52 (2015)
30. Marler, R., Arora, J.: Survey of multi-objective optimization methods for engineering. *Struct. Multi. Optim.* **26**(6), 369–395 (2004)
31. Marler, T.: A study of multi-objective optimization methods: for engineering applications. VDM Publishing, Saarbrücken (2009)
32. Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J., Vasilakos, A.V.: Cloud computing: survey on energy efficiency. *ACM Comput. Surv.* **47**(2), 33:1–33:36 (2014). <https://doi.org/10.1145/2656204>
33. Mukherjee, D., Dhara, S., Borst, S., van Leeuwen, J.S.H.: Optimal service elasticity in large-scale distributed systems. In: Proceedings of ACM SIGMETRICS (2017)
34. Nedeveschi, S., Popa, L., Iannaccone, G., Ratnasamy, S.: Reducing network energy consumption via sleeping and rate-adaptation. *NsDI* **8**, 323–336 (2008)
35. NGMN Alliance: suggestions on potential solutions to C-RAN. Technical report (2013). [http://www.ngmn.org/uploads/media/NGMN\\_CRAN\\_Suggestions\\_on\\_PotentialSolutions.to.CRAN.pdf](http://www.ngmn.org/uploads/media/NGMN_CRAN_Suggestions_on_PotentialSolutions.to.CRAN.pdf)
36. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006). <https://doi.org/10.1007/978-0-387-40065-5>
37. Öztürk, M., Tsoukiàs, A., Vincke, P.: Preference modelling. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol. 78, pp. 27–59. Springer, New York (2005). [https://doi.org/10.1007/0-387-23081-5\\_2](https://doi.org/10.1007/0-387-23081-5_2)
38. Pioro, M., Dzida, M., Kubilinskas, E., Ogryczak, W.: Applications of the max-min fairness principle in telecommunication network design. In: Next Generation Internet Networks, pp. 219–225. IEEE (2005)
39. Saaty, T.L.: Decision making with the analytic hierarchy process. *Int. J. Serv. Sci.* **1**(1), 83–98 (2008)
40. Shaw, K.: Including real-life problem preferences in genetic algorithms to improve optimisation of production schedules. In: Second International Conference on Genetic Algorithms in Engineering Systems, vol. 1997, pp. 239–244. IEE (1997)
41. Taboada, I., Jacko, P., Ayestaa, U., Liberal, F.: Opportunistic scheduling of flows with general size distribution in wireless time-varying channels. In: 2014 26th International Teletraffic Congress (ITC), pp. 1–9. IEEE, September 2014
42. Taboada, I., Aalto, S., Lassila, P., Liberal, F.: Delay- and energy-aware load balancing in ultra-dense heterogeneous 5G networks. *Trans. Emerg. Telecommun. Technol.* **28**, e3170 (2017)
43. Taboada, I., Liberal, F.: A novel scheduling index rule proposal for QoE maximization in wireless networks. *Abs. Appl. Anal.* **2014**, 1–14 (2014)
44. Taboada, I., Liberal, F., Fajardo, J.O., Ayesta, U.: QoE-aware optimization of multimedia flow scheduling. *Comput. Commun.* **36**(15–16), 1629–1638 (2013)



45. Taboada, I., Liberal, F., Jacko, P.: An opportunistic and non-anticipating size-aware scheduling proposal for mean holding cost minimization in time-varying channels. *Perform. Eval.* **79**, 90–103 (2014)
46. Tsoukias, A., Vincke, P.: A survey on non conventional preference modelling. *Ric. Operativa* **61**(5–48), 20 (1992)
47. Wajid, U., Cappiello, C., Plebani, P., Pernici, B., Mehandjiev, N., Vitali, M., Gienger, M., Kavoussanakis, K., Margery, D., García-Pérez, D., Sampaio, P.: On achieving energy efficiency and reducing co<sub>2</sub> footprint in cloud computing. *IEEE Trans. Cloud Comput.* **4**(2), 138–151 (2016). <https://doi.org/10.1109/TCC.2015.2453988>
48. Winston, W.: Optimality of the shortest line discipline. *J. Appl. Probab.* **14**(1), 181–189 (1977)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Traffic Management for Cloud Federation

Wojciech Burakowski<sup>1</sup>(✉), Andrzej Beben<sup>1</sup>, Hans van den Berg<sup>2</sup>,  
Joost W. Bosman<sup>3</sup>, Gerhard Hasslinger<sup>4</sup>, Attila Kertesz<sup>5</sup>, Steven Latre<sup>6</sup>,  
Rob van der Mei<sup>3</sup>, Tamas Pflanzner<sup>5</sup>, Patrick Gwydion Poullie<sup>7</sup>,  
Maciej Sosnowski<sup>1</sup>, Bart Spinnewyn<sup>6</sup>, and Burkhard Stiller<sup>7</sup>

<sup>1</sup> Warsaw University of Technology, Warsaw, Poland

{wojtek, abeben, m.sosnowski}@tele.pw.edu.pl

<sup>2</sup> Netherlands Organisation for Applied Scientific Research,  
The Hague, Netherlands

j.l.vandenberg@tno.nl

<sup>3</sup> Centrum Wiskunde & Informatica, Amsterdam, Netherlands

{j.w.bosman, r.d.van.der.mei}@cwi.nl

<sup>4</sup> Deutsche Telekom AG, Bonn, Germany

Gerhard.Hasslinger@telekom.de

<sup>5</sup> University of Szeged, Szeged, Hungary

{keratt, tamas.pflanzner}@inf.u-szeged.hu

<sup>6</sup> University of Antwerp - iMINDS, Antwerp, Belgium

{steven.latre, bart.spinnewyn}@uantwerpen.be

<sup>7</sup> University of Zürich - CSG@IfI, Zürich, Switzerland

{poullie, stiller}@ifi.uzh.ch

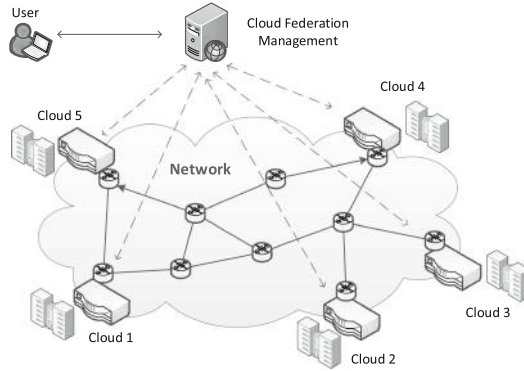
**Abstract.** The chapter summarizes activities of COST IC1304 ACROSS European Project corresponding to traffic management for Cloud Federation (CF). In particular, we provide a survey of CF architectures and standardization activities. We present comprehensive multi-level model for traffic management in CF that consists of five levels: Level 5 - Strategies for building CF, Level 4 - Network for CF, Level 3 - Service specification and provision, Level 2 - Service composition and orchestration, and Level 1 - Task service in cloud resources. For each level we propose specific methods and algorithms. The effectiveness of these solutions were verified by simulation and analytical methods. Finally, we also describe specialized simulator for testing CF solution in IoT environment.

**Keywords:** Cloud federation · Traffic management  
Multi-layer model · Service provision · Service composition

## 1 Introduction

Cloud Federation (CF) extends the concept of cloud computing systems by merging a number of clouds into one system. Thanks to this, CF has a potentiality to offer better service to the clients than it can be done by a separated cloud. This can happen since CF has more resources and may offer wider scope of services. On the other hand, the management of CF is more complex comparing to

this which is required for a standalone cloud. So, the effective management of resources and services in CF is the key point for getting additional profit from such system. CF is the system composing of a number of clouds connected by a network, as it is illustrated on Fig. 1. The main concept of CF is to operate as one computing system with resources distributed among particular clouds.



**Fig. 1.** Exemplary CF consisting of 5 clouds connected by network.

In this chapter we present a multi-level model for traffic management in CF. Each level deals with specific class of algorithms, which should together provide satisfactory service of the clients, while maintaining optimal resource utilization.

The structure of the chapter is the following. In Sect. 2 we present discussed CF architectures and the current state of standardization. The proposed multi-level model for traffic management in CF is presented in Sect. 3. Section 4 describes a simulation tool for analyzing performance of CF in Internet of Things (IoT) environment. Finally, Sect. 5 summarizes the chapter.

## 2 Cloud Federation Architectures

### 2.1 Cloud Architectural Views

In general CF is envisaged as a distributed, heterogeneous environment consisting of various cloud infrastructures by aggregating different Infrastructure as a Service (IaaS) provider capabilities coming from possibly both the commercial and academic area. Nowadays, cloud providers operate geographically diverse data centers as user demands like disaster recovery and multi-site back-ups became widespread. These techniques are also used to avoid provider lock-in issues for users that frequently utilize multiple clouds. Various research communities and standardization bodies defined architectural categories of infrastructure clouds. A current EU project on “Scalable and secure infrastructures for cloud operations” (SSICLOPS, [www.ssiclops.eu](http://www.ssiclops.eu)) focuses on techniques for the management of federated private cloud infrastructures, in particular cloud networking

techniques within software-defined data centers and across wide-area networks. The scope of the SSICLOPS project includes high cloud computing workloads e.g. within the CERN computing cloud ([home.cern/about/computing](http://home.cern/about/computing)) as well as cloud applications for securing web access under challenging demands for low delay. An expert group set up by the European Commission published their view on Cloud Computing in [1]. These reports categorize cloud architectures into five groups.

- Private Clouds consist of resources managed by an infrastructure provider that are typically owned or leased by an enterprise from a service provider. Usually, services with cloud-enhanced features are offered, therefore this group includes Software as a Service (SaaS) solutions like eBay.
- Public Clouds offer their services to users outside of the company and may use cloud functionality from other providers. In this solution, enterprises can outsource their services to such cloud providers mainly for cost reduction. Examples of these providers are Amazon or Google Apps.
- Hybrid Clouds consist of both private and public cloud infrastructures to achieve a higher level of cost reduction through outsourcing by maintaining the desired degree of control (e.g., sensitive data may be handled in private clouds). The report states that hybrid clouds are rarely used at the moment.
- In Community Clouds, different entities contribute with their (usually small) infrastructure to build up an aggregated private or public cloud. Smaller enterprises may benefit from such infrastructures, and a solution is provided by Zimory.
- Finally, Special Purpose Clouds provide more specialized functionalities with additional, domain specific methods, such as the distributed document management by Google's App Engine. This group is an extension or a specialization of the previous cloud categories.

The third category called hybrid clouds are also referred as cloud federations in the literature. Many research groups tried to grasp the essence of federation formation. In general, cloud federation refers to a mesh of cloud providers that are interconnected based on open standards to provide a universal decentralized computing environment where everything is driven by constraints and agreements in a ubiquitous, multi-provider infrastructure. Until now, the cloud ecosystem has been characterized by the steady rising of hundreds of independent and heterogeneous cloud providers, managed by private subjects, which offer various services to their clients.

Buyya et al. [2] envisioned Cloud Computing as the fifth utility by satisfying the computing needs of everyday life. They emphasized and introduced a market-oriented cloud architecture, then discussed how global cloud exchanges could take place in the future. They further extended this vision suggesting a federation oriented, just in time, opportunistic and scalable application services provisioning environment called InterCloud. They envision utility oriented federated IaaS systems that are able to predict application service behavior for intelligent down and up-scaling infrastructures. They list the research issues of

flexible service to resource mapping, user and resource centric Quality of Service (QoS) optimization, integration with in-house systems of enterprises, scalable monitoring of system components. They present a market-oriented approach to offer InterClouds including cloud exchanges and brokers that bring together producers and consumers. Producers are offering domain specific enterprise Clouds that are connected and managed within the federation with their Cloud Coordinator component.

Celesti et al. [3] proposed an approach for the federation establishment considering generic cloud architectures according to a three-phase model, representing an architectural solution for federation by means of a Cross-Cloud Federation Manager, a software component in charge of executing the three main functionalities required for a federation. In particular, the component explicitly manages:

1. the discovery phase in which information about other clouds are received and sent,
2. the match-making phase performing the best choice of the provider according to some utility measure and
3. the authentication phase creating a secure channel between the federated clouds. These concepts can be extended taking into account green policies applied in federated scenarios.

Bernstein et al. [4] define two use case scenarios that exemplify the problems of multi-cloud systems like

1. Virtual Machines (VM) mobility where they identify the networking, the specific cloud VM management interfaces and the lack of mobility interfaces as the three major obstacles and
2. storage interoperability and federation scenario in which storage provider replication policies are subject to change when a cloud provider initiates sub-contracting. They offer interoperability solutions only for low-level functionality of the clouds that are not focused on recent user demands but on solutions for IaaS system operators.

In the Federated Cloud Management solution [5], interoperability is achieved by high-level brokering instead of bilateral resource renting. Albeit this does not mean that different IaaS providers may not share or rent resources, but if they do so, it is transparent to their higher level management. Such a federation can be enabled without applying additional software stack for providing low-level management interfaces. The logic of federated management is moved to higher levels, and there is no need for adapting interoperability standards by the participating infrastructure providers, which is usually a restriction that some industrial providers are reluctant to undertake.

## 2.2 Standardization for Cloud Federation

Standardization related to clouds, cloud interoperability and federation has been conducted by the ITU (International Telecommunication Union) [6],

IETF (Internet Engineering Task Force) [7], NIST (National Institute of Standards and Technology) [8] and IEEE (Institute of Electrical and Electronics Engineers) [9]. In 2014, the ITU released standard documents on the vocabulary, a reference architecture and a framework of inter-cloud computing. The latter provides an overview, functional requirements and refers to a number of use cases. The overview distinguishes between:

- Inter-cloud Peering: between a primary and secondary CSP (i.e. Cloud Service Provider), where cloud services are provided by the primary CSP who establishes APIs (application programming interfaces) in order to utilize services and resources of the secondary CSP,
- Inter-cloud Intermediary: as an extension of inter-cloud peering including a set of secondary CSPs, each with a bilateral interface for support of the primary CSP which offers all services provided by the interconnected clouds, and
- Inter-cloud Federation: which is based on a set of peer CSPs interconnected by APIs as a distributed system without a primary CSP with services being provided by several CSPs. For each service, the inter-cloud federation may act as an inter-cloud intermediary with a primary CSP responsible for the service. The user population may also be subdivided and attributed to several CSPs.

The main functional requirements to set up and operate a cloud federation system are:

- Networking and communication between the CSPs,
- Service level agreement (SLA) and policy negotiations,
- Resource provisioning and discovery mechanisms,
- Resource selection, monitoring and performance estimation mechanisms,
- Cloud service switch over between CSPs.

Finally, the ITU [6] takes a number of use cases into account to be addressed by cloud interconnection and federation approaches:

- Performance guarantee against an abrupt increase in load (offloading),
- Performance guarantee regarding delay (optimization for user location),
- Guaranteed availability in the event of a disaster or large-scale failure,
- Service continuity (in the case of service termination of the original CSP), service operation enhancement and broadening service variety,
- Expansion and distribution of cloud storage, media and virtual data center,
- Market transactions in inter-cloud intermediary pattern and cloud service rebranding.

The standardization on cloud federation has many aspects in common with the interconnection of content delivery networks (CDN). A CDN is an infrastructure of servers operating on application layers, arranged for the efficient distribution and delivery of digital content mostly for downloads, software updates and video streaming. The CDN interconnection (CDNI) working group

of the IETF provided informational RFC standard documents on the problem statement, framework, requirements and use cases for CDN interconnection in a first phase until 2014. Meanwhile specifications on interfaces between upstream/downstream CDNs including redirection of users between CDNs have been issued in the proposed standards track [7]. CDNs can be considered as a special case of clouds with the main propose of distributing or streaming large data volumes within a broader service portfolio of cloud computing applications. The underlying distributed CDN architecture is also useful for large clouds and cloud federations for improving the system scalability and performance. This is reflected in a collection of CDNI use cases which are outlined in RFC 6770 [7] in the areas of:

- footprint extension,
- offloading,
- resilience enhancement,
- capability enhancements with regard to technology, QoS/QoE support, the service portfolio and interoperability.

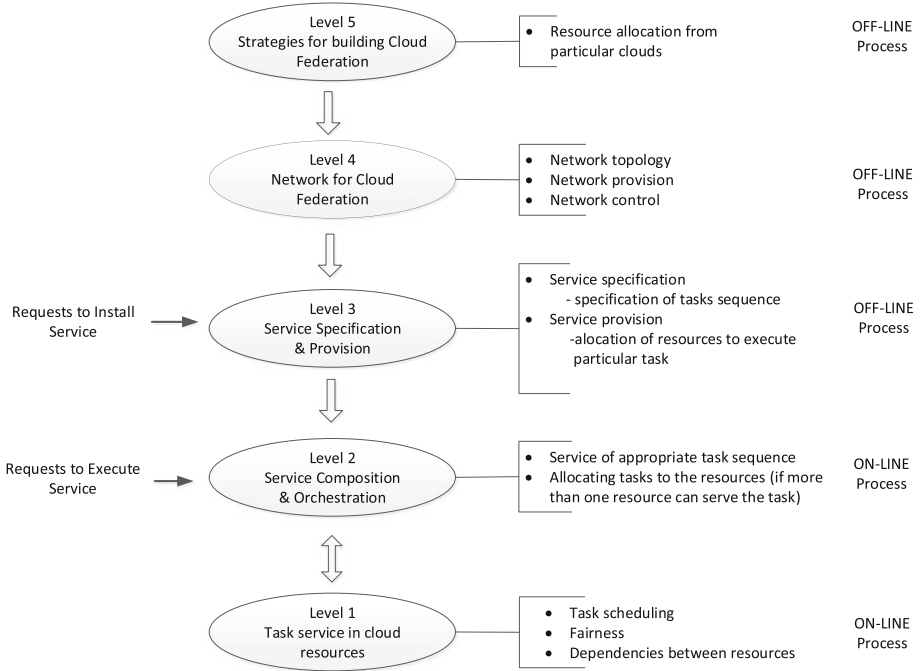
The CDNI concept is foreseen as a basis for CDN federations, where a federation of peer CDN systems is directly supported by CDNI. A CDN exchange or broker approach is not included but can be build on top of core CDNI mechanisms.

In 2013, NIST [8] published a cloud computing standards roadmap including basic definitions, use cases and an overview on standards with focus on cloud/grid computing. Gaps are identified with conclusions on priorities for ongoing standardization work. However, a recently started standards activity by the IEEE [9] towards intercloud interoperability and federation is still motivated by today's landscape of independent and incompatible cloud offerings in proprietary as well as open access architectures.

### 3 Multi-level Model for Traffic Management in Cloud Federation

Developing of efficient traffic engineering methods for Cloud Federation is essential in order to offer services to the clients on appropriate quality level while maintaining high utilization of resources. These methods deal with such issues as distribution of resources in CF, designing of network connecting particular clouds, service provision, handling service requests coming from clients and managing virtual resource environment. The proposed traffic management model for CF consists of 5 levels, as it is depicted on Fig. 2. Below we shortly discuss objectives of each level of the model.

Level 5: This is the highest level of the model which deals with the rules for merging particular clouds into the form of CF. The addressed issue is e.g. amount of resources which would be delegated by particular clouds to CF. We assume that the main reason for constituting federation is getting more profit



**Fig. 2.** Traffic management model for Cloud Federation

comparing to the situation when particular clouds work alone. So, this level deals with the conditions when CF can be attractive solution for cloud owners even if particular clouds differ in their capabilities, e.g. in amount of resources, client population and service request rate submitted by them.

**Level 4:** This level deals with design of the CF network for connecting particular clouds. Such network should be of adequate quality and, if it is possible, its transfer capabilities should be controlled by the CF network manager. The addressed issues are: required link capacities between particular clouds and effective utilization of network resources (transmission links). We assume that network capabilities should provide adequate quality of the offered by CF services even when resources allocated for a given service (e.g. virtual machines) come from different clouds. Effective designing of the network in question is especially important when CF uses network provided by a network operator based on SLA (Service Level Agreement) and as a consequence it has limited possibilities to control network. Currently such solution is a common practice.

**Level 3:** This level is responsible for handling requests corresponding to service installation in CF. The installation of new service requires: (1) specification of the service and (2) provision of the service. Specification of the service is provided in the form of definition of appropriate task sequence that is executed in CF when a client asks for execution of this service. Furthermore, provision of



the service corresponds to allocation of resources when particular tasks can be executed.

Level 2: This level deals with service composition and orchestration processes. So, the earlier specified sequence of tasks should be executed in response to handle service requests. Service composition time should meet user quality expectations corresponding to the requested service.

Level 1: The last and the lowest level deals with task execution in cloud resources in the case when more than one task is delegated at the same time to be served by a given resource. So, appropriate scheduling mechanisms should be applied in order to provide e.g. fairness for tasks execution. In addition, important issue is to understand dependencies between different types of resources in virtualized cloud environment.

### 3.1 Level 5: Strategy for Cloud Resource Distribution in Federation

#### 3.1.1 Motivation and State of the Art

Cloud Federation is the system that is built on the top of a number of clouds. Such system should provide some additional profits for each cloud owner in comparison to stand-alone cloud. In this section we focus on strategies, in which way clouds can make federation to get maximum profit assuming that it is equally shared among cloud owners.

Unfortunately, there are not too many positions dealing with discussed problem. For instance in [10] the authors consider effectiveness of different federation schemes using the M/M/1 queueing system to model cloud. They assume that profit get from a task execution depends on the waiting time (showing received QoS) of this task. Furthermore, they consider scenarios when the profit is maximized from the perspective of the whole CF, and scenarios when each cloud maximizes its profit. Another approach is presented in [11], where the author applied game theory to analyze the selfish behavior of cloud owner selling unused resources depending on uncertain load conditions.

#### 3.1.2 Proposed Model

In the presented approach we assume that capacities of each cloud are characterized in terms of number of resources and service request rate. Furthermore, for the sake of simplicity, it is assumed that both types of resources and executed services are the same in each cloud. In addition, execution of each service is performed by single resource only. Finally, we will model each cloud by well-known loss queueing system  $M/M/c/c$  (e.g. [12]), where  $c$  denotes number of identical cloud resources, arrival service request rate follows Poisson distribution with parameter  $\lambda$ , service time distribution is done by negative exponential distribution with the rate  $1/h$  ( $h$  is the mean service time). The performances of cloud system are measured by: (1)  $P_{loss}$ , which denotes the loss rate due to lack of available resources at the moment of service request arrival, and (2)  $A_{carried} = \lambda h(1 - P_{loss})$ , which denotes traffic carried by the cloud, that corresponds directly to the resource utilization ratio.

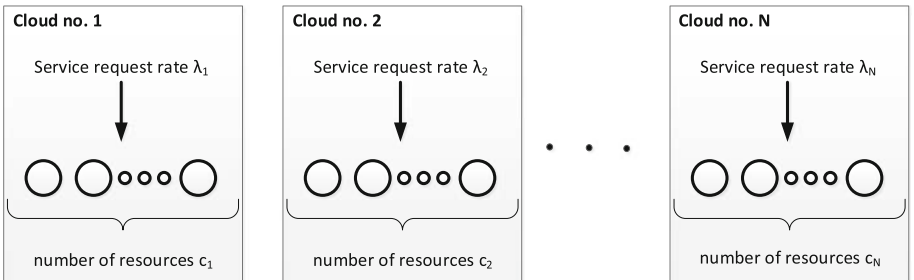
Now, let us search for the appropriate scheme for building CF system. For this purpose, let us consider a number, say  $N$ , of clouds that intend to build CF where the  $i$ -th cloud ( $i = 1, \dots, N$ ) is characterized by two parameters ( $\lambda_i$  and  $c_i$ ). In addition, the mean service times of service execution are the same in each cloud  $h_1 = h_2 = \dots = h_N = h$ . Subsequently we assume that  $h = 1$ , and as a consequence offered load  $A = \lambda h$  will be denoted as  $A = \lambda$ . Next, the assumed objective function for comparing the discussed schemes for CF is to maximize profit coming from resource utilization delegated from each cloud to CF. Furthermore, the profit is equally shared among clouds participating in CF. Such approach looks to be reasonable (at least as the first approach) since otherwise in CF we should take into account requests coming from a given cloud and which resource (from each cloud) was chosen to serve the request.

We consider three schemes:

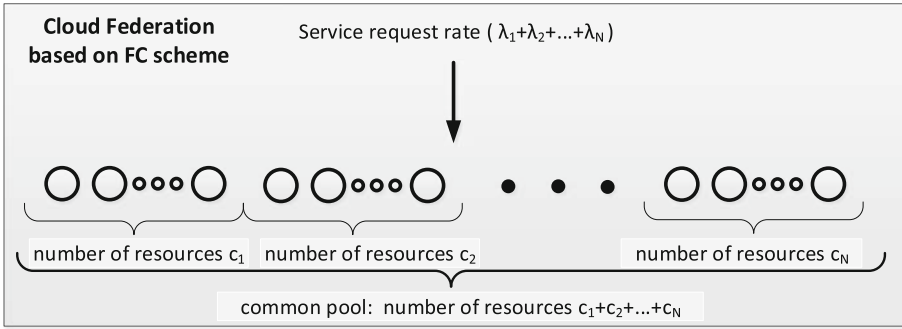
- Scheme no. 1 (see Fig. 3): this is the reference scheme when the clouds work alone, denoted by SC.
- Scheme no. 2 (see Fig. 4): this scheme is named as full federation and assumes that all clouds dedicate all their resources and clients to the CF system. This scheme we denote as FC.
- Scheme no. 3 (see Fig. 5): for this scheme we assume that each cloud can delegate to CF only a part of its resources as well as a part of service requests coming from its clients. This scheme we name as PCF (Partial CF).

First, let us compare the performances of schemes SC and FC in terms of resource utilization ratio and service request loss rate. The first observation is that FC scheme will have lower loss probabilities as well as better resource utilization ratio due to larger number of resources. But the open question is in which way to share profit gained from FC scheme when the clouds are of different capabilities? Table 1 shows exemplary results for the case, when the profit, which is consequence of better resources utilization, is shared equally among clouds.

The results from Table 1 show that, as it was expected, FC scheme assures less service request loss rate and better resource utilization ratio for most of clouds (except cloud no. 1 that is under loaded). Note, that if we share the profit equally, the clouds with smaller service requests rate can receive more profit from FC



**Fig. 3.** Scenario with clouds working in separate way



**Fig. 4.** Scenario with clouds creating Cloud Federation based on full federation scheme

scheme comparing to the SC scheme while the clouds with higher service request rate get less profit comparing to the SC scheme. So, one can conclude that FC scheme is optimal solution when the capabilities of the clouds are similar but if they differ essentially then this scheme simply fails.

Scheme no. 3 mitigates the drawbacks of the schemes no. 1 and no. 2. As it was above stated, in this scheme we assume that each cloud can delegate to CF only a part of its resources as well as a part of service request rate submitted by its clients. The main assumptions for PFC scheme are the following:

**Table 1.** Exemplary results comparing SC and FC schemes in terms of loss rate and resource utilization parameters. Number of clouds  $N = 5$ , values of  $\lambda$ :  $\lambda_1 = 0.2$ ,  $\lambda_2 = 0.4$ ,  $\lambda_3 = 0.6$ ,  $\lambda_4 = 0.8$ , the same mean service times  $h_1 = h_2 = h_3 = h_4 = h_5 = 1$ , Number of resources in each cloud:  $c_1 = c_2 = c_3 = c_4 = c_5 = 10$ .

Cloud characteristics			SC scheme		FC scheme	
No.	Service requests rate	Number of resources	Resource utilization	Loss rate [%]	Resource utilization	Loss rate[%]
1	2	10	0.2	<0.01	0.6	0.02
2	4	10	0.398	0.54	0.6	0.02
3	6	10	0.575	4.3	0.6	0.02
4	8	10	0.703	12	0.6	0.02
5	10	10	0.786	21	0.6	0.02

1. we split the resources belonging to the  $i$ -th cloud ( $i = 1, \dots, N$ ), say  $c_i$ , into 2 main subsets:
  - set of private resources that are delegated to handle only service requests coming from the  $i$ -th cloud clients
  - set of resources dedicated to Cloud Federation for handling service requests coming from all clouds creating Cloud Federation, denoted as  $c_{i3}$

2. we again split the private resources into two categories:
  - belonging to the 1st category, denoted as  $c_{i1}$ , which are dedicated as the first choice to handle service requests coming from the  $i$ -th cloud clients
  - belonging to the 2nd category, denoted as  $c_{i2}$ , which are dedicated to handle service requests coming from the  $i$ -th cloud clients that were not served by resources from 1st category as well as from common pool since all these resources were occupied.

The following relationship holds:

$$c_i = c_{i1} + c_{i2} + c_{i3}, \text{ for } i = 1, \dots, N. \quad (1)$$

The handling of service requests in PFC scheme is shown on Fig. 5. The service requests from clients belonging e.g. to cloud no.  $i$  ( $i = 1, \dots, N$ ) are submitted as the first choice to be handled by private resources belonging to the 1st category. In the case, when these resources are currently occupied, then as the second choice are the resources belonging to common pool. The number of common pool resources equals  $(c_{13} + c_{23} + \dots + c_{N3})$ . If again these resources are currently occupied then as the final choice are the resources belonging to the 2nd category of private resources of the considered cloud. The service requests are finally lost if also no available resources in this pool.

Next, we show in which way we count the resources belonging to particular clouds in order to get maximum profit (equally shared between the cloud owners). We stress that the following conditions should be satisfied for designing size of the common pool:

Condition 1: service request rate (offered load) submitted by particular clouds to the common pool should be the same. It means that

$$P_{loss1}(\lambda_1, c_{11})\lambda_1 = P_{loss2}(\lambda_2, c_{21})\lambda_2 = \dots = P_{lossN}(\lambda_N, c_{N1})\lambda_N \quad (2)$$

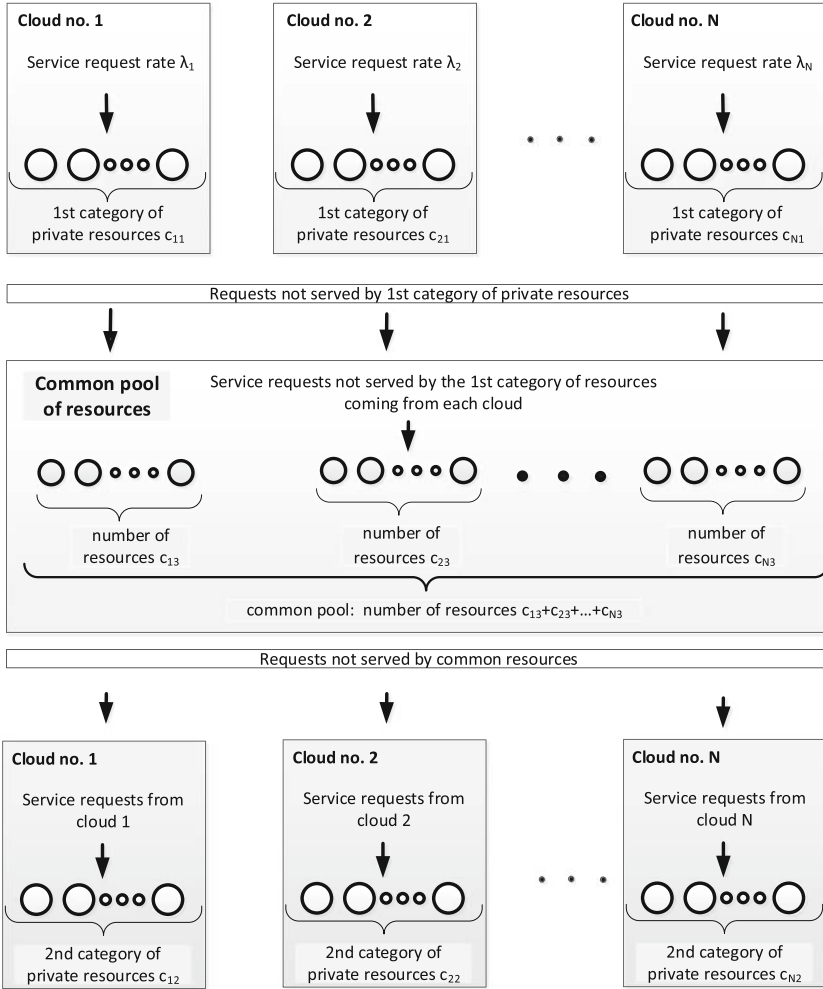
where the value of  $P_{loss}(\lambda_i, c_{i1})$  we calculate from the analysis of the system  $M/M/n/n$  by using Erlang formula:

$$P_{lossi}(\lambda_i, c_{i1}) = \frac{\frac{\lambda_i^{c_{i1}}}{c_{i1}!}}{\sum_{j=0}^{c_{i1}} \frac{\lambda_i^j}{j!}}$$

Note that we only require that mean traffic load submitted from each cloud to common pool should be the same. Let us note, that the service request arrival processes from each cloud submitted to this pool are generally different. It is due to the fact that these requests were not served by 1st category of private resources and as a consequence they are not still Poissonian.

Condition 2: the number of resources dedicated from each cloud to the common pool should be the same

$$c_{13} = c_{23} = \dots = c_{N3}.$$



**Fig. 5.** Handling of service requests in PFC scheme.

Finally, the algorithm for calculating resource distribution for each cloud is the following:

Step 1: to order  $\lambda_i$  ( $i = 1, \dots, N$ ) values from minimum value to maximum. Let the  $k$ -th cloud has minimum value of  $\lambda$ .

Step 2: to calculate (using Formula 2) for each cloud the values of the number of resources delegated to category 1 of private resources,  $c_{i1}$  ( $i = 1, \dots, N$ ) assuming that  $c_{k1} = 0$ .

Step 3: to choose the minimum value from set of  $(c_i - c_{i1})$  ( $i = 1, \dots, N$ ) and to state that each cloud should delegate this number of resources to the common pool. Let us note that if for the  $i$ -th cloud the value of  $(c_i - c_{i1}) \leq 0$  then no

common pool can be set and, as a consequence, not conditions are satisfied for Cloud Federation.

Step 4: to calculate from the Formula 1 the number of 2nd category of private resources  $c_{i2}$  ( $i = 1, \dots, N$ ) for each cloud.

### 3.1.3 Exemplary Results

Now we present some exemplary numerical results showing performances of the described schemes. The first observation is that when the size of common pool grows the profit we can get from Cloud Federation also grows.

Example: In this example we have 10 clouds that differ in service request rates while the number of resources in each cloud is the same and is equal to 10. Table 2 presents the numerical results corresponding to traffic conditions, number of resources and performances of the systems build under SC and PFC schemes. The required amount of resources belonging to particular categories were calculated from the above described algorithm.

Table 2 says that thanks to the PFC scheme we extend the volume of served traffic from 76,95 up to 84,50 (about 10%). The next step to increase Cloud Federation performances is to apply FC scheme instead of PFC scheme.

**Table 2.** Numerical results showing comparison between SC and PFC schemes.

Clouds			SC scheme		PFC scheme									
No.	Service requests rate	Number of resources	Load served by cloud	Loss rate [%]	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
1	7.5	10	6.75	10	7.50	0	5	5	0.00	2.34	4.82	7.16	3.5	0.41
2	8.4	10	7.22	14	7.50	1	4	5	0.89	2.10	4.82	7.82	6.3	0.60
3	8.4	10	7.22	14	7.50	1	4	5	0.89	2.10	4.82	7.82	6.3	0.60
4	9.3	10	7.61	18	7.50	2	3	5	1.79	1.75	4.82	8.35	10	0.74
5	9.3	10	7.61	18	7.50	2	3	5	1.79	1.75	4.82	8.35	10	0.74
6	10.2	10	7.91	22	7.50	3	2	5	2.69	1.26	4.82	8.77	14	0.86
7	10.2	10	7.91	22	7.50	3	2	5	2.69	1.26	4.82	8.77	14	0.86
8	11.1	10	8.17	26	7.50	4	1	5	3.58	0.68	4.82	9.08	19	0.91
9	11.1	10	8.17	26	7.50	4	1	5	3.58	0.68	4.82	9.08	19	0.91
10	12	10	8.38	30	7.50	5	0	5	4.49	0.00	4.82	9.31	23	0.92
Total	97.5	100	76.95		75	25	25	50	22.39	13.91	48.2	84.50		7.55

- L1: offered load to common pool
- L2: number of the 1st category of private resources
- L3: number of the 2nd category of private resources
- L4: number of resources delegated to common pool
- L5: load served by the 1st category of private resources
- L6: load served by the 2nd category of private resources
- L7: load served by common pool of resources
- L8: total load served by clouds
- L9: loss rate [%]
- L10: load served gain comparing to SC scheme

Unfortunately, it is not possible to be done in a straightforward way. It needs a moving of resources or service request rates between particular clouds. Table 3 presents moving of service request rates in the considered example to make transformation from PFC scheme into the form of FC scheme. For instance, cloud no. 1 should buy value of service request rate of 2.25 while cloud no. 10 should sell value of service request rate also of 2.25. Finally, after buying/selling process, one can observe that the profit gained from FC scheme is greater than the profit we have got from PFC scheme and now is equal to 91.50 (19% comparing to SC scheme and 8% comparing to PFC scheme).

Concluding, the presented approach for modeling different cloud federation schemes as FC and PFC could be only applied for setting preliminary rules for establishing CF. Anyway, it appears that in some cases by using simple FC scheme we may expect the problem with sharing the profit among CF owners. More precisely, some cloud owners may lost or extend their profits comparing to the case when their clouds work alone. Of course, more detailed model of CF is strongly required that also takes into account such characteristics as types of offered services, prices of resources, charging, control of service requests etc.

**Table 3.** Example showing system transformation into FC scheme.

Clouds			FC scheme						
No.	Service requests rate	Number of resources	Service requests rate to sell	Service requests rate to buy	L1	L2	L3	L4	L5
1	7.5	10	0	2.25	9.75	9.15	6.2	9.09	9.01
2	8.4	10	0	1.35	9.75	9.15	6.2	9.09	9.01
3	8.4	10	0	1.35	9.75	9.15	6.2	9.05	8.97
4	9.3	10	0	0.45	9.75	9.15	6.2	9.05	8.97
5	9.3	10	0	0.45	9.75	9.15	6.2	9.01	8.93
6	10.2	10	0.45	0	9.75	9.15	6.2	9.01	8.93
7	10.2	10	0.45	0	9.75	9.15	6.2	8.96	8.89
8	11.1	10	1.35	0	9.75	9.15	6.2	8.96	8.89
9	11.1	10	1.35	0	9.75	9.15	6.2	8.92	8.85
10	12	10	2.25	0	9.75	9.15	6.2	9.15	9.15
Total	97.5	100	5.85	5.85	97.5	91.5		91.5	91.5

L1: offered load to common pool  
L2: load served by common pool of resources  
L3: loss rate [%]  
L4: load served gain comparing to PFC scheme  
L5: load served gain comparing to SC scheme

## 3.2 Level 4: Network for Cloud Federation

### 3.2.1 Motivation and State of the Art

The services offered by CF use resources provided by multiple clouds with different location of data centers. Therefore, CF requires an efficient, reliable and secure inter-cloud communication infrastructure. This infrastructure is especially important for mission critical and interactive services that have strict QoS requirements. Currently, CF commonly exploits the Internet for inter-cloud communication, e.g. CONTRAIL [13]. Although this approach may be sufficient for non-real time services, i.e., distributed file storage or data backups, it inhibits deploying more demanding services like augmented or virtual reality, video conferencing, on-line gaming, real-time data processing in distributed databases or live video streaming. The commonly used approach for ensuring required QoS level is to exploit SLAs between clouds participating in CF. These SLAs are established on demand during the service provisioning process (see Level 3 of the model in Fig. 2) and use network resources coming from network providers. However, independently established SLAs lead to inefficient utilization of network resources, suffer scalability concerns and increase operating expenditures (OPEX) costs paid by CF. These negative effects become critical for large CFs with many participants as well as for large cloud providers offering plethora of services. For example, the recent experiences of Google cloud point out that using independent SLAs between data centers is ineffective [14]. Therefore, Google creates their own communication infrastructure that can be optimized and dynamically reconfigured following demands of currently offered services, planned maintenance operations as well as restoration actions taken to overcome failures.

### 3.2.2 Proposed Solution

The proposed approach for CF is to create, manage and maintain a Virtual Network Infrastructure (VNI), which provides communication services tailored for inter-cloud communication. The VNI is shared among all clouds participating in CF and is managed by CF orchestration and management system. Actually, VNI constitutes a new “service component” that is orchestrated during service provisioning process and is used in service composition process. The key advantages of VNI are the following:

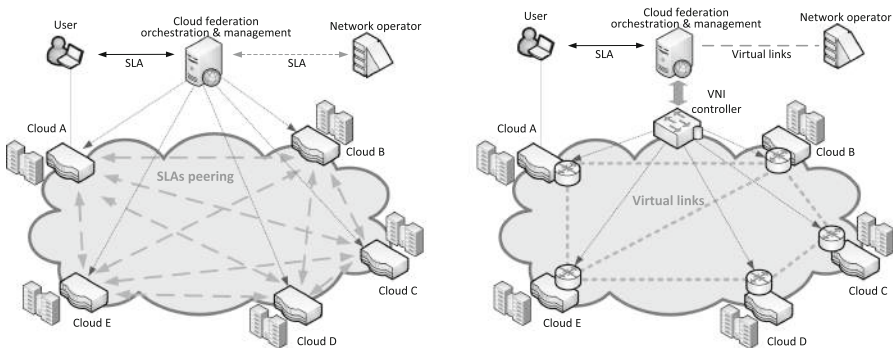
1. The common orchestration of cloud and VNI resources enables optimization of service provisioning by considering network capabilities. In particular, CF can benefit from advanced traffic engineering algorithms taking into account knowledge about service demands and VNI capabilities, including QoS guarantees and available network resources. The objective function of designed algorithms may cover efficient load balancing or maximization and fair share of the CF revenue.
2. New communication facilities tailored for cloud services:
  - The cloud services significantly differ in QoS requirements, e.g. interactive services are delay sensitive, while video on demand or big data storage



demands more bandwidth. Therefore, VNI should differentiate packet service and provide QoS guaranties following user's requirements. The key challenge is to design a set of Classes of Services (CoS) adequate for handling traffic carried by federation. These CoSs are considered in the service orchestration process.

- The VNI should offer multi-path communication facilities that support multicast connections, multi-side backups and makes effective communication for multi-tenancy scenarios. The key challenge is developing a scalable routing and forwarding mechanisms able to support large number of multi-side communications.

The VNI is created following the Network as a Service (NaaS) paradigm based on resources provided by clouds participating in CF. Each cloud should provide: (1) virtual network node, which is used to send, receive or transit packets directed to or coming from other clouds, and (2) a number of virtual links established between peering clouds. These links are created based on SLAs agreed with network provider(s). The VNI exploits advantages of the Software Defined Networking (SDN) concept supported by network virtualization techniques. It makes feasible separation of network control functions from underlying physical network infrastructure. In our approach, CF defines its own traffic control and management functions that operate on an abstract model of VNI. The management focuses on adaptation of VNI topology, provisioning of resources allocated to virtual nodes and links, traffic engineering, and costs optimization. On the other hand, this VNI model is used during the service composition phase for dynamic resource allocation, load balancing, cost optimization, and other short time scale operations. Finally, decisions taken by VNI control functions on the abstract VNI model are translated into configuration commands specific for particular virtual node.



(a) communication based on SLA peering. (b) communication based on VNI.

**Fig. 6.** Two reference network scenarios considered for CF.

Figure 6 shows the reference network scenarios considered for CF. Figure 6a presents the scenario where CF exploits only direct communication between peering clouds. In this scenario, the role of CF orchestration and management is limited to dynamic updates of SLAs between peering clouds. Figure 6b presents scenario where CF creates a VNI using virtual nodes provided by clouds and virtual links provided by network operators. The CF orchestration and management process uses a VNI controller to setup/release flows, perform traffic engineering as well as maintain VNI (update of VNI topology, provisioning of virtual links).

**The Control Algorithm for VNI.** The VNI is controlled and managed by a specialized CF network application running on the VNI controller. This application is responsible for handling flow setup and release requests received from the CF orchestration and management process as well as for performing commonly recognized network management functions related to configuration, provisioning and maintenance of VNI. The flow setup requires a specialized control algorithm, which decides about acceptance or rejection of incoming flow request. Admission decision is taken based on traffic descriptor, requested class of service, and information about available resources on routing paths between source and destination. In order to efficiently exploit network resources, CF uses multi-path routing that allows allocating bandwidth between any pair of network nodes up to the available capacity of the minimum cut of the VNI network graph. Thanks to a logically centralized VNI architecture, CF may exploit different multi-path routing algorithms, e.g. [15, 16]. We propose a new k-shortest path algorithm which considers multi-criteria constraints during calculation of alternative k-shortest paths to meet QoS objectives of classes of services offered in CF. We model VNI as a directed graph  $G(N, E)$ , where  $N$  represents the set of virtual nodes provided by particular cloud, while  $E$  is the set of virtual links between peering clouds. Each link  $u \rightarrow v, u, v \in N, u \rightarrow v \in E$ , is characterized by a  $m$ -dimensional vector of non-negative link weights  $w(u \rightarrow v) = [w_1, w_2, \dots, w_m]$  which relates to QoS requirements of services offered by CF. Any path  $p$  established between two nodes is characterized by a vector of path weights  $w(p) = [w_1(p), w_2(p), \dots, w_m(p)]$ , where  $w_i(p)$  is calculated as a concatenation of link weights  $w_i$  of each link belonging to the path  $p$ . The proposed multi-criteria, k-shortest path routing algorithm finds a set of Pareto optimum paths,  $f \in F$ , between each pair of source to destination nodes. A given path is Pareto optimum if its path weights satisfy constraints:  $w_i(f) < l_i, i = 1, \dots, m$ , where  $L$  is the vector of assumed constraints  $L = [l_1, l_2, \dots, l_m]$  and it is non-dominated within the scope of the considered objective functions. Note that proposed multi-criteria, k-shortest path routing algorithm runs off-line as a sub-process in CF network application. It is invoked in response to any changes in the VNI topology corresponding to: instantiation or release of a virtual link or a node, detection of any link or node failures as well as to update of SLA agreements.

The VNI control algorithm is invoked when a flow request arrives from the CF orchestration process. The algorithm is responsible for: (1) selection of a subset of feasible alternative routing paths which satisfy QoS requirements of

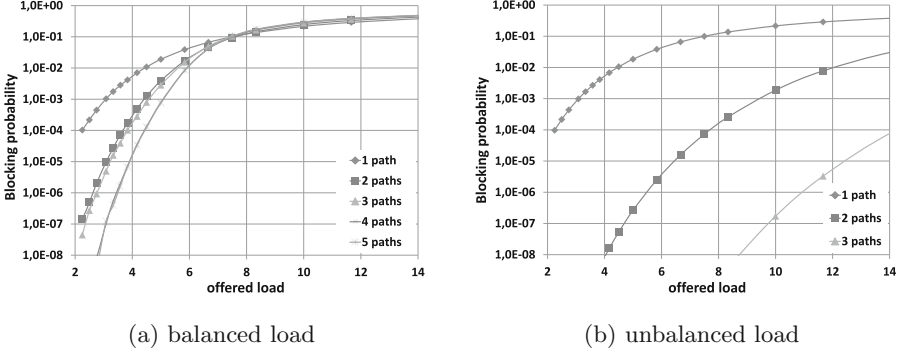
the requested flow. Notice, that bandwidth requested in the traffic descriptor may be satisfied by a number of alternative path assuming flow splitting among them, (2) allocation of the flow to selected feasible alternative routing paths, and (3) configuration of flow tables in virtual nodes on the selected path(s). The main objective of the proposed VNI control algorithm is to maximize the number of requests that are served with the success. This goal is achieved through smart allocation algorithm which efficiently use network resources. Remark, that flow allocation problem belongs to the NP-complete problems. The allocation algorithm has to take decision in a relatively short time (of second order) to not exceed tolerable request processing time. This limitation opt for using heuristic algorithm that find feasible solution in a reasonable time, although selected solution may not be the optimal one.

The proposed VNI control algorithm performs the following steps:

1. *Create a decision space.* In this step the algorithm creates a subset of feasible alternative paths that meet QoS requirements from the set of  $k$ -shortest routing paths. The algorithm matches QoS requirements with path weights  $w(p)$ . Then, it checks if selected subset of feasible alternative paths can meet bandwidth requirements, i.e. if the sum of available bandwidth on disjointed paths is greater than requested bandwidth. Finally, the algorithm returns the subset of feasible paths if the request is accepted or returns empty set  $\emptyset$ , which results in flow rejection.
2. *Allocate flow in VNI.* In this step, the algorithm allocates flow into previously selected subset of feasible paths. The allocation may address different objectives, as e.g. *load balancing*, *keeping the flow on a single path*, etc. depending on the CF strategy and policies. In the proposed algorithm, we allocate the requested flow on the shortest paths, using as much as possible limited number of alternative paths. So, we first try to allocate the flow on the latest loaded shortest path. If there is not enough bandwidth to satisfy demand, we divide the flow over other alternative paths following the load balancing principles. If we still need more bandwidth to satisfy the request, we consider longer alternative paths in consecutive steps. The process finishes when the requested bandwidth is allocated.
3. *Configure flow tables.* In the final step, the VNI control algorithm configures allocated paths using the abstract model of VNI maintained in the SDN controller. The actual configuration is performed by the management system of particular cloud using e.g. Open Flow protocol, net conf or other.

### 3.2.3 Performance Evaluation

The experiments focus on performance evaluation of the proposed VNI control algorithm. They are performed assuming a model of CF comprising  $n$  clouds offering the same set of services. A CF network assumes a full mesh topology where peering clouds are connected by virtual links. In this model the number of degree of freedom in selecting alternative paths is relatively large. Our experiments are performed by simulation. We simulate flow request arrival process and



**Fig. 7.** Blocking probabilities of flow requests served by VNI using different number of alternative paths.

analyze the system performances in terms of request blocking probabilities. We analyze the effectiveness of the VNI control algorithm under the following conditions: (1) number of alternative paths established in VNI, and (2) balanced and unbalanced load conditions. Notice, that results related to a single path, denoted as *1 path*, correspond to the strategy based on choosing only direct virtual links between peering clouds, while other cases exploit multi-path routing capabilities offered by VNI.

Figure 7 presents exemplary results showing values of request blocking probabilities as a function of offered load obtained for VNI using different number of alternative paths. Figure 7a corresponds to balanced load conditions where each relation of source to destination is equally loaded in the network. Furthermore, Fig. 7b shows values of blocking probabilities for extremely unbalanced load conditions, where flows are established between a chosen single relation. One can observe that using VNI instead of direct communication between peering clouds leads to significant decreasing of blocking probabilities under wide range of the offered load up to the limit of the working point at blocking probability at the assumed level of 0.1. One can also observe that by using alternative paths we significantly increase carried traffic under the same blocking probability. Moreover, the gain from using alternative paths is mostly visible if we use the first alternative path. Increasing the number of alternative paths above four or five practically yields no further improvement. The gain becomes especially significant under unbalanced load conditions.

### 3.3 Level 3: Service Provision

**Motivation.** While traditionally a cloud infrastructure is located within a data-center, recently, there is a need for geographical distribution [17]. For instance, cloud federation can combine the capabilities of multiple cloud offerings in order to satisfy the user's response time or availability requirements. Lately, this need for geo-distribution has led to a new evolution of decentralization. Most notably,

the extension of cloud computing towards the edge of the enterprise network, is generally referred to as fog or edge computing [18]. In fog computing, computation is performed at the edge of the network at the gateway devices, reducing bandwidth requirements, latency, and the need for communicating data to the servers. Second, mist computing pushes processing even further to the network edge, involving the sensor and actuator devices [19].

Compared to a traditional cloud computing environment, a geo-distributed cloud environment is less well-controlled and behaves in an ad-hoc manner. Devices may leave and join the network, or may become unavailable due to unpredictable failures or obstructions in the environment.

Additionally, while in a data-center heterogeneity is limited to multiple generations of servers being used, there is a large spread on capabilities within a geo-distributed cloud environment. Memory and processing means range from high (e.g. servers), over medium (e.g. cloudlets, gateways) to very low (e.g. mobile devices, sensor nodes). While some communication links guarantee a certain bandwidth (e.g. dedicated wired links), others provide a bandwidth with a certain probability (e.g. a shared wired link), and others do not provide any guarantees at all (wireless links).

Reliability is an important non-functional requirement, as it outlines *how* a software systems realizes its functionality [20]. The unreliability of substrate resources in a heterogeneous cloud environment, severely affects the reliability of the applications relying on those resources. Therefore, it is very challenging to host reliable applications on top of unreliable infrastructure [21].

Moreover, traditional cloud management algorithms cannot be applied here, as they generally consider powerful, always on servers, interconnected over wired links. Many algorithms do not even take into account bandwidth limitations. While such an omission can be justified by an appropriately over provisioned network bandwidth within a data-center, it is not warranted in the above described geo-distributed cloud networks.

**State of the Art.** In this section, the state of the art with regard to the Application Placement Problem (APP) in cloud environments is discussed. Early work on application placement merely considers nodal resources, such as Central Processing Unit (CPU) and memory capabilities. Deciding whether requests are accepted and where those virtual resources are placed then reduces to a Multiple Knapsack Problem (MKP) [22]. An MKP is known to be NP-hard and therefore optimal algorithms are hampered by scalability issues. A large body of work has been devoted to finding heuristic solutions [23–25].

When the application placement not only decides where computational entities are hosted, but also decides on how the communication between those entities is routed in the Substrate Network (SN), then we speak of *network-aware* APP. Network-aware application placement is closely tied to Virtual Network Embedding (VNE) [26]. An example of a network-aware approach is the work from Moens et al. [27]. It employs a Service Oriented Architecture (SOA), in which applications are constructed as a collection of communicating services. This optimal approach performs node and link mapping simultaneously.

In contrast, other works try to reduce computational complexity by performing those tasks in distinct phases [28,29].

While the traditional VNE problem assumes that the SN network remains operational at all times, the Survivable Virtual Network Embedding (SVNE) problem does consider failures in the SN. For instance, Ajtai et al. try and guarantee that a virtual network can still be embedded in a physical network, after  $k$  network components fail. They provide a theoretical framework for fault-tolerant graphs [30]. However, in this model, hardware failure can still result in service outage as migrations may be required before normal operation can continue.

Mihailescu et al. try to reduce network interference by placing Virtual Machines (VMs) that communicate frequently, and do not have anti-collocation constraints, on Physical Machines (PMs) located on the same racks [31]. Additionally, they uphold application availability when dealing with hardware failures by placing redundant VMs on separate server racks. A major shortcoming is that the number of replicas to be placed, and the anti-collocation constraints are user-defined.

Csorba et al. propose a distributed algorithm to deploy replicas of VM images onto PMs that reside in different parts of the network [32]. The objective is to construct balanced and dependable deployment configurations that are resilient. Again, the number of replicas to be placed is assumed predefined.

SiMPLE allocates additional bandwidth resources along multiple disjoint paths in the SN [33]. This proactive approach assumes splittable flow, i.e. the bandwidth required for a Virtual Link (VL) can be realized by combining multiple parallel connections between the two end points. The goal of SiMPLE is to minimize the total bandwidth that must be reserved, while still guaranteeing survivability against single link failures. However, an important drawback is that while the required bandwidth decreases as the number of parallel paths increases, the probability of more than one path failing goes up exponentially, effectively reducing the VL's availability.

Chowdhury et al. propose Dedicated Protection for Virtual Network Embedding (DRONE) [34]. DRONE guarantees Virtual Network (VN) survivability against single link or node failure, by creating two VNEs for each request. These two VNEs cannot share any nodes and links.

Aforementioned SVNE approaches [30–34] lack an availability model. When the infrastructure is homogeneous, it might suffice to say that each VN or VNE need a predefined number of replicas. However, in geo-distributed cloud environments the resulting availability will largely be determined by the exact placement configuration, as moving one service from an unreliable node to a more reliable one can make all the difference. Therefore, geo-distributed cloud environments require SVNE approaches which have a computational model for availability as a function of SN failure distributions and placement configuration.

The following cloud management algorithms have a model to calculate availability. Jayasinghe et al. model cloud infrastructure as a tree structure with arbitrary depth [35]. Physical hosts on which Virtual Machines (VMs) are hosted

are the leaves of this tree, while the ancestors comprise regions and availability zones. The nodes at bottom level are physical hosts where VMs are hosted. Wang et al. were the first to provide a mathematical model to estimate the resulting availability from such a tree structure [36]. They calculate the availability of a single VM as the probability that neither the leaf itself, nor any of its ancestors fail. Their work focuses on handling workload variations by a combination of vertical and horizontal scaling of VMs. Horizontal scaling launches or suspends additional VMs, while vertical scaling alters VM dimensions. The total availability is then the probability that at least one of the VMs is available. While their model suffices for traditional clouds, it is ill-suited for a geo-distributed cloud environment as link failure and bandwidth limitations are disregarded.

In contrast, Yeow et al. define reliability as the probability that critical nodes of a virtual infrastructure remain in operation over all possible failures [37]. They propose an approach in which backup resources are pooled and shared across multiple virtual infrastructures. Their algorithm first determines the required redundancy level and subsequently performs the actual placement. However, decoupling those two operations is only possible when link failure can be omitted and nodes are homogeneous.

**Availability Model.** In this section we introduce an availability model for geo-distributed cloud networks, which considers any combination of node and link failures, and supports both node and link replication. Then, building on this model, we will study the problem of guaranteeing a minimum level of availability for applications. In the next section, we introduce an Integer Linear Program (ILP) formulation of the problem. The ILP solver can find optimal placement configurations for small scale networks, its computation time quickly becomes unmanageable when the substrate network dimensions increase. Subsequently two heuristics are presented: (1) a distributed evolutionary algorithm employing a pool-model, where execution of computational tasks and storage of the population database (DB) are separated (2) a fast centralized algorithm, based on subgraph isomorphism detection. Finally, we evaluate the performance of the proposed algorithms.

*3.3.0.1 Application Requests.* We consider a SOA, which is a way of structuring IT solutions that leverage resources distributed across the network [38]. In a SOA, each application is described as its composition of services. Throughout this work, the collected composition of all requested applications will be represented by the instance matrix ( $\mathbf{I}$ ).

Services have certain CPU ( $\omega$ ) and memory requirements ( $\gamma$ ). Additionally, bandwidth ( $\beta$ ) is required by the VLS between any two services. A sub-modular approach allows sharing of memory resources amongst services belonging to multiple applications.

*3.3.0.2 Cloud Infrastructure.* Consider a substrate network consisting of nodes and links. Nodes have certain CPU ( $\Omega$ ) and memory capabilities ( $\Gamma$ ). Physical links between nodes are characterized by a given bandwidth ( $\mathbf{B}$ ). Both links and

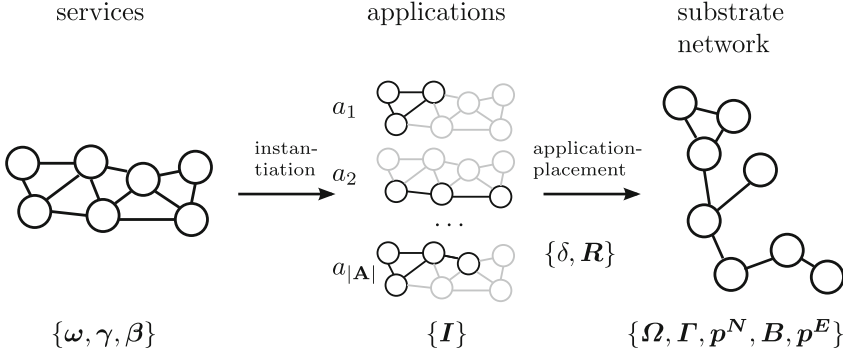
**Table 4.** Overview of input variables to the Cloud Application Placement Problem (CAPP).

Symbol	Description
<b>A</b>	Set of requested applications
<b>S</b>	Set of services
$\omega_s$	CPU requirement of service $s$
$\gamma_s$	Memory requirement of service $s$
$\beta_{s_1, s_2}$	Bandwidth requirement between services $s_1$ and $s_2$
$I_{a,s}$	Instantiation of service $s$ by application $a$ : 1 if instanced, else 0
<b>N</b>	Set of physical nodes comprising the substrate network
<b>E</b>	Set of physical links (edges) comprising the substrate network
$\Omega_n$	CPU capacity of node $n$
$\Gamma_n$	Memory capacity of node $n$
$p_n^N$	Probability of failure of node $n$
$B_e$	Bandwidth capacity of link $e$
$p_e^E$	Probability of failure of link $e$
$R_a$	Required total availability of application $a$ : lower bound on the probability that at least one of the duplicates for $a$ is available
$\delta$	Maximum allowed number of duplicates

nodes have a known probability of failure,  $p^N$  and  $p^E$  respectively. Failures are considered to be independent.

*3.3.0.3 The VAR Protection Method.* Availability not only depends on failure in the SN, but also on how the application is placed. Non-redundant application placement assigns each service and VL at most once, while its redundant counterpart can place those virtual resources more than once. The survivability method presented in this work, referred to as VAR, guarantees a minimum availability by application level replication, while minimizing the overhead imposed by allocation of those additional resources. VAR uses a static failure model, i.e. availability only depends on the current state of the network. Additionally, it is assumed that upon failure, switching between multiple application instances takes place without any delay. These separate application instances will be referred to as duplicates. Immediate switchover yields a good approximation, when the duration of switchover is small compared to the uptime of individual components. A small switchover time is feasible, given that each backup service is preloaded in memory, and CPU and bandwidth resources have been preallocated. Furthermore, immediate switchover allows condensation of the exact failure dynamics of each component, into its expected availability value, as long as the individual components fail independently (a more limiting assumption).





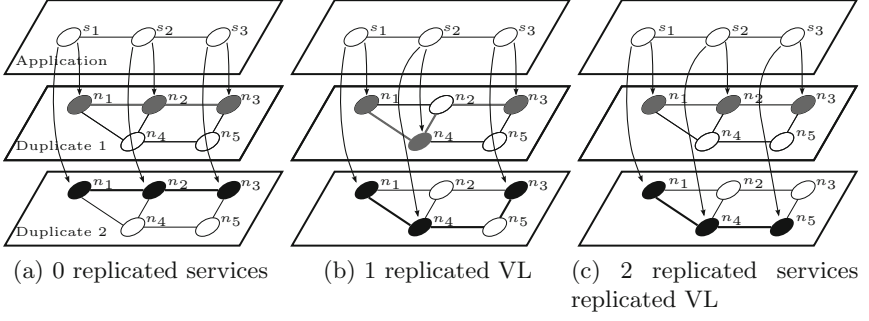
**Fig. 8.** Overview of this work: services  $\{\omega, \gamma, \beta\}$ , composing applications  $\{I\}$ , are placed on a substrate network where node  $\{p^N\}$  and link failure  $\{p^E\}$  is modeled. By increasing the redundancy  $\delta$ , a minimum availability  $R$  can be guaranteed.

**Table 5.** An overview of resource sharing amongst identical services and VLs.

	Sharing of resources		
	CPU	Memory	Bandwidth
Within application	Yes	Yes	Yes
Amongst applications	No	Yes	No

In the VAR model, an application is available if at least one of its duplicates is on-line. A duplicate is on-line if none of the PMs and Physical Links (PLs), that contribute its placement, fail. Duplicates of the same application can share physical components. An advantage of this reuse is that a fine-grained tradeoff can be made between increased availability, and decreased resource consumption. An overview of resources' reuse is shown in Table 5. In Fig. 9 three possible placement configurations using two duplicates are shown for one application. In Fig. 9a both duplicates are identical, and no redundancy is introduced. The nodal resource consumption is minimal, as CPU and memory for  $s_1$ ,  $s_2$ , and  $s_3$  are provisioned only once. Additionally, the total bandwidth required for  $(s_1, s_2)$ , and  $(s_2, s_3)$  is only provisioned once. The bandwidth consumption of this configuration might not be minimal, if consolidation of two or three services onto one PM is possible. This placement configuration does not provide any fault-tolerance, as failure of either  $n_1$ ,  $n_2$  or  $n_3$ , or  $(n_1, n_2)$ ,  $(n_2, n_3)$  results in downtime.

When more than one duplicate is placed and the resulting arrangements of VLs and services differ, then the placement is said to introduce redundancy. However, this increased redundancy results in a higher resource consumption. In Fig. 9b the application survives a singular failure of either  $(n_4, n_2)$ ,  $(n_2, n_3)$ ,  $(n_4, n_5)$ , or  $(n_5, n_3)$ . The placement configuration depicted in Fig. 9c survives all singular failures in the SN, except for a failure of  $n_1$ .



**Fig. 9.** Illustration of the VAR protection method.

**Formal Problem Description.** The algorithms presented in this work are based on the optimisation model proposed in [39]. In this section we briefly describe the model but refer to [39] for a more elaborate discussion. Our model consists of two main blocks: the cloud-environment and the set of applications. To model the problem we define the following constraints. We refer to [39] for the mathematical representation.

- The total amount of duplicates for each application is limited by  $\delta$ .
- An application  $a$  is placed correctly if and only if at least one duplicate of  $a$  is placed.
- A service is correctly placed if there is enough CPU and memory available in all PMs.
- A service will only be placed on a PM if and only if it is used by at least one duplicate.
- The total bandwidth of a PL cannot be higher than the aggregate bandwidth of the VLs that use the PL.
- A VL can use a PL if and only if the PL has sufficient remaining bandwidth.
- An application is only placed if the availability of the application can be guaranteed.

If a service is placed on the same PM, for multiple duplicates or for multiple applications, or the same VL is placed on a PL, they can reuse resources (see Table 5). Therefore, if service  $s$  is placed twice on PM  $n$  for the same application then there is no need to allocate CPU and memory twice. Only if service  $s$  is placed for a different application additional CPU resources must be allocated.

The problem we solve is to maximise the number of accepted applications.

**Results.** For a description of the proposed heuristics, and an extensive performance analysis, featuring multiple application types, SN types and scalability study we refer the interested reader to [40].

In reliable cloud environments (or equivalently, under low availability requirements) it is often acceptable to place each VN only once, and not bother about availability [27]. However, when the frequency of failures is higher (or if availability requirements increase), then one of the following measures should be

taken. First, one can improve the availability by placing additional backups, which fail independently of one another. However, this approach works best in homogeneous cloud environments, where one can use the same number of backup VN embeddings, regardless of the exact placement configuration. In heterogeneous environments a fixed redundancy level for each application either results in wasted SN resources, or a reduced placement ratio. In the context of cloud federation, the reliability of the links interconnecting the different cloud entities can be highly heterogeneous (leased lines, or best-effort public internet). Therefore, to further improve revenue, cloud federation should take these failure characteristics into consideration, and estimate the required replication level.

### 3.4 Level 2: Service Composition and Orchestration

Service composition and orchestration have become the predominant paradigms that enable businesses to combine and integrate services offered by third parties. For the commercial viability of composite services, it is crucial that they are offered at sharp price-quality ratios. A complicating factor is that many attractive third-party services often show highly variable service quality. This raises the need for mechanisms that promptly adapt the composition to changes in the quality delivered by third party services. In this section, we discuss a real-time QoS control mechanism that dynamically optimizes service composition in real time by learning and adapting to changes in third party service response time behaviors. Our approach combines the power of learning and adaptation with the power of dynamic programming. The results show that real-time service recompositions lead to dramatic savings of cost, while meeting the service quality requirements of the end-users.

#### 3.4.1 Background and Motivation

In the competitive market of information and communication services, it is crucial for service providers to be able to offer services at competitive price/quality ratios. Succeeding to do so will attract customers and generate business, while failing to do so will inevitably lead to customer dissatisfaction, churn and loss of business. A complicating factor in controlling quality-of-service (QoS) in service oriented architectures is that the ownership of the services in the composition (sub-services) is decentralized: a composite service makes use of sub-services offered by third parties, each with their own business incentives. As a consequence, the QoS experienced by the (paying) end user of a composite service depends heavily on the QoS levels realized by the individual sub-services running on different underlying platforms with different performance characteristics: a badly performing sub-service may strongly degrade the end-to-end QoS of a composite service. In practice, service providers tend to outsource responsibilities by negotiating Service Level Agreements (SLAs) with third parties. However, negotiating multiple SLAs in itself is not sufficient to guarantee end-to-end QoS levels as SLAs in practice often give probabilistic QoS guarantees and SLA violations can still occur. Moreover probabilistic QoS guarantees do not necessarily capture time-dependent behavior e.g. short term service degradations.

Therefore, the negotiation of SLAs needs to be supplemented with *run-time QoS-control* capabilities that give providers of composite services the capability to properly respond to short-term QoS degradations (real-time composite service adaptation). Motivated by this, in this section we propose an approach that adapts to (temporary) third party QoS degradations by tracking the response time behavior of these third party services.

### 3.4.2 Literature and Related Work

The problem of QoS-aware optimal composition and orchestration of composite services has been well-studied (see e.g. [41, 42]). The main problem addressed in these papers is how to select one concrete service per abstract service for a given workflow, in such a way that the QoS of the composite service (as expressed by the respective SLA) is guaranteed, while optimizing some cost function. Once established, this composition would remain unchanged the entire life-cycle of the composite web service. In reality, SLA violations occur relatively often, leading to providers' losses and customer dissatisfaction. To overcome this issue, it is suggested in [43–45] that, based on observations of the actually realised performance, re-composition of the service may be triggered. During the re-composition phase, new concrete service(s) may be chosen for the given workflow. Once re-composition phase is over, the (new) composition is used as long as there are no further SLA violations. In particular, the authors of [43–45] describe *when* to trigger such (re-composition) event, and *which adaptation actions* may be used to improve overall performance.

A number of solutions have been proposed for the problem of *dynamic, run-time* QoS-aware service selection and composition within SOA [46–49]. These (proactive) solutions aim to adapt the service composition dynamically at run-time. However, these papers do not consider the stochastic nature of response time, but its expected value. Or they do not consider the cost structure, revenue and penalty model as given in this paper.

In the next section, we extend the approach presented in [48] such that we can learn an exploit response-time distributions on the fly. The use of classical reinforcement-learning techniques would be a straight forward approach. However, our model has a special structure that complicates the use of the classical Temporal Difference learning (TD) learning approaches. The solution of our DP formulation searches the stochastic shortest path in a stochastic activity network [50]. This DP can be characterized as a hierarchical DP [51, 52]. Therefore classical Reinforcement Learning (RL) is not suitable and hierarchical RL has to be applied [52]. Also changes in response-time behavior are likely to occur which complicates the problem even more. Both the problem structure and volatility are challenging areas of research in RL. Typically RL techniques solve complex learning and optimization problems by using a simulator. This involves a Q value that assigns utility to state-action combinations. Most algorithms run off-line as a simulator is used for optimization. RL has also been widely used in on-line applications. In such applications, information becomes available gradually with

time. Most RL approaches are based on environments that do not vary over time. We refer to [51] for a good survey on reinforcement learning techniques.

In our approach we tackle both the hierarchical structure, and time varying behavior challenges. To this end we are using empirical distributions and updating the lookup table if significant changes occur. As we are considering a sequence of tasks, the number of possible response time realizations combinations explodes. By discretizing the empirical distribution over fixed intervals we overcome this issue.

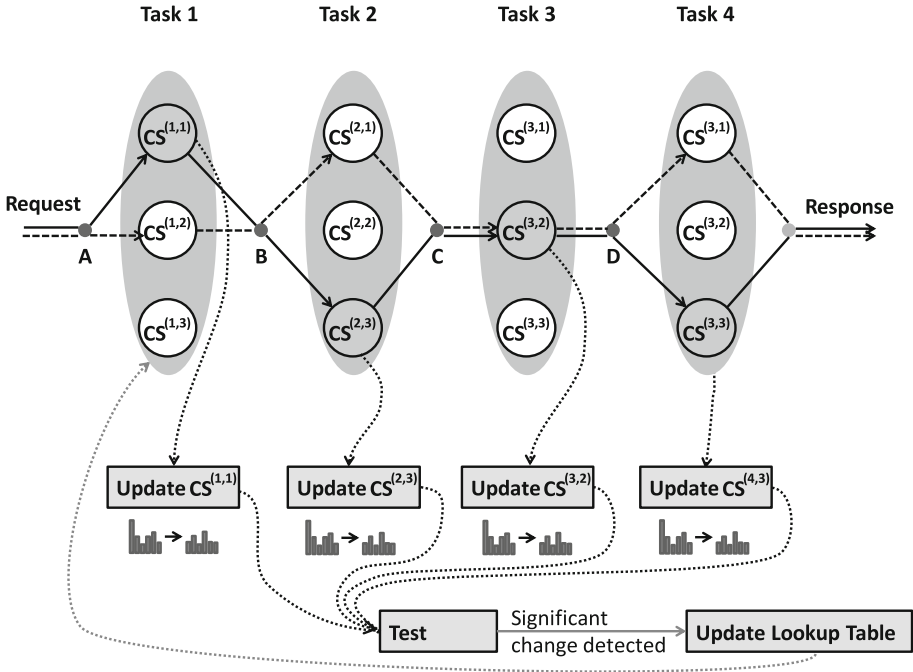
### 3.4.3 Composition and Orchestration Model

We consider a composite service that comprises a sequential workflow consisting of  $N$  tasks identified by  $T_1, \dots, T_N$ . The tasks are executed one-by-one in the sense that each consecutive task has to wait for the previous task to finish. Our solution is applicable to any workflow that could be aggregated and mapped into a sequential one. Basic rules for aggregation of non-sequential workflows into sequential workflows have been illustrated in, e.g. [48, 50, 53]. However, the aggregation leads to coarser control, since decisions could not be taken for a single service within the aggregated workflow, but rather for the aggregated workflow patterns themselves.

The workflow is based on an unambiguous functionality description of a service (“abstract service”), and several functionally identical alternatives (“concrete services”) may exist that match such a description [54]. Each task has an abstract service description or interface which can be implemented by external service providers.

The workflow in Fig. 10 consists of four abstract tasks, and each task maps to three concrete services (alternatives), which are deployed by (independent) third-party service providers. For each task  $T_i$  there are  $M_i$  concrete service providers  $CS^{(i,1)}, \dots, CS^{(i,M_i)}$  available that implement the functionality corresponding to task  $T_i$ . For each request processed by  $CS^{(i,j)}$  cost  $c^{(i,j)}$  has to be paid. Furthermore there is an end-to-end response-time deadline  $\delta_p$ . If a request is processed within  $\delta_p$  a reward of  $R$  is received. However, for all requests that are not processed within  $\delta_p$  a penalty  $V$  had to be paid. After the execution of a single task within the workflow, the orchestrator decides on the next concrete service to be executed, and composite service provider pays to the third party provider per single invocation. The decision points for given tasks are illustrated at Fig. 10 by A, B, C and D. The decision taken is based on (1) execution costs, and (2) the remaining time to meet the end-to-end deadline. The response time of each concrete service provider  $CS^{(i,j)}$  is represented by the random variable  $D^{(i,j)}$ . After each decision the observed response time is used for updating the response time distribution information of the selected service. Upon each lookup table update the corresponding distribution information is stored as reference distribution. After each response the reference distribution is compared against the current up-to date response time distribution information.

In our approach response-time realizations are used for learning an updating the response-time distributions. The currently known response-time distribution



**Fig. 10.** Orchestrated composite web service depicted by a sequential workflow. Dynamic run-time service composition is based on a lookup table. Decisions are taken at points A–D. For every used concrete service the response-time distribution is updated with the new realization. In this example a significant change is detected. As a result for the next request concrete service 2 is selected at task 1.

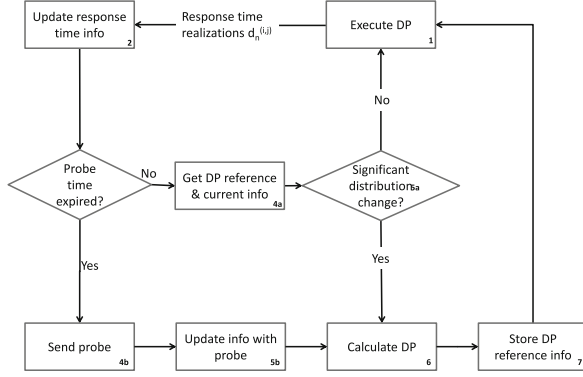
is compared against the response-time distribution that was used for the last policy update. Using well known statistical tests we are able to identify if an significant change occurred and the policy has to be recalculated. Our approach is based on fully dynamic, run-time service selection and composition, taking into account the response-time commitments from service providers and information from response-time realizations. The main goal of this run-time service selection and composition is profit maximization for the composite service provider and ability to adapt to changes in response-time behavior of third party services.

By tracking response times the actual response-time behavior can be captured in empirical distributions. In [48] we apply a dynamic programming (DP) approach in order to derive a service-selection policy based on response-time realizations. With this approach it is assumed that the response-time distributions are known or derived from historical data. This results in a so called lookup table which determines what third party alternative should be used based on actual response-time realizations.

### 3.4.4 Real Time QoS Control

In this section we explain our real-time QoS control approach. The main goal of this approach is profit maximization for the composite service provider, and ability to adapt to changes in response-time behavior of third party services. We realize this by monitoring/tracking the observed response-time realizations. The currently known empirical response-time distribution is compared against the response-time distribution that was used for the last policy update. Using well known statistical tests we are able to identify if a significant change occurred and the policy has to be recalculated. Our approach is based on fully dynamic, run-time service selection and composition, taking into account the response-time commitments from service providers and information from response-time realizations. We illustrate our approach using Fig. 11. The execution starts with an initial lookup table at step (1). This could be derived from initial measurements on the system. After each execution of a request in step (2) the empirical distribution is updated at step (3). A DP based lookup table could leave out unattractive concrete service providers. In that case we do not receive any information about these providers. These could become attractive if the response-time behavior changes. Therefore in step (4), if a provider is not visited for a certain time, a probe request will be sent at step (5b) and the corresponding empirical distribution will be updated at step (6a). After each calculation of the lookup table, the current set of empirical distributions will be stored. These are the empirical distributions that were used in the lookup table calculation and form a reference response-time distribution. Calculating the lookup table for every new sample is expensive and undesired. Therefore we propose a strategy where the lookup table will be updated if a significant change in one of the services is detected. For this purpose the reference distribution is used for detection of response-time distribution changes. In step (5a) and step (6a) the reference distribution and current distribution are retrieved and a statistical test is applied for detecting change in the response-time distribution. If no change is detected then the lookup table remains unchanged. Otherwise the lookup table is updated using the DP. After a probe update in step (5b) and step (6b) we immediately proceed to updating the lookup table as probes are sent less frequently. In step (7) and step (8) the lookup table is updated with the current empirical distributions and these distributions are stored as new reference distribution. By using empirical distributions we are directly able to learn and adapt to (temporarily) changes in behavior of third party services.

Using a lookup table based on empirical distributions could result in the situation that certain alternatives are never invoked. When other alternatives break down this alternative could become attractive. In order to deal with this issue we use probes. A probe is a dummy request that will provide new information about the response time for that alternative. As we only receive updates from alternatives which are selected by the dynamic program, we have to keep track of how long ago a certain alternative has been used. For this purpose to each concrete service provider a probe timer  $U^{(i,j)}$  is assigned with corresponding probe time-out  $t_p^{(i,j)}$ . If a provider is not visited in  $t_p^{(i,j)}$  requests ( $U^{(i,j)} > t_p^{(i,j)}$ )



**Fig. 11.** Real-time QoS control approach.

then the probe timer has expired and a probe will be collected incurring probe cost  $c_p^{(k,j)}$ . If for example, in Fig. 10, the second alternative of the third task has not been used in the last ten requests, the probe timer for alternative two has value  $U^{(3,2)} = 10$ . After a probe we immediately update the corresponding distribution. No test is applied here as probes are collected less frequent compared to processed requests.

In order to evaluate the proposed QoS control methods we have performed extensive evaluation testing in an experimental setting. The results show that real-time service re-compositions indeed lead to dramatic savings in cost, while still meeting QoS requirements of the end users. The reader is referred to [55] for the details.

### 3.5 Level 1: Resource Management in Virtualized Infrastructure

Level 1 deals with the dependencies of different physical resources, such as Central Processing Unit (CPU) time, Random Access Memory (RAM), disk I/O, and network access, and their effect on the performance that users perceive. These dependencies can be described by functions that map resource combinations, i.e. resource vectors, to scalars that describe the performance that is achieved with these resources. Therefore, such *utility functions* describe how the combination of different resources influences the performance users perceive [56]. Accordingly, utility functions (a) indicate in which ratios resources have to be allocated, in order to maximize user satisfaction and efficiency, (b) are determined by technical factors, and (c) are investigated in this section.

#### 3.5.1 Methodology

In order to get an idea about the nature of utility functions that VMs have during runtime, dependencies between physical resources, when utilized by VMs, and effects on VM performance are investigated as follows. Different workloads are



executed on a VM with a changing number of Virtual CPUs (VCPU) and Virtual RAM (VRAM) (this influences how many physical resources the VM can access) and varying load levels of the host system (this simulates contention among VMs and also influences how many physical resources the VM can access).

A machine with a 2.5 Gigahertz (GHz) AMD Opteron 6180 SE processor with 24 cores and 6 and 10 MB of level 2 and 3 cache, respectively, and 64 GB of ECC DDR3 RAM with 1333 Mhz is used as host system. VM and host have a x86-64 architecture and run Ubuntu 14.04.2 LTS, Trusty Tahr, which was the latest Ubuntu release, when the experiments were conducted.

**3.5.1.1 Measurement Method.** Resource consumption of VMs is measured by monitoring the VM's (qemu [57]) process. In particular, the VM's CPU time and permanent storage I/O utilization is measured with *psutil* (a python system and process utilities library) and the VM's RAM utilization by the VM's proportional set size, which is determined with the tool *smem* [58].

**3.5.1.2 Workloads.** Workloads are simulated by the following benchmarks of the Phoronix test suite [59].

**Apache.** This workload measures how many requests the Apache server can sustain concurrently.

**Aio-stress.** This benchmark assesses the speed of permanent storage I/O (hard disk or solid state drive). In a virtualized environment permanent storage can be cached in the host system's RAM. Therefore, this test not necessarily results in access to the host system's permanent storage.

**7zip.** This benchmark uses 7zip's integrated benchmark feature to measure the system's compression speed.

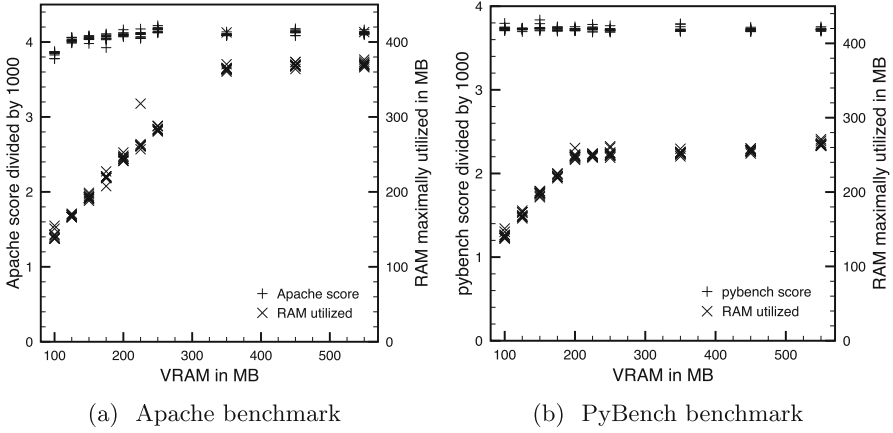
**PyBench.** This benchmark measures the execution time of Python functions such as BuiltinFunctionCalls and NestedForLoops. Contrary to all other benchmarks, here a lower score is better.

### 3.5.2 Results

This section presents selected results from [60] that were achieved with the setup described above.

**3.5.2.1 RAM.** Figure 12 shows the scores a VM achieves on the Apache and PyBench benchmark and the RAM it utilizes depending on the VRAM. For each VRAM configuration 10 measurements are conducted.

Figure 12a shows that when the VM executes Apache, it never utilizes more than 390 MB of RAM. In particular, for a VM with 100 to 350 MB of VRAM the amount of RAM that is maximally utilized continuously increases but does not further increase, when more than 350 MB of VRAM are added. Therefore, Fig. 12a shows that a VM with less than 350 MB of VRAM utilizes all RAM that is available, which seems to imply, that this amount of RAM is critical for performance. However, Fig. 12a also depicts that the Apache score only increases for up to 250 MB of VRAM and that this increase is marginal compared to the



**Fig. 12.** Benchmark scores and RAM utilization depending on a VM's VRAM

increase of RAM that is utilized. Therefore, the dependency between VRAM and utilized RAM is much stronger than the dependency between VRAM/used RAM and Apache score. In particular, while the RAM utilization more than doubles, the Apache scores vary by less than 10%. This is particularly interesting, because this configuration range includes 100 MB of VRAM which constrains the VM's RAM utilization to less than half of what the VM alone (without executing any workload) would utilize.

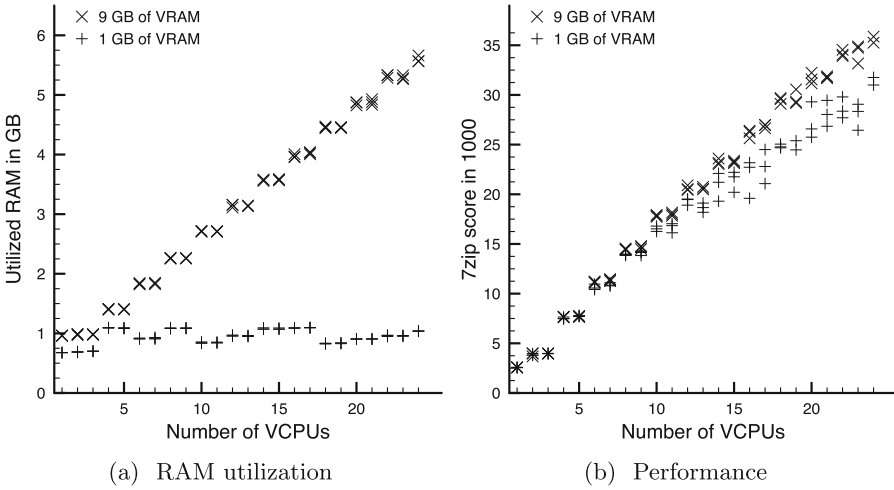
Figure 12b shows that when the VM executes PyBench, the VM process utilizes 270 MB of RAM at most. Although the VM is constraint in its RAM utilization, when it has less than 250 MB of VRAM, there is no correlation between the achieved PyBench score and the VM's VRAM, as the PyBench score does not increase.

Therefore, Fig. 12 shows that RAM, which is actively utilized by a VM (be it on startup or when executing an application), not necessarily impacts the VM's performance. In particular, even if the RAM utilized by a VM varies from 100 MB to 350 MB, the VM's Apache score, i.e., its ability to sustain concurrent server requests, only changed by 10%. For PyBench the score was entirely independent of the available RAM. This is particularly interesting, because not even a VM with 100 MB of VRAM showed decreased performance, while this is the minimum amount of RAM that avoids a kernel panic and even a VM that not executes any workload utilizes more, if possible.

**3.5.2.2 VCPUs and Maximal RAM Utilization.** The 7zip benchmark reveals an interesting dependency of VCPUs and RAM utilization (cf. Fig. 13). As Fig. 13a shows, for one to three VCPUs a VM executing the 7zip benchmark utilizes 1 GB of RAM and for every two additional cores the RAM utilization increases by 400 MB (the VM had 9 GB of VRAM).

The distinct pattern in which RAM is utilized gives reason to believe, that it is essential for performance. Therefore, Fig. 13b compares the 7zip scores

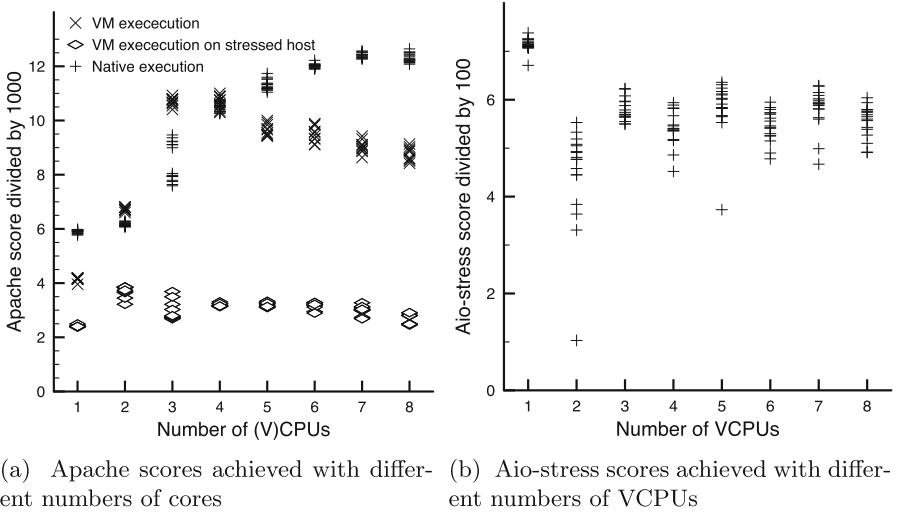
achieved by VMs with 1 and 9 GB of VRAM. As Fig. 13a shows, the more VCPUs a VM has, the more it will be constrained by only having 1 GB of VRAM, while 9 GB of VRAM not even constrain a VM with 24 VCPUs. In line with this observation, Fig. 13b shows that the difference between the 7zip scores achieved by VMs with 1 and 9 GB of VRAM grows with the number of VCPUs. However, the score difference is rather moderate compared to the large difference in terms of RAM utilization. In particular, a VM with 24 VCPUs utilizes more than 5 GB of RAM, if available. This is five times as much, as a VM with 1 GB of VRAM utilizes. However, the 7zip scores achieved by these VMs only differ by 15%.



**Fig. 13.** RAM utilization and performance, depending on the number of VCPUs and amount of VRAM, of a VM executing the 7zip benchmark

**3.5.2.3 Multi Core Penalty.** Figure 14a plots the Apache scores achieved by a VM with 1 to 9 VCPUs, whereat 16 measurements per configuration were conducted. The figure shows that the best performance is achieved, when the VM has three or four VCPUs, while additional VCPUs linearly decrease the Apache score. As the figure depicts, up to three VCPUs significantly increase performance and four VCPUs perform equally well. However, adding additional VCPUs continuously decreases performance. This effect, which is termed *multi-core-penalty* occurred, independent of whether VCPUs were pinned to physical CPUs. Figure 14a also demonstrates that, while three VCPUs perform best for an unstressed host, two VCPUs perform best, when the host is stressed. Furthermore, the multi-core-penalty does not occur, when the benchmark is executed natively, i.e., directly on the host and not inside a VM. This shows that the it is caused by the virtualization layer. Despite the decrease of the Apache score with the number of VCPUs, the VM's utilization of CPU time increases with the number of VCPUs. For example, for the Apache benchmark it was found that for 9 VCPUs the

utilized CPU time is roughly twice as high as the CPU time utilized by one to three VCPUs (although the Apache score was significantly lower for 9 VCPUs).



**Fig. 14.** Two example of the multi-core-penalty

Figure 14b shows that the multi-core penalty also occurs for the aio-stress benchmark, where a VM with one VCPU constantly achieves a higher aio-stress score than any VM with more VCPUs. In particular, the aio-stress score of a VM with only one VCPU is on average a 30% higher than the aio-stress score of VMs with more VCPUs. However, unlike the Apache benchmark, the aio-stress score does not decrease with the number of VCPUs.

### 3.5.3 New Findings

Most work on data center resource allocation assumes that resources such as CPU and RAM are required in static or at least well defined ratios and that the resulting performance is clearly defined. The results of this section do not confirm these idealistic assumptions.

Section 3.5.2 did not find any significant effect of a VRAM on VM performance. Notably, even for workloads that seem to be RAM critical, as they utilize RAM in distinct patterns, or workloads running on VMs with just enough VRAM to avoid a kernel panic during boot, no significant effect was found. Even if a lack of RAM impedes performance, the impediment is minor compared to the amount of RAM that is missing (cf. Sect. 3.5.2). In contrast, a lack of RAM bandwidth significantly effects performance [61] but is rarely considered, when investigating data center fairness. Section 3.5.2 showed that the amount of RAM that is utilized by a VM may depend on the number of VCPUs. Section 3.5.2 presents the most counter-intuitive finding, which is that, when multi-core benchmarks

are executed inside a VM, the performance often decreases, when more VCPUs are added to the VM.

This section showed that it is a complex task to determine a class of utility functions that properly models the allocation of a node's PRs to VMs. However, a realistic class of utility functions would greatly aid cloud resource allocation, as it would allow to theoretically determine allocations that are practically more efficient. Therefore, positive results on this topic would also greatly aid the performance of cloud federations, as it would also allow to execute tasks in the cloud of a federation, that performs best for this task. Nonetheless, no work exists on this topic. This lack of work is caused by the topic's complexity. For example, resource dependencies vary over time, and depend on the workload that is executed inside a VM and the host's architecture. Also, the performance of a VM is determined by a combination of resources as diverse as CPU time, RAM, disk I/O, network access, CPU cache capacity, and memory bandwidth, where substitutabilities may or may not apply.

## 4 Cloud Federation for IoT

### 4.1 State-of-the-Art in IoT Cloud Research

The integration of IoT and clouds has been envisioned by Botta et al. [62] by summarizing their main properties, features, underlying technologies, and open issues. A solution for merging IoT and clouds is proposed by Nastic et al. [63]. They argued that system designers and operations managers faced numerous challenges to realize IoT cloud systems in practice, due to the complexity and diversity of their requirements in terms of IoT resources consumption, customization and runtime governance. They also proposed a novel approach for IoT cloud integration that encapsulated fine-grained IoT resources and capabilities in well-defined APIs in order to provide a unified view on accessing, configuring and operating IoT cloud systems, and demonstrated their framework for managing electric fleet vehicles.

Atzori et al. [64,65] examined IoT systems in a survey. They identified many application scenarios, and classified them into five application domains: transportation and logistics, healthcare, smart environments (home, office, plant), personal, social and futuristic domains. They described these domains in detail, and defined open issues and challenges for all of them. Concerning privacy, they stated that much sensitive information about a person can be collected without their awareness, and its control is impossible with current techniques.

Escribano [66] discussed the first opinion [67] of the Article 29 Data Protection Working Party (WP29) on IoT. According to these reports four categories can be differentiated: the first one is wearable computing, which means the application of everyday objects and clothes, such as watches and glasses, in which sensors were included to extend their functionalities. The second category is called the 'quantified self things', where things can also be carried by individuals to record information about themselves. With such things we can examine physical activities, track movements, and measure weight, pulse or other health indicators. The third one is home automation, which covers applications using

devices placed in offices or homes such as connected light bulbs, thermostats, or smoke alarms that can be controlled remotely over the Internet. They also mention smart cities as the fourth category, but they do not define them explicitly. They argue that sharing and combining data through clouds will increase locations and jurisdictions, where personal data resides. Therefore it is crucial to identify and realize which stakeholder is responsible for data protection. WP29 named many challenges concerning privacy and data protection, like lack of user control, intrusive user profiling and communication and infrastructure related security risks.

IoT application areas and scenarios have already been categorized, such as by Want et al. [68], who set up three categories: Composable systems, which are ad-hoc systems that can be built from a variety of nearby things by making connections among these possibly different kinds of devices. Since these devices can discover each other over local wireless connections, they can be combined to provide higher-level capabilities. Smart cities providing modern utilities could be managed more efficiently with IoT technologies. As an example traffic-light systems can be made capable of sensing the location and density of cars in the area, and optimizing red and green lights to offer the best possible service for drivers and pedestrians. Finally, resource conservation scenarios, where major improvements can be made in the monitoring and optimization of resources such as electricity and water.

## 4.2 MobIoTSim for Simulating IoT Devices

Cloud Federation can help IoT systems by providing more flexibility and scalability. Higher level decisions can be made on where to place a gateway service to receive IoT device messages, e.g. in order to optimize resource usage costs and energy utilization. Such complex IoT cloud systems can hardly be investigated in real world, therefore we need to turn to simulations.

The main purpose of MobIoTSim [69], our proposed mobile IoT device simulator, is to help cloud application developers to learn IoT device handling without buying real sensors, and to test and demonstrate IoT applications utilizing multiple devices. The structure of the application lets users create IoT environment simulations in a fast and efficient way that allows for customization.

MobIoTSim can simulate one or more IoT devices, and it is implemented as a mobile application for the Android platform. Sensor data generation of the simulated devices are random generated values in the range given by the user, or replayed data from trace files. The data sending frequency can also be specified for every device. The application uses the MQTT protocol to send data with the use of the Eclipse Paho opensource library. The data is represented in a structured JSON object compatible with the IBM IoT Foundation message format [70].

The basic usage of the simulator is to (i) connect to a cloud gateway, where the data is to be sent, (ii) create and configure the devices to be simulated and (iii) start the (data generation of the) required devices. These main steps are represented by three main parts of the application: the *Cloud settings*, the *Devices* and the *Device settings* screens. In the *Cloud settings* screen, the user

can set the required information about the targeted cloud, where the data will be received and processed. Currently there are two types of clouds supported: IBM Bluemix and MS Azure. For the IBM cloud we have two options: the Bluemix quickstart and the standard Bluemix IoT service. The Bluemix quickstart is a public demo application, it can visualise the data from a selected device. For a fast and easy setup (i.e. to try out the simulator) this type is recommended. The standard Bluemix IoT service type can be used if the user has a registered account for the Bluemix platform, and already created an IoT service. This IoT service can be used to handle devices, which have been registered before. The main part of the IoT service is an MQTT broker, this is the destination of the device messages, and it forwards them to the cloud applications. Such cloud applications can process the data, react to it or just perform some visualisation. The required configuration parameters for the standard Bluemix IoT service in MobIoTSim are: the Organization ID, which is the identifier of the IoT service of the user in Bluemix, and an authentication key, so that the user does not have to register the devices on the Bluemix web interface, and the command and event IDs, which are customizable parts of the used MQTT topics to send messages from the devices to the cloud and vice versa. MobIoTSim can register the created devices with these parameters automatically, by using the REST interface of Bluemix.

The *Devices* screen lists the created devices, where every row is a device or a device group. These devices can be started and stopped by the user at will, both together or separately for the selected ones. Some devices have the ability to display warnings and notifications sent back by a gateway. In this screen we can also create new devices or device groups. There are some pre-defined device templates, which can be selected for creation. These device templates help to create often used devices, such as a temperature sensor, humidity sensor or a thermostat. If the user selects a template for the base of the device, the message content and frequency will be set to some predefined values. The *Thermostat* template has a temperature parameter, it turns on by reaching a pre-defined low-level value and turns off at the high-level value. The On/Off state of the device is displayed all the time. It is possible to select the *Custom* template to configure a device in detail.

The new device creation and the editing of an existing one are made in the *Device settings* screen. The user can add more parameters to a device and can customize it with its own range. The range will be used to generate random values for the parameters. A device group is a group of devices with the same base template and they can be started and stopped together. If a device wants to send data to the Bluemix IoT service, it has to be registered beforehand. The registered devices have device IDs and tokens for authentication. The MobIoT-Sim application handles the device registration in the cloud with REST calls, so the user does not have to register the devices manually on the graphical web interface. There is an option to save the devices to a file and load them back to the application later. The device type attribute can be used to group devices. The simulation itself can also be saved, so the randomly generated data can be

replayed later many times. Even trace files from real world applications can be played from other sources, i.e. saved samples from the OpenWeatherMap public weather data provider [71]. The OpenWeatherMap monitors many cities and stores many parameters for them, including temperature, humidity, air pressure and wind speed. Using this trace loader feature, the simulation becomes closer to a real life scenario. In some cases, the user may want to send data to not just one but more cloud gateways at the same time. This is also possible by changing the organization ID attribute of a device to one of the already saved ones in the cloud settings.

We modified the Bluemix visualisation application to create a new private gateway to handle more than one device at the same time. In this way we can see the data from all devices in a real time chart. The node.js application subscribes to all device topics with the MQTT protocol, and waits for the data. In this revised gateway we use paging to overcome device management limitations (25 devices at a time). In order to enhance and better visualize many device data at the same time, we introduced device grouping for the chart generation.

To summarize, MobIoTSim together with the proposed gateways provide a novel solution to enable the simulation and experimentation of IoT cloud systems. Our future work will address extensions for additional thing and sensor templates, and will provide cases for scalability investigations involving multiple cloud gateways.

## 5 Summary

In this chapter we have reported activities of the COST IC1304 ACROSS European Project corresponding to traffic management for Cloud Federation. In particular, we have provided survey of discussed CF architectures and corresponding standardization activities, we have proposed comprehensive multi-level model for traffic management for CF together with proposed solutions for each level. The effectiveness of these solutions were verified by simulation and analytical methods. The proposed levels are: Level 5 - Strategies for building CF, Level 4 - Network for CF, Level 3 - Service specification and provision, Level 2 - Service composition and orchestration, Level 1 - Task service in cloud resources. Finally, we have presented specialized simulator for testing CF solution in IoT environment.

## References

1. Schubert, L., Jeffery, K.: Advances in Clouds - Research in Future Cloud Computing, Report from the Cloud Computing Expert Working Group Meeting. Cordis (Online), BE: European Commission (2012). <http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf>
2. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. *Softw. Pract. Exper.* (2012). <https://doi.org/10.1002/spe.2168>



3. Celesti, A., Tusa, F., Villari, M., Puliafito, A.: How to enhance cloud architectures to enable cross-federation. In: *Proceedings of the 3rd International Conference on Cloud Computing (CLOUD 2010)*, Miami, Florida, USA, pp. 337–345. IEEE (2010)
4. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., Morrow, M.: Blueprint for the intercloud - protocols and formats for cloud computing interoperability. In: *Proceedings of the Fourth International Conference on Internet and Web Applications and Services*, pp. 328–336 (2009)
5. Marosi, A.C., Kecskemeti, G., Kertesz, A., Kacsuk, P.: FCM: an architecture for integrating IaaS cloud systems. In: *Proceedings of the Second International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2011)*, IARIA, pp. 7–12, Rome, Italy (2011)
6. International Telecommunication Union (ITU-T): *Framework of Inter-Cloud Computing* (2014)
7. Internet Engineering Task Force (IETF): *Working group on Content Delivery Network Interconnection (CDNI)* (2011)
8. National Institute of Standards and Technology [NIST]: U.S. Dept. of Commerce, *NIST Cloud Computing Standards Roadmap*, Spec. Publ. 500–291 (2013)
9. Institute of electrical and electronics engineering (IEEE): *Inter-cloud working group, Standard for Intercloud Interoperability and Federation (SIIF)* (2017)
10. Darzanos, G., Koutsopoulos, I., Stamoulis, G.D.: Economics models and policies for cloud federations. In: *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, Vienna, pp. 485–493 (2016). <https://doi.org/10.1109/IFIPNetworking.2016.7497246>
11. Samaan, N.: A novel economic sharing model in a federation of selfish cloud providers. *IEEE Trans. Parallel Distrib. Syst.* **25**(1), 12–21 (2014). <https://doi.org/10.1109/TPDS.2013.23>
12. Kleinrock, L.: *Queueing Systems Volume 1: Theory*, p. 103. Wiley, Hoboken (1975). ISBN 0471491101
13. Carlini, E., Coppola, M., Dazzi, P., Ricci, L., Righetti, G.: Cloud federations in contrail. In: Alexander, M., et al. (eds.) *Euro-Par 2011. LNCS*, vol. 7155, pp. 159–168. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29737-3\\_19](https://doi.org/10.1007/978-3-642-29737-3_19)
14. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A.: B4: experience with a globally-deployed software defined WAN. In: *ACM SIGCOMM 2013 Conference*, New York, USA (2013)
15. Yen, J.Y.: Finding the K shortest loopless paths in a network. *Manag. Sci. JSTOR* **17**(11), 712–716 (1971). [www.jstor.org/stable/2629312](http://www.jstor.org/stable/2629312)
16. Aljazzar, H., Leue, S.: K\*: a heuristic search algorithm for finding the  $k$  shortest paths. *Artif. Intell.* **175**(18), 2129–2154 (2011). <https://doi.org/10.1016/j.artint.2011.07.003>. ISSN 0004–3702
17. Puleri, M., Sabella, R.: Cloud robotics: 5G paves the way for mass-market automation. In: *Charting the Future of Innovation*, 5th edn., vol. 93, Ericsson, Stockholm (2016)
18. Bonomi, F., Mito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, pp. 13–16. ACM (2012). <https://doi.org/10.1145/2342509.2342513>

19. Al-Muhtadi, J., Campbell, R., Kapadia, A., Mickunas, M.D., Yi, S.: Routing through the mist: privacy preserving communication in ubiquitous computing environments. In: Proceedings 22nd International Conference on Distributed Computing Systems, pp. 74–83 (2002). <https://doi.org/10.1109/ICDCS.2002.1022244>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1022244>
20. ISO/IEC-25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, Standard, International Organization for Standardization, Geneva, CH, March 2010
21. Spinnewyn, B., Latré, S.: Towards a fluid cloud: an extension of the cloud into the local network. In: Latré, S., Charalambides, M., François, J., Schmitt, C., Stiller, B. (eds.) AIMS 2015. LNCS, vol. 9122, pp. 61–65. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-20034-7\\_7](https://doi.org/10.1007/978-3-319-20034-7_7)
22. Camati, R., Calsavara, A., Lima Jr., L.: Solving the virtual machine placement problem as a multiple multidimensional Knapsack problem. In: ICN 2014, no. c, pp. 253–260 (2014). <https://www.thinkmind.org/download.php?articleid=icn.2014.11.10.30065>
23. Xu, J., Fortes, J.A.B.: Multi-objective virtual machine placement in virtualized data center environments. In: 2010 IEEE/ACM International Conference on \& International Conference on Cyber, Physical and Social Computing (CPSCom), GREENCOM-CPSCom 2010, IEEE Computer Society, Washington, DC, USA, pp. 179–188 (2010). <https://doi.org/10.1109/GreenCom-CPSCom.2010.137>
24. Ren, Y., Suzuki, J., Vasilakos, A., Omura, S., Oba, K.: Cielo: an evolutionary game theoretic framework for virtual machine placement in clouds. In: Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014, pp. 1–8 (2014). <https://doi.org/10.1109/FiCloud.2014.11>
25. Moens, H., Truyen, E., Walraven, S., Joosen, W., Dhoedt, B., De Turck, F.: Cost-effective feature placement of customizable multi-tenant applications in the cloud. *J. Netw. Syst. Manag.* **22**(4), 517–558 (2014). <https://doi.org/10.1007/s10922-013-9265-5>
26. Fischer, A., Botero, J.F., Beck, M.T., De Meer, H., Hesselbach, X.: Virtual network embedding: a survey. *IEEE Commun. Surv. Tutor.* **15**(4), 1888–1906 (2013). <https://doi.org/10.1109/SURV.2013.013013.00155>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6463372>
27. Moens, H., Hanssens, B., Dhoedt, B., De Turck, F.: Hierarchical network-aware placement of service oriented applications in clouds. In: IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World, pp. 1–8 (2014). <https://doi.org/10.1109/NOMS.2014.6838230>
28. Cheng, X., Su, S., Zhang, Z., Wang, H., Yang, F., Luo, Y., Wang, J.: Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Rev.* **41**(2), 38 (2011). <https://doi.org/10.1145/1971162.1971168>
29. Zhu, Y., Ammar, M.: Algorithms for assigning substrate network resources to virtual network components. In: Proceedings - IEEE INFOCOM, pp. 1–12 (2006). <https://doi.org/10.1109/INFOCOM.2006.322>
30. Ajtai, M., Alon, N., Bruck, J., Cypher, R., Ho, C., Naor, M., Szemerédi, E.: Fault tolerant graphs, perfect hash functions and disjoint paths. In: Proceedings, 33rd Annual Symposium on Foundations of Computer Science, pp. 693–702 (1992). <https://doi.org/10.1109/SFCS.1992.267781>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=267781>

31. Mihailescu, M., Sharify, S., Amza, C.: Optimized application placement for network congestion and failure resiliency in clouds. In: 2015 IEEE 4th International Conference on Cloud Networking, CloudNet 2015, pp. 7–13 (2015). <https://doi.org/10.1109/CloudNet.2015.7335272>
32. Csorba, M.J., Meling, H., Heegaard, P.E.: Ant system for service deployment in private and public clouds. In: Proceeding of the 2nd Workshop on Bio-inspired Algorithms for Distributed Systems - BADS 2010, p. 19. ACM (2010). <https://doi.org/10.1145/1809018.1809024>. <http://portal.acm.org/citation.cfm?doid=1809018.1809024>
33. Khan, M.M.A., Shahriar, N., Ahmed, R., Boutaba, R.: SiMPLE: survivability in multi-path link embedding. In: Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015, pp. 210–218 (2015). <https://doi.org/10.1109/CNSM.2015.7367361>
34. Chowdhury, S., Ahmed, R., Alamkhan, M.M., Shahriar, N., Boutaba, R., Mitra, J., Zeng, F.: Dedicated protection for survivable virtual network embedding. In: IEEE Transactions on Network and Service Management, p. 1 (2016). <https://doi.org/10.1109/TNSM.2016.2574239>. <http://ieeexplore.ieee.org/document/7480798/>
35. Jayasinghe, D., Pu, C., Eilam, T., Steinder, M., Whalley, I., Snible, E.: Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement. In: Proceedings - 2011 IEEE International Conference on Services Computing, SCC 2011, pp. 72–79. IEEE (2011). <https://doi.org/10.1109/SCC.2011.28>
36. Wang, W., Chen, H., Chen, X.: An availability-aware virtual machine placement approach for dynamic scaling of cloud applications. In: Proceedings - IEEE 9th International Conference on Ubiquitous Intelligence and Computing and IEEE 9th International Conference on Autonomic and Trusted Computing, UIC-ATC 2012, pp. 509–516 (2012). <https://doi.org/10.1109/UIC-ATC.2012.31>
37. Yeow, W.-L., Westphal, C., Kozat, U.: Designing and embedding reliable virtual infrastructures. In: Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures - VISA 2010, vol. 41(2), p. 33 (2010). [arXiv:1005.5367](https://arxiv.org/abs/1005.5367). <https://doi.org/10.1145/1851399.1851406>. <http://portal.acm.org/citation.cfm?doid=1851399.1851406>
38. Laskey, K.B., Laskey, K.: Service oriented architecture. Wiley Interdisc. Rev. Comput. Stat. **1**(1), 101–105 (2009). <https://doi.org/10.1002/wics.8>
39. Spinnewyn, B., Braem, B., Latre, S.: Fault-tolerant application placement in heterogeneous cloud environments. In: Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015, pp. 192–200. IEEE (2015). <https://doi.org/10.1109/CNSM.2015.7367359>
40. Spinnewyn, B., Mennes, R., Botero, J.F., Latre, S.: Resilient application placement for geo-distributed cloud networks. J. Netw. Comput. Appl. **85**(1), 14–31 (2017). <https://doi.org/10.1016/j.jnca.2016.12.015>
41. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075. ACM (2005)
42. Yu, T., Zhang, Y., Lin, K.J.: Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans. Web (TWEB) **1**, 6 (2007). ACM
43. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: A framework for QoS-aware binding and re-binding of composite web services. J. Syst. Softw. **81**, 1754–1769 (2008). Elsevier

44. Zeng, L., Lingenfelder, C., Lei, H., Chang, H.: Event-driven quality of service prediction. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 147–161. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-89652-4\\_14](https://doi.org/10.1007/978-3-540-89652-4_14)
45. Leitner, P.: Ensuring cost-optimal SLA conformance for composite service providers. ICSOC/ServiceWave 2009. Ph.D. symposium, p. 49 (2009)
46. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F.: Adaptive management of composite services under percentile-based service level agreements. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 381–395. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17358-5\\_26](https://doi.org/10.1007/978-3-642-17358-5_26)
47. Gao, A., Yang, D., Tang, S., Zhang, M.: Web service composition using Markov decision processes. In: Fan, W., Wu, Z., Yang, J. (eds.) WAIM 2005. LNCS, vol. 3739, pp. 308–319. Springer, Heidelberg (2005). [https://doi.org/10.1007/11563952\\_28](https://doi.org/10.1007/11563952_28)
48. Živković, M., Bosman, J.W., van den Berg, J.L., van der Mei, R.D., Meeuwissen, H.B., Núñez-Queija, R.: Run-time revenue maximization for composite web services with response time commitments. In: 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA), pp. 589–596. IEEE (2012)
49. Doshi, P., Goodwin, R., Akkiraju, R., Verma, K.: Dynamic workflow composition using Markov decision processes. *Int. J. Web Serv. Res.* **2**, 1–17 (2005)
50. Choudhury, G.L., Houck, D.J.: Combined queuing and activity network based modeling of sojourn time distributions in distributed telecommunication systems. In: Labetoulle, J., Roberts, J.W. (eds.) *The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks*, Proceedings ITC, vol. 14, pp. 525–534 (1994)
51. Gosavi, A.: Reinforcement learning: a tutorial survey and recent advances. *INFORMS J. Comput.* **21**, 178–192 (2009)
52. Barto, A.G., Mahadeva, S.: Recent advances in hierarchical reinforcement learning. *Discrete Event Dyn. Syst.* **13**, 341–379 (2004). <https://doi.org/10.1023/A:1022140919877>
53. Zheng, H., Zhao, W., Yang, J., Bouguettaya, A.: QoS analysis for web service composition. In: 2009 IEEE International Conference on Services Computing, pp. 235–242. IEEE (2009)
54. Preist, C.: A conceptual architecture for semantic web services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 395–409. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30475-3\\_28](https://doi.org/10.1007/978-3-540-30475-3_28)
55. Bosman, J.W., van den Berg, J.L., van der Mei, R.D.: Real-time QoS control for service orchestration. In: 27-th International Teletraffic Congress, Ghent, Belgium (2015)
56. Poullie, P., Bocek, T., Stiller, B.: A survey of the state-of-the-art in fair multi-resource allocations for data centers. *IEEE Trans. Netw. Serv. Manag.* **15**(1), 169–183 (2017). TNSM 2017
57. Bellard, F.: QEMU, a fast and portable dynamic translator. In: Annual Conference on USENIX Annual Technical Conference, ATEC 2005, p. 41, Anaheim, CA, USA (2005)
58. Selenic Consulting: smem memory reporting tool. <https://www.selenic.com/smem/>. Accessed 7 Feb 2017
59. Phoronix Media: Phoronix test suite (2017). <http://www.phoronix-test-suite.com>. Accessed 18 Jan 2017

60. Poullie, P.: Decentralized multi-resource allocation in clouds. Dissertation, University of Zurich, Zurich, Switzerland, September 2017
61. Gruhler, A.L.: Investigation of resource reallocation capabilities of KVM and OpenStack. Bachelor Thesis, Universität Zürich, Zurich, Switzerland, August 2015. <https://files.ifi.uzh.ch/CSG/staff/poullie/extern/theses/BAGruhler.pdf>
62. Botta, A., de Donato, W., Persico, V., Pescapé, A.: On the integration of cloud computing and Internet of Things. In: The 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014), August 2014
63. Nastic, S., Sehic, S., Le, D., Truong, H., Dustdar, S.: Provisioning software-defined IoT cloud systems. In: The 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014), August 2014
64. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010)
65. Farris, I., Militano, L., Nitti, M., Atzori, L., Iera, A.: MIFaaS: a Mobile-IoT-Federation-as-a-Service model for dynamic cooperation of IoT cloud providers. *Future Gene. Comp. Syst.* **70**, 126–137 (2017)
66. Escribano, B.: Privacy and security in the Internet of Things: challenge or opportunity. In: OLSWANG, November 2014. [http://www.olswang.com/me-dia/48315339/privacy\\_and\\_security\\_in\\_the.iot.pdf](http://www.olswang.com/me-dia/48315339/privacy_and_security_in_the.iot.pdf)
67. Opinion 8/2014 on the on Recent Developments on the Internet of Things, October 2014. <http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2014/wp223.en.pdf>
68. Want, R., Dustdar, S.: Activating the Internet of Things. *Computer* **48**(9), 16–20 (2015)
69. Pflanzner, T., Kertesz, A., Spinnewyn, B., Latre, S.: MobIoTSim: towards a mobile IoT device simulator. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp. 21–27 (2016)
70. IBM IoT Foundation message format. <https://docs.internetofthings.ibmcloud.com/gateways/mqtt.html#/managed-gateways#managed-gateways>. Accessed Mar 2017
71. OpenWeatherMap. <http://www.openweathermap.org>. Accessed Mar 2017

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Efficient Simulation of IoT Cloud Use Cases

Andras Markus<sup>2</sup>, Andre Marques<sup>1</sup>, Gabor Kecskemeti<sup>1</sup>, and Attila Kertesz<sup>2</sup>(✉)

<sup>1</sup> Liverpool John Moores University, Liverpool, UK  
g.kecskemeti@ljmu.ac.uk

<sup>2</sup> University of Szeged, Szeged, Hungary  
keratt@inf.u-szeged.hu

**Abstract.** In the paradigm of Internet of Things (IoT), sensors, actuators and smart devices are connected to the Internet. Application providers utilize the connectivity of these devices with novel approaches involving cloud computing. Some applications require in depth analysis of the interaction between IoT devices and clouds. Research in this area is facing questions like how we should govern such large cohort of devices, which may easily go up often to tens of thousands. In this chapter we investigate IoT Cloud use cases, and derive a general IoT use case. Distributed systems simulators could help in such analysis, but they are problematic to apply in this newly emerging domain, since most of them are either too detailed, or not extensible enough to support the to be modelled devices. Therefore we also show how generic IoT sensors could be modelled in a state of the art simulator using our generalized case to exemplify how the fundamental properties of IoT entities can be represented in the simulator. Finally, we validate the applicability of the introduced IoT extension with a fitness and a meteorological use case.

**Keywords:** Internet of Things · Cloud computing · Simulation

## 1 Introduction

The Internet of Things (IoT) groups connected sensors (e.g. heart rate, heat, motion, etc.) and actuators (e.g. motors, lighting devices) allowing for automated and customisable systems to be utilised [8]. IoT systems are currently expanding rapidly as the amount of smart devices (sensors with networking capabilities) is growing substantially, while the costs of sensors decreases.

IoT solutions are often used a lot within businesses to increase the performance in certain areas and allow for smarter decisions to be made based on more accurate and valuable data. Businesses have grown to require IoT systems to be accurate as decisions based on their data is relied on heavily. An example of IoT in industry is the tracking of parcels for delivery services. The system can provide users with real time information of where their parcel currently is and notify them of potential arrival times. This requires a large infrastructure to facilitate as there is a lot of data being produced.

Many sensors have different behaviour. For example, a heart rate sensor has different behaviour to a light sensor in that a heart rate sensor relies on human behaviour which is inheritably unpredictable, whereas a light sensor could be predicted quite accurately based on the time of day/location. Predicting how a sensor may impact a system is important as companies generally want to leverage the most out of an IoT system however an incorrect estimation of the performance impact can damage the performance of other systems (e.g. using too many sensors could flood the network, potentially causing inaccurate data, slow responses, or system crashes). As there are many ways a sensor can behave it is difficult to predict the impact they may have on a scalable system, therefore they must be tested to determine what the system can handle. Performing this testing could be costly, time consuming, and high risk if the infrastructure has to be created and a wide range of sensors are purchased before any information is obtained about the system. It is even more difficult to determine the impact of a prototype system on the network as there may be limited or no physical sensors to perform tests with. An example of this is the introduction of soil moisture sensors that analyse soil in real time and adjust water sprinklers to ensure crops have the correct conditions to grow. In order to test this IoT system effectively, a lot of these sensors are required, however they can become quite costly and difficult to implement.

There are cloud simulators that provide the tools required to perform a customised simulation of an IoT system which can somewhat accurately simulate the performance impact that a particular setup may have on an infrastructure. The issue with simulators is that due to the wide range of sensor behaviours, to be useful to a wide range of people the simulators cannot be too specific and instead rely on extensions to be implemented in order to function. This requires a lot of specialised code (Such as the sensor's behaviour and the network infrastructure) to be implemented on top of the chosen simulator which can take a lot of time and may have to be altered frequently when situations change. This limits the simulators application as it demands programming skills, a lot of time, and a firm understanding of the API.

In this research work we develop extensions for the DISSECT-CF [5] simulator, which already has the ability to model cloud systems, and has the potential to provide accurate representation of IoT systems. Therefore the goal of this research is to: (i) investigate IoT Cloud use cases, and (ii) derive a general IoT use case. We also show (iii) how generic IoT sensors could be modelled in a state of the art simulator using our generalized case to exemplify how the fundamental properties of IoT entities can be represented in the simulator. Finally, we (iv) validate the applicability of the introduced IoT extension with a fitness and a meteorological use case.

The remainder of this paper is as follows: Sect. 2 presents related work, and in Sect. 3, we detail our proposal for a general use case. In Sects. 5 and 4 we discuss two concrete applications, and the contributions are summarised in Sect. 6.



## 2 Related Work

There are many simulators available to examine distributed and specifically cloud systems. These existing simulators are mostly general network simulators, e.g. Qualnet [1] and OMNeT++ [14]. With these tools IoT-related processes can be examined such as device placement planning and network interference. The OMNeT++ discrete event simulation environment [14] is one of these examples, and it can be used in numerous domains from queuing network simulations to wireless and ad-hoc network simulations, from business process simulation to peer-to-peer network, optical switch and storage area network simulations.

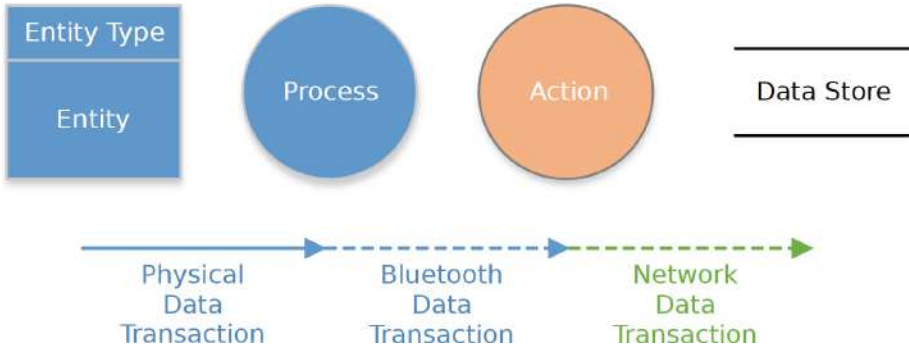
There are more specific IoT simulators, which are closer to our approach. As an example, Han et al. [4] have designed DPWSim, which is a simulation toolkit to support the development of service-oriented and event-driven IoT applications with secure web service capabilities. Its aim is to support the OASIS standard Devices Profile for Web Services (DPWS) that enables the use of web services on smart and resource-constrained devices. SimIoT [13] is derived from the SimIC simulation framework [12], which provides a deeper insight into the behavior of IoT systems, and introduces several techniques that simulates the communication between an IoT sensor and the cloud, but it is limited by its compute oriented activity modeling.

Moschakis and Karatza [9] have introduced several simulation concepts to be used in IoT systems. They showed how the interfacing of the various cloud providers and IoT systems could be modeled in a simulation. They also provided a novel approach to apply IoT related workloads, where data is gathered and processed from sensors taking part in the IoT system. Unfortunately, their work do not consider actuators, and they rather focus on the behavior of cloud systems that support the processing of data originated from the IoT world. The dynamic nature of IoT systems is addressed by Silva et al. [11]. They investigate fault behaviors and introduce a fault model to these systems. Although faults are important for IoT modeling, the scalability of the introduced fault behaviors and concepts are not sufficient for investigating large scale systems that would benefit from decentralized control mechanisms.

Khan et al. [6] introduce a novel infrastructure coordination technique that supports the use of larger scale IoT systems. They build on CloudSim [3], which can be used to model a community cloud based on residential infrastructures. On top of CloudSim they provide customizations that are tailored for their specific home automation scenarios and therefore limit the applicability of their extensions for evaluating new IoT coordination approaches. These papers are also limited on sensors/smart objects thus not allowing to evaluate a wide range of IoT applications that are expected to rise to widespread use in the near future. Zeng et al. [15] proposed IOTSim that supports and enables simulation of big data processing in IoT systems using the MapReduce model. They also presented a real case study that validates the effectiveness of their simulator.

In the field of resource abstraction for IoT, good efforts have been made towards the description and implementation of languages and frameworks for efficient representation, annotation and processing of sensed data. The integration





**Fig. 1.** Model elements of IoT use cases

of IoT and clouds has been envisioned by Botta et al. [2] by summarizing their main properties, features, underlying technologies, and open issues. A solution for merging IoT and clouds is proposed by Nastic et al. [10]. They argue that system designers and operations managers face numerous challenges to realize IoT cloud systems in practice, due to the complexity and diversity of their requirements in terms of IoT resources consumption, customization and runtime governance. We generally share these views in this work, and build on these results by specifying our own contribution in the field of IoT Cloud simulations.

### 3 General IoT Extension for Cloud Simulators

The following section provides a small selection of use cases that display a wide range of behaviours, communication models, and data flows. A wide scope of use cases can provide a much better understanding of the drawbacks with current simulation solutions and will allow us to gain an insight into how we can find a common ground between them. This list is only a partial selection of possible use cases as they were selected based on the potential differences they may have, together building a fairly large pool of behavioural patterns after which introducing more use cases would have had little impact on the overall experiment. The use case figures primarily display data flows (With minor context actions when necessary) as they provide an accurate enough description of the system to understand its behaviour and because simulators generally work via modelling the data transactions between entities.

In Fig. 1 we introduce the basic elements of a generic IoT use case. We use these notations to represent certain properties and elements of these systems. Next we list and define these elements:

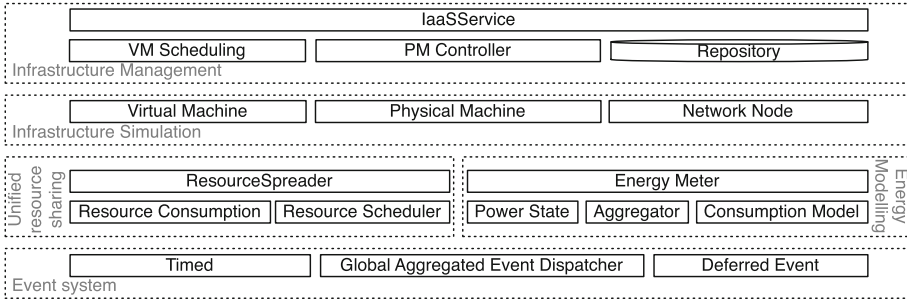
- Entity/Entity Type. The entity box symbolises a physical device with some form of processing or communication powers. We have split the entities into 3 categories: Sensors, Gateway and Server.

**Table 1.** Use case feature requirements

Use cases	Trace model	Trace replay	Custom device	Responsive device
1. Meteorological analysis	✓		✓	✓
2. Automated waste management systems	✓			
3. Real time industrial water contamination system	✓		✓	✓
4. Automated car parking space detector		✓		
5. Vehicle black box insurance system	✓			
6. Fitness watch activity tracker		✓		
7. Smartphone step counter	✓			

- Process. The Process circle represents some form of data processing within the linked Entity. It is used to symbolise the transformation, testing, and/or checking of data flows to produce either more data flows, or a contextual event to trigger. An example of this function can be the interpretation of analog input data from a sensor into something usable.
- Action. The Action circle simply represents a contextual event which generally comes in the form of a physical event. Actions usually require some form of data processing in order to trigger and thus are mostly used at the end of a data flow process. An example of this is a smartphone notification displaying a message from a cloud service.
- Data Store. The Data Store is used primarily by gateways and servers and symbolises the physical disk storage that a device might read/write to. Although this isn't necessary to model, it may help understand some of the diagrams as to where the data may be coming from (As sometimes the data stores are used as a buffer to hold the data).
- Data Transactions. Data Transactions display the movement of data between entities and processes via a range of methods. A Physical Data Transaction refers to a direct link that entities and processes may have, such as a wired connection. Alternatively Bluetooth and Network transactions are differentiated to assist get understanding of how links are formed (To give a small reflection in the distances that can be assumed. Bluetooth having a shorter range than a network transaction).

In Table 1 we gathered the basic feature requirements of representative IoT use cases. We have identified 4 requirements to be supported by simulations focusing on IoT device behaviour:



**Fig. 2.** The architecture of DISSECT-CF, showing the foundations for our extensions

**Trace model.** Allow device behaviour to be characterised by its statistical properties (e.g., distribution functions and their properties like mean, median data packet size, communication frequency etc.).

**Trace replay.** Let devices behave according to real-life recordings from the past. Here we expect devices to be defined with pointers to trace files that contain network, storage and computing activities in a time series.

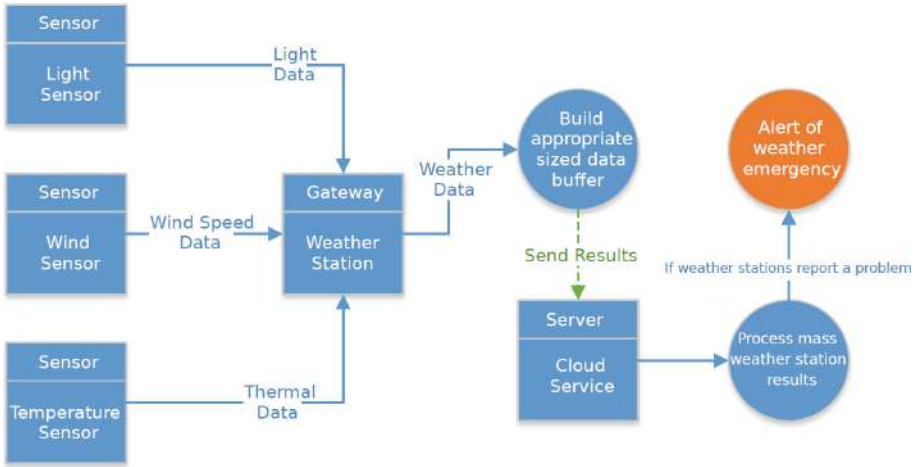
**Custom device.** In general, we expect that most of the simulations could be described by fulfilling the above two requirements. On the other hand, if the built in behaviour models are not sufficient, and there are no traces available, the simulation could incorporate specialised device implementations which implement the missing models.

**Responsive device.** We expect that some custom devices would react to the surrounding simulated environment. Thus the device model is not exclusively dependent on the internals of the device, but on the device context (e.g., having a gateway that can dynamically change its behaviour depending on the size of its monitored sensor set).

Based on these requirements, we examined seven cases ranging from smart region down to smart home applications. We chose to examine these cases by means of simulations, and we will focus on two distinguished cases further on: cases no. 1. and 6.

DISSECT-CF [5] is a compact, highly customizable open source<sup>1</sup> cloud simulator with special focus on the internal organization and behavior of IaaS systems. Figure 2 presents its architecture. It groups the major components with dashed lines into subsystems. There are five major subsystems implemented independently, each responsible for a particular aspect of internal IaaS functionality: (i) event system – for a primary time reference; (ii) unified resource sharing – to resolve low-level resource bottleneck situations; (iii) energy modeling – for the analysis of energy-usage patterns of individual resources (e.g., network links, CPUs) or their aggregations; (iv) infrastructure simulation – to model physical

<sup>1</sup> Available from: <https://github.com/kecskemeti/dissect-cf>.



**Fig. 3.** 1. Use case: meteorological application

and virtual machines as well as networked entities; and finally (v) infrastructure management – to provide a real life cloud like API and encapsulate cloud level scheduling.

As we aim at supporting the simulation of several thousand (or even more) devices participating in previously unforeseen IoT scenarios, or possibly existing systems that have not been examined before in more detail (e.g. in terms of scalability, responsiveness, energy efficiency or management costs). Since the high performance of a simulator’s resource sharing mechanism is essential, we have chosen to use the DISSECT-CF simulator, because of its unified resource sharing foundation. Building on this foundation, it is possible to implement the basic constructs of IoT systems (e.g., smart objects, sensors or actuators) and keep the performance of the past simulator.

The proposed extension provides a runnable Application interface that can take an XML file defining the Machine Data (Such as Physical Machines, Repositories, and their Connection data) and an XML file defining the Simulation Data (Such as the Devices and their behaviours). The Simulation Data can contain a scalable number of Devices and each device has its own independent behaviour model defined. The behaviour of the Device can be modelled in a combination of 3 ways; a direct link to a Trace File (Which should contain the target device, timestamp, and data size), a Trace Producer Model which contains the Distribution set to produce an approximation of the device trace, or finally the simulator can accept device extensions which allow custom devices to be included in the source to programmatically model more specific behaviours.

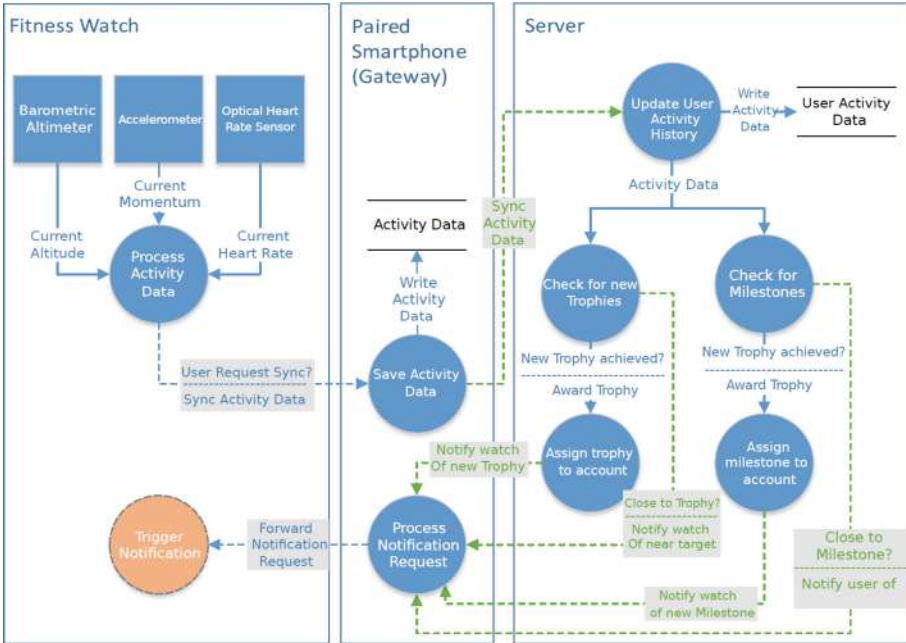


Fig. 4. 6. Use case: fitness tracker application

### 3.1 1. Use Case: Meteorological Application

In Fig. 3 we reveal the typical data flow of a weather forecasting service. This application aims to make weather analysis more efficient by allowing the purchase of a small weather station kit including light sensors (to potentially capture cloud coverage), wind sensors (to collect wind speed), and temperature sensors (to capture the current ambient temperature). The weather station will then create a summary of the sensors findings over a certain period of time and report it to a Cloud service for further processing such as detecting hurricanes or heat waves in the early stages. If many of these stations are set up over a region, it can provide accurate and detailed data flow to the cloud service to produce accurate results.

In order to simulate this application, the simulator need to provide appropriate tools for performing the communications and processing, defining the behaviour of the sensors and the weather station require a modelling technique to be implemented on top of the simulator (which was achieved by programming the sensors data production and the stations buffer reporting).

### 3.2 6. Use Case: Fitness Tracking Application

In Fig. 4 we reveal the data flow typically encountered when wearables or fitness trackers like fitbit are used. This use case aims to track and encourage the activity

of a user by collecting a wide range of data about the user (Such as current heart rate, step count, floors climbed, etc.). This data is generally collected by the wearable device and sent to the smart phone when the user accesses the smartphone applications and requests the devices to synchronise, after which the data will then be synced from the smartphone to the cloud as well, for more data processing (which could result in trophies and milestones encouraging further use of the wearable).

This provides an interesting range of behaviour as it contains a feedback mechanism to provide incentive to the user to perform specific actions based on certain circumstances. This is displayed within the Trophy and Milestone system that is implemented server side that will track certain metrics (such as average time being active daily) and provide notifications when they are reaching a goal (like a daily milestone of 1 h active per day).

This mechanism introduces an important behaviour model whereby the sensors produce data that can trigger events that indirectly change the behaviour of the sensors via a feedback loop. An example of this feedback loop can be the daily activity milestone whereby a user may perform 45 min of activity and decide to take a rest, at this point the sensors will revert back to their baseline behaviour (user is inactive therefore the sensors provide less data), however the system notifies the user that only an extra 15 min is necessary to reach their milestone (the feedback), and thus the user may decide they want to hit their target and perform more activity which will then change the behaviour of the sensors yet again.

It would be difficult to simulate this case via modelling strategies as the feedback mechanism combined with the unpredictable and wide ranging human activity (most users will have different times that they are active, levels of intensity, and duration of exercise) have too many variables to take into consideration. There is also the consideration of the time of day being a large factor to the behaviour of the sensor, as it can be expected that the sensor will provide far less activity data during the night when the user is likely sleeping when compared to the day time. This is further compounded by time zone differences whereby if the system is used in multiple time zones it would be harder to model due to differences in when a user base may be asleep or not.

Due to the above reasons it would be required that a wide range of traces were collected in order to be able to obtain a large enough sample size of different behaviour models to run an accurate simulation of the system (which could be scaled up/down as required). This introduces problems with current simulator solutions as not only is replay functionality needed, but there must be the possibility of replaying several different traces simultaneously in order to test a system with the multitude of different behavioural models that can be expected (As there would be no point in running a simulation of a single behaviour model considering the real world application is vastly different).

## 4 Implementing the Extension for a Meteorological Application

Based on the generic plans discussed before, we performed the extension of the DISSECT-CF simulator towards a meteorological application covering a wider region. To derive the sensor models for the extension, we started by modelling a real-world IoT system: as one of the earliest examples of sensor networks are from the field of meteorology and weather prediction, we choose to model the crowdsourced meteorological service of Hungary called Idokep.hu. It has been established in 2004, and it is one of the most popular websites on meteorology in Hungary. Since 2008 weather information can be viewed on Croatia and even on Germany. Detailed information of its system architecture and operation can also be found on the website: more than 400 stations send sensor data to their system (including temperature, humidity, barometric pressure, rainfall and wind properties), and the actual weather conditions are refreshed every 10 min. They also provide forecasts up to a week. They also produce and sell sensor stations capable to extend their sensor network and improve their weather predictions. These can be bought and installed at buyer specific locations.

We followed a bottom-up approach to add IoT functionalities to the simulator, and implemented a weather prediction application using public data available on sensors and their behaviour at <http://www.idokep.hu>.

Each entity that aims to perform repeated events in DISSECT-CT has to use the **Timed** class (see Fig. 2), by implementing the **tick()** method. We added two of such classes, the **Application** and the **Station**. The **Station** is an entity acting as a gateway. I.e., it provides the network connection for sensors, and optimises the network usage of the sensors by caching and bundling outgoing metering data of its supervised sensors. Figure 5 depicts how data stored about each station in an IoT system. This description is useful to set up predefined stations from files. The **tasksize** attribute of **Application** defines the amount of data (in bytes) to be gathered in a cloud storage (sent by the stations) before their processing in a VM.

**Stations** have unique identifiers (i.e., a **name**). We can specify their lifetime with the tag **time** by defining their **starttime** and **stoptime**. The cardinality of the supervised sensor set is set via **sbnumber**. Alongside the set cardinality, one can also specify the average data **size** produced by one of the sensors in the set. To set up more stations with the same properties, one can use the **count** option in the **name** tag. Data generation frequency (**freq**) could be set for the sensor set (in milliseconds). The station's caching mechanism is influenced with the tag **ratio**. This defines the amount of data to be kept at the local storage relative to the average dataset produced by the sensors at each data generation event. If the unsent data in the local storage (which is defined in **storage**) overreaches the caching limit, the station is modelled to send the cached items to the cloud's storage (identified with its network node id specified in the **torepo** tag). The local storage is also keeping a log of previously sent data until its capacity (defined in the **storage** tag) is exceeded. The station's network connectivity to the outside world is specified by the tags **maxinbw** and **maxoutbw**.

```

<Application tasksize='250000'>
<Station>
  <name count='1'>Szeged</name>
  <freq>60000</freq>
  <snumber size='200'>10</snumber>
  <time starttime='500'
    stoptime='1000'>
    1000
  </time>
  <maxinbw>100</maxinbw>
  <maxoutbw>100</maxoutbw>
  <storagebw>100</storagebw>
  <torepo>sztakilpdsceph</torepo>
  <storage>60000</storage>
  <ratio>1</ratio>
</Station>
</Application>

```

**Fig. 5.** XML-based description of IoT systems

Individual Station entries in the XML are saved in the **StationData** java bean. The actual data generation of the sensors is performed by the **Metering** class.

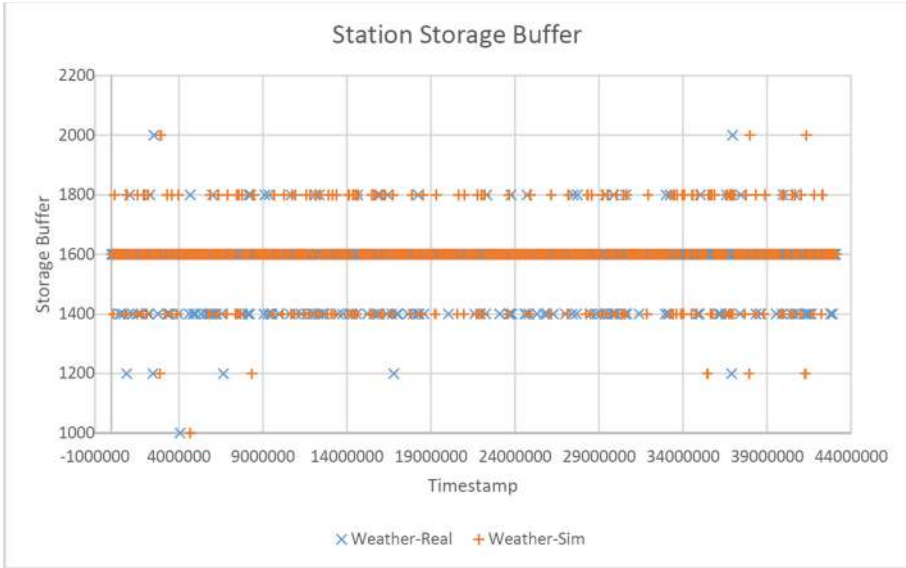
The **Cloud** class can be used to specify and set up a cloud environment. This class uses DISSECT-CF's XML based cloud loader to set up a cloud environment to be used for storing and processing data from stations. This class should also be used to define Virtual Appliances modeling the application binaries doing the in cloud processing.

The scenarios to be examined through simulations should be defined by the **Application** class. Users are expected to implement custom IoT Cloud use cases here by examining various management and processing algorithms of sensor data in VMs of a specific cloud environment. The **VmCollector** class can be used to manage such VMs, and its **VmSearch()** method can be used to check if there is a free VM available in the cloud to be utilized for a certain task. If this is not the case, the **generateAndAdd()** method can be used to deploy a new one.

#### 4.1 Implementation with the Generic IoT Oriented Extensions

The weather station's caching behaviour is a prime example for the need of responsive device implementations. As the sensors produce data independently from each other, and they could have varying frequencies and data sizes, the station must cache all produced data before sending it to the cloud for processing. This behaviour was modelled as a custom, responsive device for which we overrode the **tick()** function of our new device sub-class. In DISSECT-CF terminology, this function is the one that is used to represent periodic events in the simulation,





**Fig. 6.** Analysis of the buffering behaviour in the alternative simulations of a weather station

in this particular case it was used to simulate the data reporting requests from the cloud. Each station has connections to its 8 sensors, which produced randomly sized data with the frequency of  $[\frac{1}{60} - 1]$  Hz. Upon every tick call, our custom device determines if there is a need to send its buffered contents to the cloud or not. This is based on the buffered data size that was set to be at least 1 kB before emptying the buffer.

The implementation was tested by running the original and the new implementations side-by-side so that we could analyse the network traffic differences. Due to the random nature of the data production the two solutions don't completely line up, however Fig. 6 displays how the simulation extension produces a very similar result to the original implementation in that although there is a lot of randomness to the investigated scenario, the mean and median values are having a close match. The distribution is also following the same pattern: whereby the bulk of the buffer loads are within 1600 bytes and are less frequent the further away from this value it goes.

At it can be observed, the basic extensions described here are mainly focusing on device behaviour. The application level operations are completely up to the user to define. E.g., application logic for how many virtual machines do we need for processing the sensor data is not to be described by the XML descriptors. In the next sub-section we will discuss such situations and explore how to combine application level behaviour with the new sensor and device models.

## 4.2 Evaluation with Alternative Application Level Scenarios

During our implementation and evaluation, where applicable, we used publicly available information to populate our experiments. Unfortunately, some details are unpublished (e.g. sensor data sizes, data-processing times), for those, we have provided estimates and listed them below.

In the website of Idokep.hu<sup>2</sup>, we learnt that the service operates with 487 stations. Each of them has sensors at most monitoring the following environmental properties:

1. timestamp;
2. air and dew point temperature – °C;
3. humidity – %;
4. barometric pressure – in hPa;
5. rainfall – mm/hour and mm/day;
6. wind speed – km/h;
7. wind direction;
8. and UV-B level.

Concerning the size of such sensor data, we expect them to be save in a structured text file (eg., CSV). Stored this way, we can estimate that approximately 50 bytes (e.g., based on the website of the Murdoch University Weather Station<sup>3</sup>) are produced if each sensor produces data in every measurement.

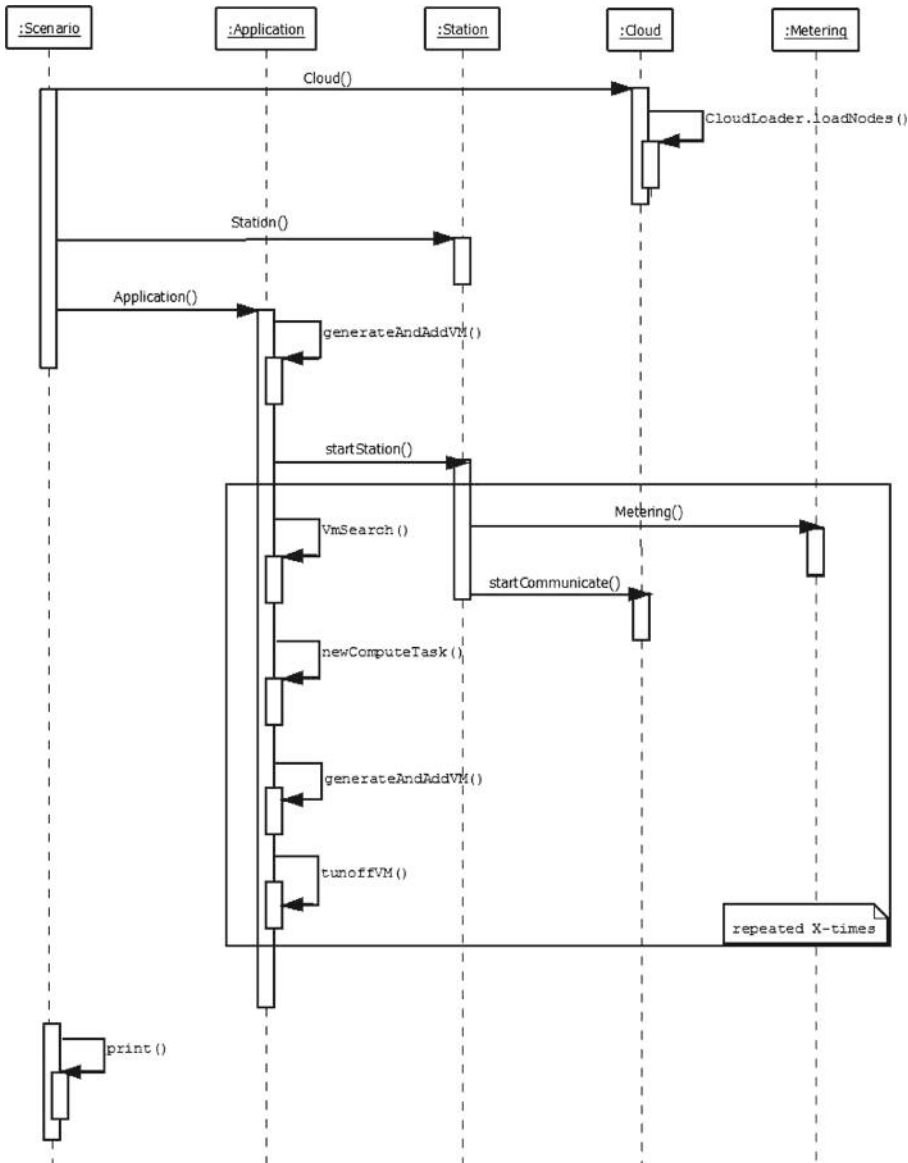
Next, we detail the steps of the behaviour of our **Application** implementation which was used for all evaluation scenarios later (see Fig. 7):

1. Set up the cloud using an XML. As we expect meteorological scenarios will often use private clouds, we used the model of our local private infrastructure (the LPDS Cloud of MTA SZTAKI);
2. Set up the 487 stations (using a scenario specific XML description) with the previously listed 8 sensors per station;
3. Start the **Application** to deploy an initial VM (`generateAndAddVM()`) for processing and to start the metering process in all stations (`startStation()`);
4. The stations then monitor (`Metering()`), save and send (`startCommunicate()`) sensor data (to the cloud storage) according to their XML definition;
5. A daemon service checks regularly if the cloud repository received a scenario specific amount of data (see the `tasksize` attribute in Fig. 5). If there so, then the **Application** generates tasks which will finish processing within a predefined amount of time.
6. Next, for each generated task, a free VM is searched (by `VmSearch()`). If a VM is found, the task and the relevant data is sent to it for processing.
7. In case there are no free VMs found, the daemon initiates a new VM deployment and holds back the not yet mapped tasks.

<sup>2</sup> <http://idokep.hu/automata>.

<sup>3</sup> <http://wwwmet.murdoch.edu.au/downloads>.

8. If at the end of the task assignment phase, there are still free VMs, they are all decommissioned (by `turnoffVM()`) except the last one (allowing the next rounds to start with an already available VM). Note this behaviour could be turned on/off at will.
9. Finally, the `Application` returns to step 5.



**Fig. 7.** Sequence diagram of the weather station modelling use case and its relations to our DISSECT-CF extensions

### 4.3 Evaluation

In this sub-section, we reveal five scenarios investigating questions likely to be investigated with the help of extended DISSECT-CF. Namely, our scenarios mainly focus on how resource utilization and management patterns alter based on changing sensor behaviour (e.g., how different sensor data sizes and varying number of stations and sensors affect the operation of the simulated IoT system). Note, the scope of these scenarios is solely focused on the validation of our proposed IoT extensions and thus the scenarios are mostly underdeveloped in terms of how a weather service would behave internally.

Before getting into the details, we clarify the common behaviour patterns, we used during all of the scenarios below. First of all, to limit simulation runtime, all of our experiments limited the station lifetimes to a single day. The start-up period of the stations were selected randomly between 0 and 20 min. The task creator daemon service of our **Application** implementation spawned tasks after the cloud storage received more than 250 kB of metering data (see the **tasksize** of Fig. 5). This step ensured the estimated processing time of 5 min/task. VMs were started for each 250 kB data set. The cloud storage was completely run empty by the daemon: the last spawned task was started with less than 250 kB to process – scaling down its execution time. Finally, we disabled the dynamic VM decommissioning feature of the application (see step 8 in Sect. 4.2).

In scenario N°1, we varied the amount of data produced by the sensors: we set 50, 100 and 200 bytes for different cases (allowing overheads for storage, network transfer, different data formats and secure encoding etc.). We simulated the 487 stations of the weather service. Our results can be seen in Fig. 8a and b. For the first case with 50 bytes of sensor data we measured 256 MBs of produced data in total, while in the second case of 100 bytes we measured 513 MBs, and in the third of 200 bytes we measured 1.02 GBs (showing linear scaling up). In the 3 cases we needed 6, 10 and 20 VMs to process all tasks respectively.

In scenario N°2, we wanted to examine the effects of varying sensor numbers and varying sensor data sizes per stations to mimic real world systems better. Therefore, we defined a fixed case using 744 stations having 7 sensors each, producing 100 bytes of sensor data per measurement, and a random case, in which we had the 744 stations with randomly sized sensor set (ranging between 6–8) and sensor data size (50, 100 or 200 bytes/sensor). The results can be seen in Fig. 9a and b. As we can see we experienced minimal differences; the random case resulted in slightly more tasks.

In scenario N°3, we examined random sensor data generation frequencies. We set up 600 stations, and defined cases for two static frequencies (1 and 5 min), and a third case, in which we randomly set the sensing frequency between 1 and 5. In real life, the varying weather conditions may call for (or result in) such changes. In both cases, the sensors generated our previously estimated 50 bytes. The results can be seen in Fig. 10a, b and c. As we can see the generated data in total: 316 MBs for 1 min frequency, 63 MBs for 5 min frequency, and 143 MBs for the randomly selected frequencies. Here we can see that the first

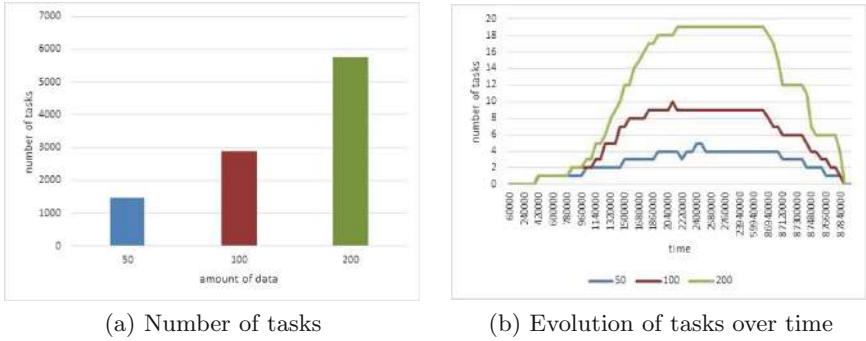


Fig. 8. Scenario N°1

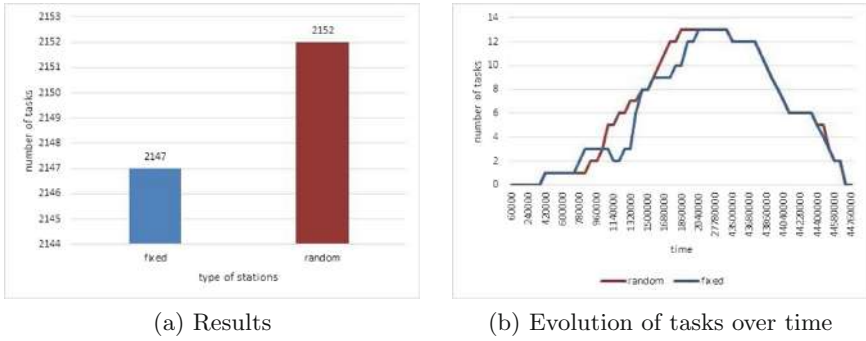
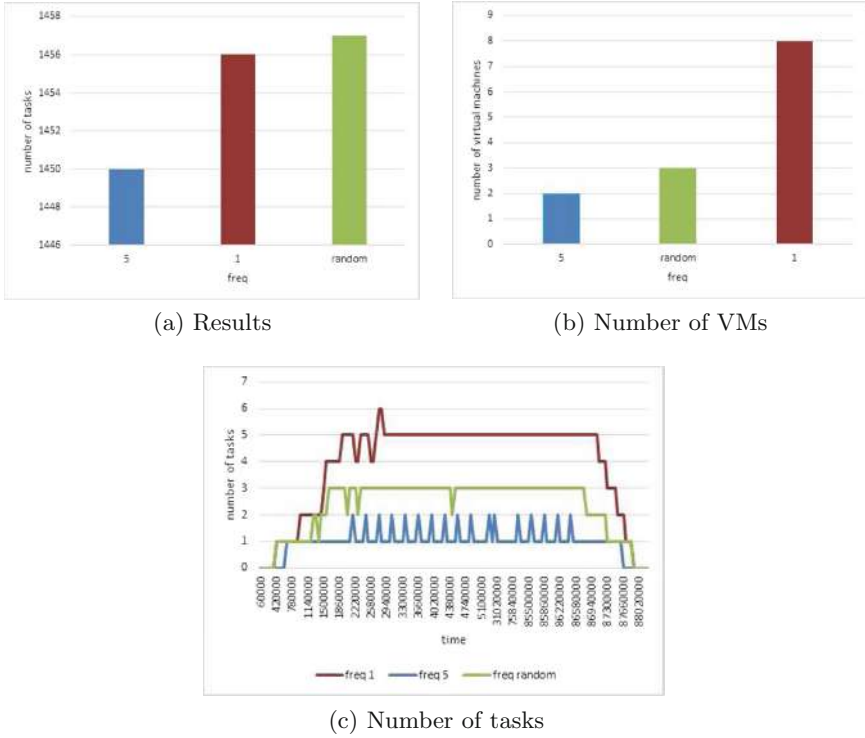


Fig. 9. Scenario N°2

case required the highest number of VMs to process the sensed data, but the randomly modified sensing frequency resulted in the highest number of tasks.

In the three scenarios executed so far the main application, responsible for processing the sensor data in the cloud, checked the repository for new transfers in every minute. In some cases we experienced that only small amount of data has arrived within this interval (i.e. task creation frequency). Therefore in scenario N°4, we examined what happens if we widen this interval to 5 min. We executed three cases here with 200, 487 and 600 stations. The results can be seen in Fig. 11a. In Fig. 11b, we can read the number of VMs required for processing the tasks in the actual case. The first case has the highest difference in terms of task numbers: data coming from sensors of 200 stations needed more than 1400 tasks with 1 min interval, while less than 600 with 5 min interval. It is also interesting that with 600 stations almost the same amount of tasks were generated, but with the 5 min interval we needed more VMs to process them.

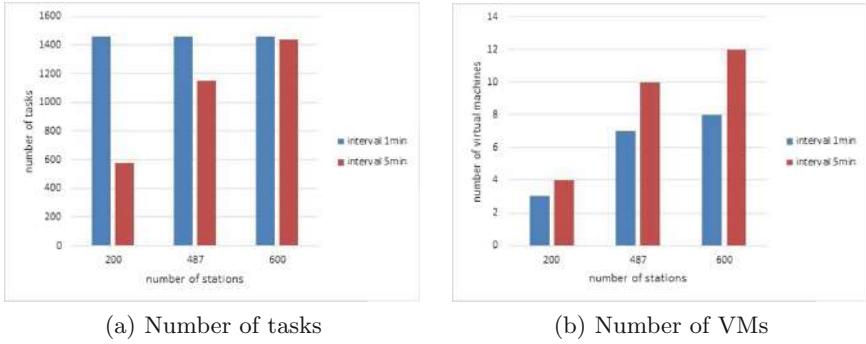
As we model a crowdsourced service, we expect to see a more dynamic behaviour regarding stations. In the previous cases we used static number of



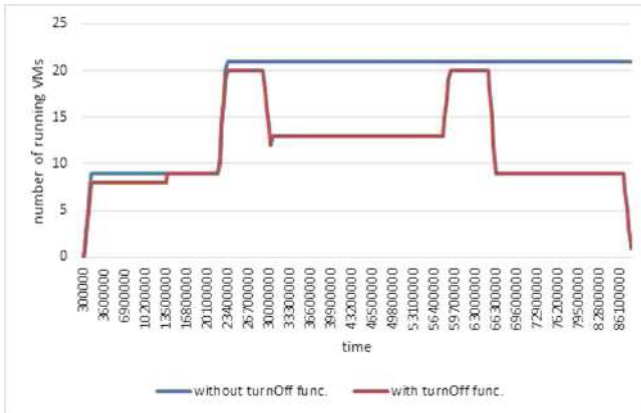
**Fig. 10.** Scenario N°3

stations per experiment, while in our final scenario, N°5, we ensured station numbers dynamically change. Such changes may occur due to station or sensor failures, or even by sensor replacement. In this scenario we performed these changes by specific hours of the day: from 0–5 am we started 200 stations, from 6–8 am we operated 500 stations, from 9 am to 15 pm we scaled them down to 300, then from 16–18 up to 500, finally the last round from 19–24 pm we set it back to 200. In this experiment we also wanted to examine the effects of VM decommissioning, therefore we executed two different cases, one with and one without turning off unused VMs. In both cases we set the `tasksize` attribute to 10 kB (instead of the usual 250 kB). The results can be seen in Fig. 12. We can see that without turning off the unused VMs from 6 pm we kept more than 20 VMs alive (resulting in more overprovisioning), while in the other case the number of running VMs dynamically changed to the one required by the number of tasks to be processed.

As a summary, in this section we presented five scenarios focusing on various properties of IoT systems. We have shown that with our extended simulator, we can investigate the behaviour of these systems and contribute to the development of better design and management solutions in this research field.



**Fig. 11.** Scenario N°4



**Fig. 12.** Results of scenario N°5

## 5 Implementing the Extension for a Fitness Application

This use case was selected for implementation to allow us to replay real world data logs for multiple devices so that we could test the simulators trace replaying capabilities. It is important that the application can run through the trace logs for each device individually and correctly perform the network transfers that are detailed in it. The trace logs to be played were acquired with a special traffic interception application developed for the smartphone. Our application collected access and network traffic logs for the watch, smartphone, and the cloud. After data collection, the logs were saved in a file format ready to be used as an input trace to the simulator. This extension has been performed within a BSc thesis work [7] at the Liverpool John Moores University, UK.

## 5.1 Trace Collection

Initially, we aimed to collect all of the network traffic between the three devices with a packet analysing software (such as Wireshark) on a laptop that acted as a wireless hotspot for the smartphone. However, this severely limited the accuracy of the traces as this requires disabling the network of the fitness application, when the phone is not connected to the laptop (to ensure all its communication with the cloud is caught). On top of this, we would have lost the ability to trace the Bluetooth traffic between the watch and the smartphone.

As a result, we turned our attention of to methods that intercept network traffic directly through the phone. Despite the multitude of third party android network traffic analysers, we could not find one that met our requirements: (i) should run at the background (allowing us to use the fitness application at will); (ii) should have output logs on network and bluetooth activity either directly processable by the simulator or in a format that could be easily transformed to the needed form; and (iii) should remain active for long periods of time (as the log collection ran for days).

As a result, we have decided to create an application that met all of these requirements and would allow us to localise the data collection into one place. The Fitbit connection monitor application<sup>4</sup> is built on top of an android subsystem called the Xposed Framework. Using this framework, we were able to intercept socket streams for network I/O, while for bluetooth, we have used intercepted traffic through android's GATT service.

A sample of intercepted data traces is shown in Fig. 13. This figure shows the data that was collected from the Fitbit Connection Monitor over the course of around 2 weeks (over 20,000 trace entries of real life data). There are several interesting situations one can observe in the raw data. First, it shows peaks of network activity in cases when: (i) there was a manually invoked data synchronisation (ii) or when the user issued firmware update request for the watch. In contrast, there were gaps in the data collection as well. These gaps represent situations such as: (i) the user did not wear his/her watch, (ii) Bluetooth was disabled on the smartphone or (iii) the watch was not switched on (e.g., because of running out of battery power).

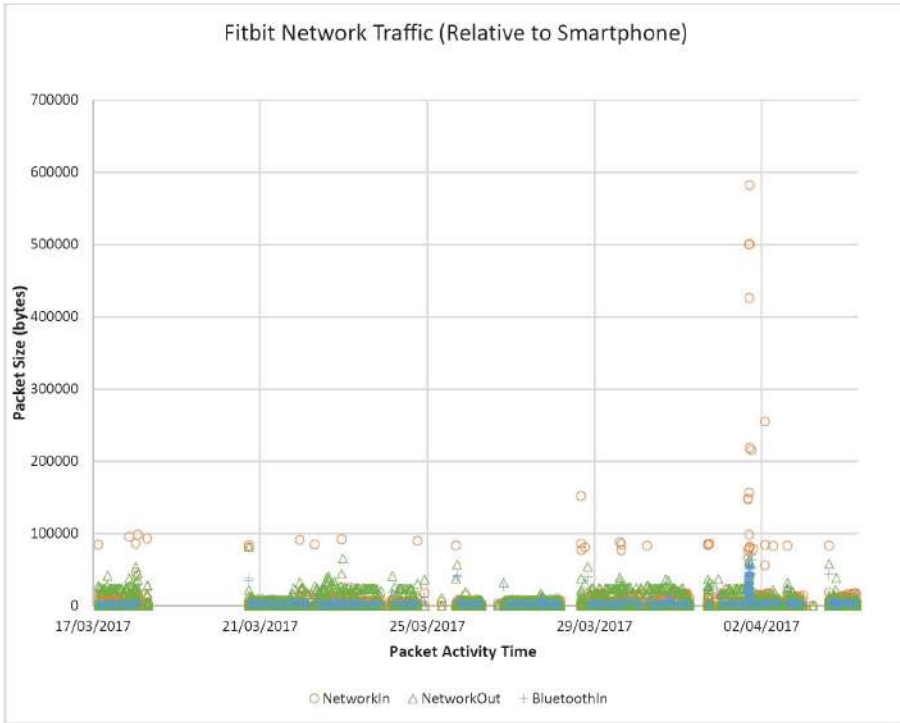
## 5.2 Implementation and IoT Extensions to DISSECT-CF

In our initial implementation, we have followed a similar approach as we did with the meteorological case. We have implemented the fitness use case with the original DISSECT-CF APIs. Then we also implemented a solution that was built on top of the our new IoT oriented extensions of DISSECT-CF APIs<sup>5</sup>. To better understand this solution, first we summarize the extensions.

<sup>4</sup> The application is open source and available at <https://github.com/Andrerm124/FitbitConnectionMonitor>.

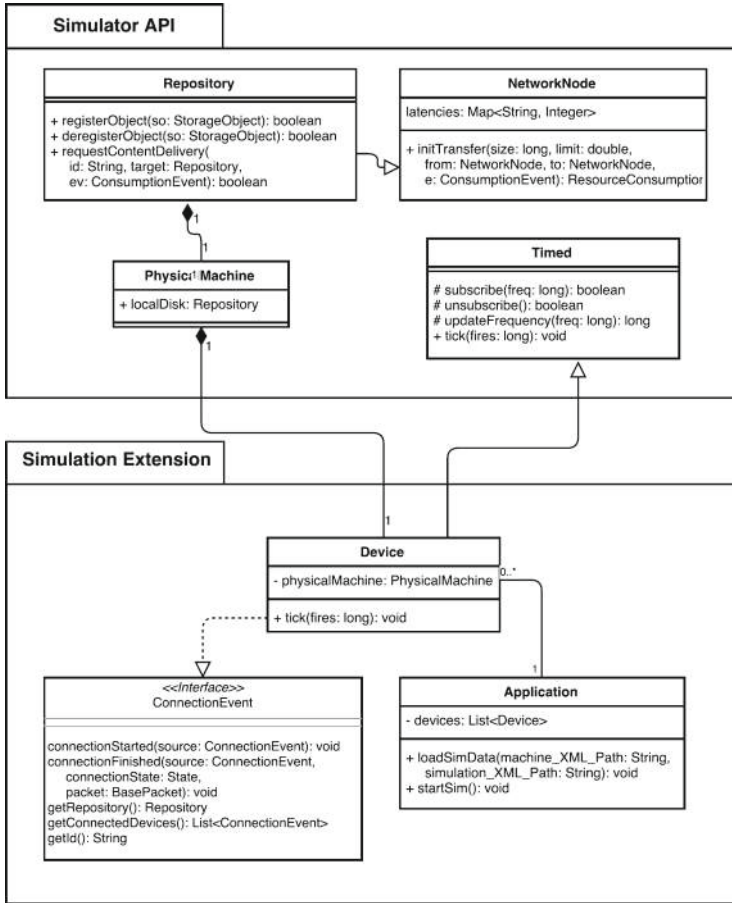
<sup>5</sup> The source code of the second implementation is available online at <https://github.com/Andrerm124/dissect-cf/tree/FitbitSimulation>.





**Fig. 13.** Real-life network traffic in the fitness use case according to the long term trace collection results

Figure 14 presents the new extensions to DISSECT-CF. With the extension, one can define a simulation with two XML files. First, the original simulator API loads all of the physical machines from the supplied Machine XML file (the loaded up machines will represent the computational, network and storage capabilities of the IoT devices). In the second XML, device models can be linked to each of the previously loaded machines. Each model can be customised independently by altering the desired attributes of the built in device templates. In these templates, one can define the following details: (i) machine id to bind to, (ii) time interval for the presence of the device, (iii) custom attributes and behaviour – this part still must be coded in java –, (iv) network behaviour – in the form of a trace or a distribution function, (v) typical network endpoints and (vi) data storage and caching options (both device local and remote – e.g., in the cloud). The loading of these XML files and the management of the device objects is accomplished by the `Application` class. Finally, the extension provides alternative packet routing models as well in the form of the several implementations for the `ConnectionEvent` interface.



**Fig. 14.** The IoT oriented DISSECT-CF extensions

To analyse the effectiveness of our extensions, we have compared the development time and the simulation results for the fitness application. The initial implementation has been created as custom classes for all devices participating in the use case. This required approximately 3 days of development time. In contrast, with the new extensions, barely more than 20 lines of XML code (shown in Fig. 15) plus the previously collected trace files were required to define the whole simulation. To validate the new implementation, we also compared the data produced from this new and the initial completely java based implementation. We have concluded that the two implementations produced equivalent results (albeit the XML based one allowed much more rapid changes to device configurations and to their behaviour).

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Simulation>
  <Devices>
    <Device>
      <ID>Watch</ID>
      <TraceFileReader>
        <SimulationFilePath>bluetooth_in.csv</SimulationFilePath>
      </TraceFileReader>
    </Device>
    <Device>
      <ID>Smartphone</ID>
      <TraceFileReader>
        <SimulationFilePath>network_out.csv</SimulationFilePath>
      </TraceFileReader>
    </Device>
    <Device>
      <ID>Cloud</ID>
      <TraceFileReader>
        <SimulationFilePath>network_in.csv</SimulationFilePath>
      </TraceFileReader>
    </Device>
  </Devices>
</Simulation>

```

**Fig. 15.** XML model of the fitness use case

### 5.3 Evaluation

To evaluate our extensions, we have set up the exact same situation in the simulation as we have had during the trace collection. We also ensured the simulation writes its output in terms of simulated network and computing activities in the same format as the originally collected traces. This allowed easy comparison between the simulated and the real-life traces. Figure 16 show the comparison of the bluetooth trace. According to the figure, the simulation can accurately reproduce the real-life traces, i.e., the simulated data transfers occur at the prescribed times and have the same levels of data movement as the ones recorded in real-life. The network communication between the cloud and the smartphone has shown similar trends (thus the simulation was capable to reproduce the complete Fig. 13).

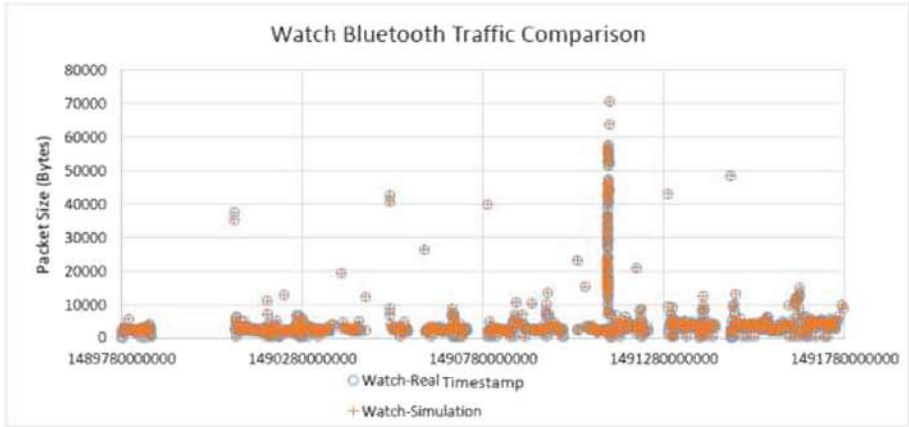


Fig. 16. Watch network traffic comparison

## 6 Conclusion

Distributed systems simulators are not generic enough to be applied in newly emerging domains, such as IoT Cloud systems, which require in depth analysis of the interaction between IoT devices and clouds. Research in this area is facing questions like how we should govern such large cohort of devices, which may easily go up often to tens of thousands.

In this chapter we investigated various IoT Cloud use cases, and derived a general IoT use case. We have shown, how generic IoT sensors could be modelled in the DISSECT-CF simulator, and exemplified how the fundamental properties of IoT entities can be represented. Finally, we validated the applicability of the introduced IoT extension with a fitness and a meteorological application.

**Acknowledgments.** The research leading to these results has received funding from the European COST programme under Action identifier IC1304 (ACROSS), and it was supported by the UNKP-17-4 New National Excellence Program of the Ministry of Human Capacities of Hungary. A part of this research has been performed within a BSc thesis work of A. Marques [7] at the Liverpool John Moores University, UK.

## References

1. QualNet communications simulation platform. <http://web.scalable-networks.com/content/qualnet>. Accessed Jan 2016
2. Botta, A., De Donato, W., Persico, V., Pescapé, A.: On the integration of cloud computing and internet of things. In: International Conference on Future Internet of Things and Cloud (FiCloud), pp. 23–30. IEEE (2014)
3. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Experience* **41**(1), 23–50 (2011)

4. Han, S.N., Lee, G.M., Crespi, N., Heo, K., Van Luong, N., Brut, M., Gatellier, P.: DPWSim: a simulation toolkit for IoT applications using devices profile for web services. In: IEEE World Forum on Internet of Things (WF-IoT), pp. 544–547. IEEE (2014)
5. Kecskemeti, G.: DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simul. Model. Pract. Theor.* **58**(P2), 188–218 (2015)
6. Khan, A.M., Navarro, L., Sharifi, L., Veiga, L.: Clouds of small things: provisioning infrastructure-as-a-service from within community networks. In: 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 16–21. IEEE (2013)
7. Marques, A.: Abstraction and Simplification of IoT System Modelling Using a Discrete Cloud Event Simulator. B.Sc. thesis, Department of Computer Science, Liverpool John Moores University, Liverpool, UK, April 2017
8. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: vision, applications and research challenges. *Ad Hoc Netw.* **10**(7), 1497–1516 (2012)
9. Moschakis, I.A., Karatza, H.D.: Towards scheduling for internet-of-things applications on clouds: a simulated annealing approach. *Concurrency Comput. Pract. Experience* **27**(8), 1886–1899 (2015). <https://doi.org/10.1002/cpe.3105>
10. Nastic, S., Sehic, S., Le, D.H., Truong, H.L., Dustdar, S.: Provisioning software-defined IoT cloud systems. In: 2014 International Conference on Future Internet of Things and Cloud (FiCloud), pp. 288–295. IEEE (2014)
11. Silva, I., Leandro, R., Macedo, D., Guedes, L.A.: A dependability evaluation tool for the internet of things. *Comput. Electr. Eng.* **39**(7), 2005–2018 (2013)
12. Sotiriadis, S., Bessis, N., Antonopoulos, N., Anjum, A.: SimIC: designing a new inter-cloud simulation platform for integrating large-scale resource management. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 90–97. IEEE (2013)
13. Sotiriadis, S., Bessis, N., Asimakopoulou, E., Mustafee, N.: Towards simulating the internet of things. In: 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 444–448. IEEE (2014)
14. Varga, A., et al.: The OMNeT++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference (ESM 2001), vol. 9, p. 185. sn (2001)
15. Zeng, X., Garg, S.K., Strazdins, P., Jayaraman, P.P., Georgakopoulos, D., Ranjan, R.: IOTSim: a simulator for analysing IoT applications. *J. Syst. Architect.* **72**, 93–107 (2016)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Security of Internet of Things for a Reliable Internet of Services

Ahmet Arı<sup>1</sup>✉, Sema F. Oktug<sup>1</sup>, and Thiemo Voigt<sup>2</sup>

<sup>1</sup> Faculty of Computer and Informatics Engineering,  
Istanbul Technical University, Istanbul, Turkey  
{[arisahmet,oktug@itu.edu.tr](mailto:arisahmet,oktug@itu.edu.tr)}

<sup>2</sup> Swedish Institute of Computer Science (SICS), Kista, Sweden  
[thiemo@sics.se](mailto:thiemo@sics.se)

**Abstract.** The Internet of Things (IoT) consists of resource-constrained devices (e.g., sensors and actuators) which form low power and lossy networks to connect to the Internet. With billions of devices deployed in various environments, IoT is one of the main building blocks of future Internet of Services (IoS). Limited power, processing, storage and radio dictate extremely efficient usage of these resources to achieve high reliability and availability in IoS. Denial of Service (DoS) and Distributed DoS (DDoS) attacks aim to misuse the resources and cause interruptions, delays, losses and degrade the offered services in IoT. DoS attacks are clearly threats for availability and reliability of IoT, and thus of IoS. For highly reliable and available IoS, such attacks have to be prevented, detected or mitigated autonomously. In this study, we propose a comprehensive investigation of Internet of Things security for reliable Internet of Services. We review the characteristics of IoT environments, cryptography-based security mechanisms and D/DoS attacks targeting IoT networks. In addition to these, we extensively analyze the intrusion detection and mitigation mechanisms proposed for IoT and evaluate them from various points of view. Lastly, we consider and discuss the open issues yet to be researched for more reliable and available IoT and IoS.

**Keywords:** IoT · IoT security · IoS · DoS · DDoS  
Internet of Things · Internet of Services · Reliable IoS

## 1 Introduction

Internet of Things is a network of sensors, actuators, embedded and wearable devices that can connect to the Internet. Billions of devices are expected to be part of this network and make houses, buildings, cities and many other deployment areas smarter [17]. In order to reach populations as much as billions, elements of IoT network are expected to be cheap and small form-factor devices with limited resources.

IoT is a candidate technology in order to realize the future Internet of Services and Industry 4.0 revolution. Accommodation of billions of devices with sensing

and/or actuation capabilities will introduce crucial problems with management, interoperability, scalability, reliability, availability and security. Autonomous control and reliability of future IoS are directly related to reliability and availability of IoT. However, there are serious threats for IoT, which aim to degrade the performance of the network, deplete the batteries of the devices and cause packet losses and delays. These attacks are called as Denial of Service attacks, which are already notorious for their effects in existing communication systems. Limited power, processing, storage and radio dictate extremely efficient usage of these resources to achieve high reliability and availability in IoS. However, DoS and DDoS attacks aim to misuse the resources and cause interruptions, delays, losses and degrade the offered services in IoT. DoS attacks are clearly threats for availability and reliability of IoT, and thus of IoS. For highly reliable and available IoS, such attacks have to be prevented, detected or mitigated autonomously.

DoS and DDoS attacks can target any communication system and cause devastation. Such attacks make use of the vulnerabilities in the protocols, operating systems, applications and actual physical security of the target system. Readers can easily find several incident news related to D/DoS attacks on the Internet. These attacks are so common that every day it is possible to see them (e.g., please check the digital attack map of Arbor Networks and Google Ideas [3]). It is not hard to predict that IoT will face with D/DoS attacks, either as a target or source of the attacks. In fact, quite recently one of the major Domain Name System (DNS) infrastructure provider of popular web sites and applications was the target of DDoS attacks where a botnet called as *Mirai* compromised thousands of cameras and digital video recorder players [2]. This incident was the first example of IoT being used as an attack source for DDoS. It clearly showed that, protection of IoT networks from attacks is not sufficient and protection of the Internet from IoT networks is needed as well.

A very interesting report [53] on how security of IoT will be playing an important role in defining the cybersecurity of future was published by UC Berkeley Center for Long-Term Cybersecurity in 2016. A group of people from various disciplines developed five scenarios regarding with what will security be like in the future considering various dimensions including people, governments, organizations, companies, society, culture, technological improvements and of course attackers. Although all of the scenarios are related to the security of IoT, the last two scenarios have direct relations. The fourth scenario puts the emphasis on the ubiquity of IoT in a way that IoT will be everywhere and will be playing a vital role on the management of several applications and systems. This will give attackers more chance to target. In such a world, attackers will be able to affect organizations, governments and the daily life of people easier than now. Thus, cybersecurity term will be transformed to just *security* since it will be able to affect everything. The last scenario considers the wearable devices and their novel purpose of use. According to the hypothesis, the wearables of future will not only perform basic measurement tasks, but will be used to track emotional states of humans. Advancements in the technologies will allow such a change.

Emotional, mental and physical state information which is very important for individuals will be the target of attackers and will be used as a weapon against them. Of course in such a scenario, it will be very crucial for people to manage their emotional, mental and physical state and this will affect the society in various ways which we can not imagine. This report clearly shows that if we fail to secure the IoT networks, then the ubiquity and proliferation of IoT will not transform the future to smarter but will cause catastrophic effects on human life, environment, culture and society.

Securing IoT networks is not an easy problem since we have to think of device, network and application characteristics, affordable cryptography-based solutions, physical security of the network and devices, compromise scenarios, intrusion detection systems. Designers and administrators will face many trade-offs, where security will be on one side and cost, network lifetime, Quality-of-Service (QoS), reliability and many more will be on the other side. When we are considering all of these dimensions, we should not avoid the user side. We have to bear in mind that users may not be security-aware. We also have to pay attention to propose user-friendly solutions which consider the usability and the user experience. If our solutions in the services that we provide are not satisfactory, then our efforts will be in vain, making the attackers' job easier.

The goal of this study is to present researchers a comprehensive investigation of IoT security for reliable future IoS. In order to be comprehensive, we analyzed the majority of the digital libraries (i.e., IEEE, ACM, Web of Science, Springerlink, Google Scholar) for quality conference, journal and magazine proposals. Studies published between 2008 and 2017 were included in this work where seventeen studies were analyzed to examine the D/DoS attacks for IoT networks and twenty-six studies were evaluated which either analyze the effects of the attacks, or propose a mitigation or a detection system against such attacks.

The remaining sections of this work are organized as follows: In Sect. 2, we briefly explain the related works. Section 3 explores the characteristics of IoT environments with devices, networks and applications. Section 4 considers Internet of Things security extensively. In Sect. 5, we examine D/DoS attacks for IoT. Section 6 consists of studies which analyze the effects of the D/DoS attacks for IoT networks. In Sect. 7, we examine the mitigation systems against D/DoS attacks, as well as security solutions for specific protocols. Section 8 is on the intrusion detection systems proposed for IoT, where we analyze several proposals from various points of views. In Sect. 9, we discuss the open problems and issues in IoT security and aim to provide new research directions. Finally Sect. 10 concludes this study.

## 2 Related Works

The Internet of Things is one of the most active topic of research nowadays. There are several surveys which address the security of IoT, attacks, countermeasures and Intrusion Detection Systems for IoT.

Zarpelao et al. [60] proposed a taxonomy of IDSes based on the placement approaches, detection methods and validation strategies. In their work, the authors



point out that IoT has unique characteristics, which will bring unique threats and novel requirements for IDSes. According to their findings, IDSes proposed for IoT need to address more attacks, more communication technologies and more protocols. They also indicated that IDS traffic should be managed securely and IDS designs should pay attention to the privacy of the host.

Adat et al. [5] proposed a literature review on the security of IoT where history of IoT security, taxonomy of security challenges and requirements, cryptography-based defense mechanisms and IDSes were evaluated. The authors suggested readers to research lightweight authentication schemes, to target 6LoWPAN and RPL security and to consider the resource limitations of IoT devices.

Samaila et al. [46] proposed an extensive analysis of security challenges of IoT. In this study the authors considered several issues including implementation of security in IoT, resource limitations, heterogeneity of IoT environments, applications and devices, security awareness of the users and maintenance of security after deployment.

Yang et al. [58] studied security and privacy issues in IoT. Their work considered the limitations of IoT environments which affect the security and privacy. They provided a classification of the attacks based on the layers of an IoT architecture and analyzed the cryptography-based security solutions for IoT networks in depth.

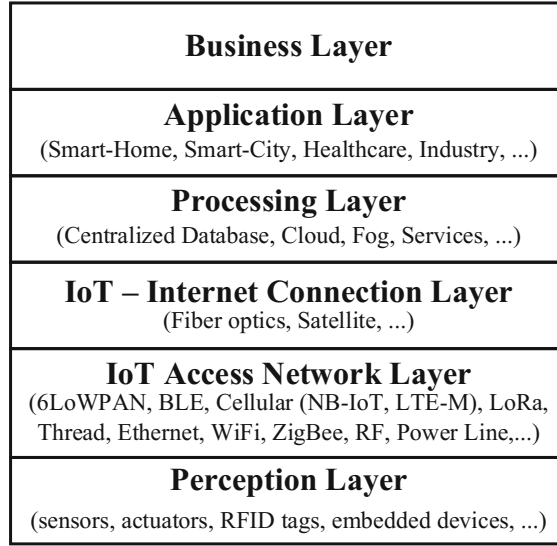
In this study, we aimed to provide a comprehensive view on security of IoT for reliable IoS. Although there are some topics of interest and points of view in common with the previous reviews, our work tries to depict a more complete picture of security of IoT.

### 3 Internet of Things

Internet of Things can be defined in several ways from various angles and there is no standard definition for it. However, from the engineering point of view, IoT is a network of any *things*, each supplied with a computing system (i.e., CPU, memory, power source and a communication interface like radio or Ethernet), each is uniquely identifiable and addressable and connected to the Internet. In this section, we will firstly propose a generic architecture for IoT which we think will be helpful to understand IoT environments better. After that, we will summarize the standardized protocol stack [37] we focus on in this study.

#### 3.1 Internet of Things Architecture

We believe that, exploring the architectural components is a very useful way to see the complete picture and understand IoT environments better. In Fig. 1, we outline a generic IoT architecture which is based on the general architectures previously proposed in [24, 57, 59]. The only difference of our architecture from the reference works is that we separated the IoT Access Network Layer from the IoT-Internet Connection Layer, whereas the reference studies combine them into a single layer called either as *Network Layer* or *Transport Layer*.



**Fig. 1.** Generic architecture of IoT

In the generic architecture, the lowest layer is the Perception Layer. It consists of sensors, actuators, RFID tags and any other embedded devices. Most of these devices are expected to be small form-factor devices with constrained resources (i.e., power source, processing, storage and communication interface). The majority of IoT devices will use battery as the power source. However, based on the application environment, mains-powered devices or energy-harvesting elements may exist as well. Since power will be a scarce resource, power consumption of the nodes (i.e., devices in the network) has to be minimized. In addition to various techniques to reduce the power consumption, IoT devices use low-power radios to keep the energy footprint as small as possible and lengthen the network lifetime. Typically low-end microcontrollers with RAM and ROM in the order of KBs constitute the big portion of nodes accommodated in IoT networks. In addition to the resource characteristics, mobility of the devices is important as well. Devices in the Perception Layer can be either static or mobile, but the percentage of mobile devices will be smaller than the static ones.

The IoT Access Network Layer is the second layer in our architecture, in which the nodes in the Perception Layer form a network. In this layer, there are several communication technologies (i.e., 6LoWPAN, Bluetooth Low Energy (BLE), LoRa and LoRaWAN, WiFi, Ethernet, Cellular, ZigBee, RF and Thread) which are candidates for the in-network communication. Most of them are open technologies, whereas some of them are (e.g., ZigBee, LoRa, Cellular) proprietary. These communication technologies provide varying data rates and transmission ranges in return of different power consumptions and costs. Hence, depending on the several design constraints, the nodes in the Perception Layer

can form IoT networks with different characteristics. Among these technologies, BLE, WiFi, LoRa and Cellular offer star-based topologies. However, 6LoWPAN, ZigBee and Thread support mesh topologies, where elements of the network can forward others' packets. Some of them are proposed for specific application areas (i.e., Thread was proposed for smart-home environments). Most of these technologies require a gateway or border router which is used to connect the nodes in IoT network to the Internet.

The third layer in our generic architecture is the IoT - Internet Connection Layer, where a border router or gateway connects the inner IoT network to the Internet via communication technologies, such as fiber optics or satellite communication.

Processing, analysis and storage of the collected data are performed at the Processing Layer. Designers can choose centralized storage and processing systems, or distributed storage and processing systems (e.g., cloud or fog computing environments). Middleware services are provided in this layer based on the processed and analyzed data. This is one of the most important layer in the architecture of IoT, since valuable information is extracted here from the collected data which can be in big volumes, variety and veracity.

The Application Layer is the fifth layer within the generic IoT architecture. In this layer, we see applications in various deployment areas, which make use of the meaningful information obtained from Processing Layer. Applications of IoT can be in home, building, industry, urban or rural environments. Applications of home environments can be health-reporting and monitoring, alarm systems, lighting applications, energy conservation, remote video surveillance [13]. Building environments IoT applications can be Heating Ventilation and Air Conditioning (HVAC) applications, lighting, security and alarm systems, smoke and fire monitoring and elevator applications [31]. Industrial IoT applications can be safety, control and monitoring applications with different emergency classes [38]. In urban environments, there may be broad range of applications. Lighting applications, waste monitoring, intelligent transportation system applications, monitoring and alert reporting are only a few of them. Rural environments may include monitoring applications (e.g., bridges, forests, agriculture, etc.).

The Business Layer is the last layer in the generic architecture, which includes organization and management of IoT networks. Business and profit models are constructed here in addition to charging and management operations [57].

### 3.2 Standardized Protocol Stack for Low Power and Lossy Networks

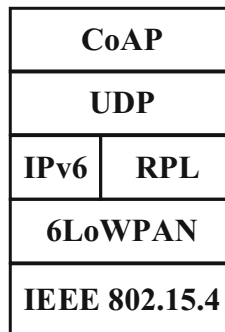
Multiple communication technologies exist for the IoT-Access Network Layer as we mentioned in Sect. 3.1. Since Thread, NB-IoT, LTE-M, LoRa/LoRaWAN are very new communication technologies, there were not any studies which focus on the D/DoS attacks that may target such networks during the time we were working on this proposal. ZigBee is a proprietary technology and it also uses the same physical and MAC layers as 6LoWPAN-based networks. Thus PHY and MAC layer attacks for 6-LoWPAN-based networks covers the PHY and MAC

layer attacks for ZigBee-based networks too. WiFi targets more resource-rich devices than 6LoWPAN. Therefore, it may not be a good candidate for low power and lossy networks-based IoT applications where majority of the devices will be battery-powered devices with small form factors and reasonable costs. Bluetooth Low Energy technology might be a good option for the IoT-Access Network Layer with very low power consumption, increased data rate and range. However, it suffers from the scalability problem where Bluetooth-based networks face with issues when the number of slaves exceeds seven [23,25]. Up until Bluetooth 5, park state was supported by Bluetooth which was allowing more than seven slaves to be part of the Bluetooth network in turns. But Bluetooth 5 does not support it any more and instead it brought scatternets, which aims to create multi-hop Bluetooth networks with specific nodes acting as routers between piconets. However, currently no commercial Bluetooth radio supports it and synchronization and routing operations will make the scatternet operation in Bluetooth networks a complex issue to deal with. Hence, considering the aforementioned reasons, we focus on the 6LoWPAN-based IoT networks in this study.

IEEE and IETF proposed several standards and protocols in order to connect resource-constrained nodes to the Internet within the concept of IoT. Palattella et al. [37] proposed a protocol stack for low power and lossy IoT networks which makes use of the protocols/standards proposed by IEEE and IETF. The standardized protocol stack is shown in Fig. 2.

The standardized protocol stack includes IEEE 802.15.4 [1] for physical layer and MAC layers. This standard promises energy-efficient PHY and MAC operations for low power and lossy networks and is also used by Thread and ZigBee technologies.

The expected cardinality of the IoT networks (e.g., of the order of billions) and already exhausted IPv4 address space force IoT to use IPv6 addresses. However, when IPv6 was proposed, low power and lossy networks were not considered, which resulted in the incompatible packet size issue. The maximum transmission unit of IEEE 802.15.4-based networks is far too small compared to IPv6 packet sizes. In order to solve this problem, IETF proposed an adaptation layer,



**Fig. 2.** Standardized protocol stack

IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN) [18]. 6LoWPAN makes use of header compressions to permit transmission of IEEE 802.15.4 fragments carrying IPv6 packets.

RPL [56] was proposed by the IETF as IPv6 routing protocol for low power and lossy networks. Formation of IEEE 802.15.4-based mesh networks was made possible by the RPL routing protocol, which constructs Destination Oriented Directed Acyclic Graphs (DODAG). A DODAG root creates a new RPL instance and lets other nodes to join the network by means of control messages. There are four types of control messages, which are DODAG Information Solicitation (DIS), DODAG Information Object (DIO), Destination Advertisement Object (DAO) and DAO-Acknowledgment (DAO-ACK) messages. DIS messages are broadcasted by new nodes to obtain the information about the RPL instance in order to join the network. Neighbor nodes reply with DIO messages which carry information about the RPL network (i.e., DODAG ID, instance ID, rank, version number, mode of operation, etc.) and their position in the network. The position of a node, which is the relative distance of a node from the DODAG root is named as *rank*. Rank is carried in DIO messages and it is calculated by each node based on the Objective Function (OF) and the rank of neighbor nodes. OF types, include, but are not limited to, hop count, expected transmission count, remaining energy. RPL lets network administrators to select a suitable OF based on the QoS requirements. When a node receives DIO messages from its neighbors, it calculates its rank and informs its neighbors about its rank with a new DIO message. Based on the rank of its neighbors, it selects the one with the lowest rank value as a preferred parent and informs that node with a DAO message. The receiving node replies with a DAO acknowledgment message and thus a parent-child relationship is set up. An example RPL network is shown in Fig. 3.

In RPL upward routes (i.e., the routes towards the DODAG root) are created by means of DIO messages, whereas downward routes are created by DAO messages. In order to minimize the overhead of control messages, RPL uses Trickle

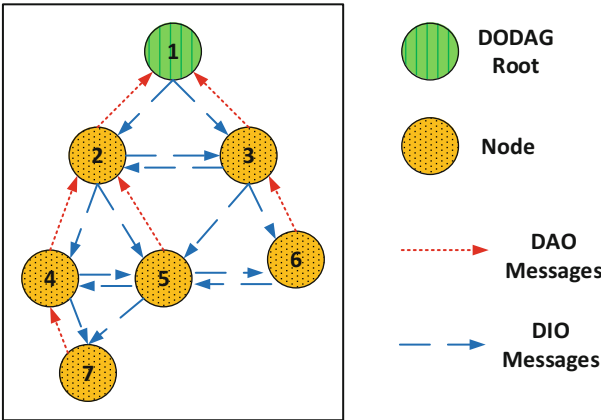


Fig. 3. An example RPL network

Timer [30] to reduce the number of control messages created as network gets more stable. Nodes are expected to follow the rules of the RPL specification in order to create loop-free and efficient RPL DODAGs. In low power and lossy networks, faults and problems tend to occur. To recover from such issues, RPL accommodates repair mechanisms (i.e., global repair and local repair).

The standardized protocol stack for low power and lossy networks employs the Constrained Application Protocol (CoAP) [50] for the application layer. CoAP is built on top of UDP and supports Representational State Transfer (REST) architecture. By means of CoAP, even resource-constrained nodes can be part of the World Wide Web (WWW). In order to optimize the data carried by CoAP messages, the IETF proposed another standard for the binary representation of the structured data called Concise Binary Object Representation (CBOR) [12] on top of CoAP.

## 4 Internet of Things Security

In Sect. 3.2, we briefly summarized the standardized protocol stack which consists of standards and protocols proposed by IEEE and IETF for low power and lossy networks. In this section, we focus on the security of IoT networks which accommodate the standardized protocol stack.

Securing a communication network is not an easy task and requires a comprehensive approach. In such a study, we have to determine assets, think of threats and consider compromise scenarios and possible vulnerabilities. Following these, we have to find the suitable solutions which will help us to ensure a *secure* system. When we think of the solutions, the first thing that probably comes into our minds is the cryptography. Cryptography promises to provide *confidentiality* and *integrity* of the messages, *authentication* of the users and systems and *non-repudiation* of the transactions. Confidentiality means that the content of the message is kept secret from eavesdroppers. Integrity ensures that the content of the message is not changed and is still the same as the first time it was produced. Authentication allows the end points of the communicating parties to identify each other and determine the correct target of the communication. Non-repudiation prevents one end of the communication to deny its actions that it performs and protects the other end.

In this section, we firstly outline the cryptography-based security solutions for the low power and lossy networks which employ the standardized protocol stack. After that, we analyze the protocols and point out the advantages and disadvantages. Then we will inquire whether cryptography is enough for us or not.

### 4.1 Cryptography-Based Security Solutions for Low Power and Lossy Networks

A number of cryptography-based solutions exist so as to secure the low power and lossy networks that employ the standardized protocol stack. These solutions are shown in Fig. 4.

<b>CoAP → CoAPs</b>
<b>UDP → DTLS</b>
<b>RPL &amp; IPv6 → IPSec, Secure RPL Control Messages</b>
<b>6LoWPAN →</b>
<b>IEEE 802.15.4 → IEEE 802.15.4 PHY &amp; Link Layer Security</b>

**Fig. 4.** Cryptography-based security solutions for low power and lossy networks

IEEE 802.15.4 PHY and Link Layer Security [1] provides security for the communication between two neighbors in IEEE 802.15.4-based networks. This hop-by-hop security solution promises confidentiality, authenticity and integrity against insider attackers.

The Internet Protocol Security (IPSec) [22] aims to provide end-to-end security. It consists of a set of protocols, which are Authentication Headers (AH), Encapsulating Security Payloads (ESP) and Security Associations (SA). AH provides authentication and integrity, whereas ESP promises confidentiality in addition to authentication and integrity. Designers can select either of them but regardless of the selection, SA has to run initially to setup the security parameters. IPSec provides security for IP-based protocols and it is independent from the protocols above the network layer.

In addition to IPSec, RPL provides secure versions of the control messages. Although it is optional, confidentiality, integrity and authentication of the control messages are assured.

Datagram Transport Layer Security (DTLS) [44] aims to secure UDP-based applications. Similar to the other solutions, it ensures the confidentiality, integrity and authenticity of datagrams.

CoAP provides security bindings for DTLS in CoAPs scheme. It lets designers to choose to run DTLS with preshared keys, public keys and/or certificates in order to secure CoAP traffic. Although Fig. 4 does not show any other security mechanisms working at the application layer and above, the IETF has draft documents (i.e., Object Security of CoAP (OSCoAP), CBOR Object Signing and Encryption (COSE) and Ephemeral Diffie-Hellman over COSE (EDHOC)) which aim to provide security at the application layer and above.

## 4.2 Which Security Solution to Use?

As we can see, there are a number of security solutions to protect low power and lossy networks and it is hard to determine which solution to use.

IEEE 802.15.4 PHY and Link Layer Security is independent from the network layer protocols and most of the radios support it. Independence from the

upper layer protocols means that we do not have to change anything with them. However, since IEEE 802.15.4 PHY and Link Layer Security provides the security between two neighbors, trustworthiness of every node on the routing path becomes a very crucial issue. If the routing path has a malicious node, then the security of the messages routed through this path cannot be guaranteed. In addition to this, IEEE 802.15.4 PHY and Link Layer Security works only in the IoT - Access Network Layer in our generic architecture and when messages leave this layer and enter the Internet, they are no more protected [40].

IPSec provides end-to-end security and is independent from the upper layer protocols. End-to-end security guarantees security between two hosts which can be in different networks. Designers do not have to worry about the trustworthiness of the other nodes, devices or networks on the path. However, it brings burden to 6LoWPAN layer, where packets with IPSec require header compression [40]. In addition to this, Security Associations is connection-oriented and simplex, which means if two hosts want to send packets secured with IPSec to each other, then each of them individually need to establish SAs [26]. Furthermore, firewalls may limit the packets with IPSec and Internet Service Providers (ISPs) tend to welcome packets with IPSec as business-class packets and prefer to charge them more. So, if IoT data will be secured with IPSec, there are a number of issues we have to consider before using it.

DTLS serves as the security solution between two UDP-based applications running on different end-points. Although it aims to protect the application layer data, it does not promise security for anything else. This means, if we employ DTLS as the only security solution, then we cannot protect IP headers when packets are passing through the IoT - Access Network Layer and through the Internet. So, security of the routing becomes susceptible to the attacks, such as DoS and DDoS attacks. This is why the primary security concern of DTLS is on D/DoS attacks.

### 4.3 We Have Cryptography-Based Solutions, Are We All Set?

In Sect. 4.1 we outlined the cryptography-based security solutions very briefly. As we explained in Sect. 4.2, each solution comes with its advantages and disadvantages. It is not an easy task to select the appropriate solution. However, there are a number of other issues which we have to consider when protecting IoT networks.

First of all, cryptography is generally thought to be heavy weight, and is full of resource consuming operations implemented in software and/or hardware. When we consider the resource limitations of the devices in low power and lossy networks, affordability of such solutions becomes questionable. Designers have to face the trade off between security and very crucial parameters such as cost, network life time and performance.

Secondly, although cryptography-based solutions are proved to be secure, proper implementation of the protocols and algorithms is extremely important. However, most of the implementations of these solutions have vulnerabilities as



reported by researchers [8]. In addition to this, in order to shorten the development time, engineers tend to use the code examples shared on forums. These code examples working properly does not mean that they are vulnerability-free [4].

Physical security of the networks and devices are as important as our other concerns. It is directly related to the applicable type of attacks. If the physical security of the deployment area is weak, which is the case for most of the deployments, and if devices do not have protection mechanisms against tampers which is due to reduce the cost, then it is possible for attackers to insert a malicious device or grab a device and extract the security parameters and leave a malicious device back.

In addition to the cost of cryptography, issues with correct implementation and physical security, we have to consider users as well. We know that most of the people are not security-aware and usability of security mechanisms have problems [47]. Therefore, compromise scenarios have to think of users and external people involving with the IoT network, applications and deployment areas.

Although we have cryptography, our networks and systems are still susceptible to some type of attacks, called Denial of Service attacks [54]. In the next section, we will examine the DoS and DDoS attacks which may target low power and lossy networks employing the standardized protocol stack.

## 5 Denial of Service Attacks Targeting Internet of Things Networks

Denial of Service attacks aim to misuse the available resources in a communication network and degrade or stop the services offered to ordinary users. Since

**Table 1.** D/DoS attacks which may target IoT networks

Physical layer	MAC layer	6LoWPAN layer	Network layer	Transport and application layer
Node Capture	Jamming	Fragment Dupl.	Rank	Flooding
Jamming	GTS	Buffer Reserv.	Version Number	Desynchronization
Spamming	Backoff Manip.		Local Repair	SYN Flood
	CCA Manip.		DODAG Inconsist.	Protocol Parsing
	Same Nonce		DIS	Processing URI
	Node Spec. Flooding		Neighbor	Proxying and Caching
	Replay Protection		Sybil	Risk of Amplification
	ACK Attack		Sinkhole	Cross-Protocol
	Man-in-the-Middle		Selective Forw.	IP Address Spoofing
	Ping-Pong Effect		Wormhole	
	Boostrapping		CloneID	
	Stenography			
	PANID Conflict			

IoT will be one of the main building block of Internet of Services, detection, mitigation and prevention of such attacks are very crucial.

In this section, we present and explain the D/DoS attacks which may target IoT networks. Table 1 categorizes such attacks with respect to the layers of the standardized protocol stack. This categorization is an extended version of our previous study [10].

### 5.1 D/DoS Attacks to the Physical Layer

Physical Layer D/DoS attacks are *node capture*, *jamming* and *spamming*.

As its name implies, in *node capture* attacks, attackers capture the physical nodes within the network. The aim of the attackers may be creating routing holes or tampering the device and extracting security parameters. After that, they may place the node back with the compromised software or place the node with a replica of it. By this way, they can apply various attacks (e.g., other attacks categorized as higher layer attacks).

Physical Layer *jamming* attacks comprise of malicious devices creating interference to the signals transmitted in the physical layer [7]. Attackers can constantly, randomly or selectively (i.e., jamming signals carrying specific packets, such as routing or data packets) apply jamming.

In *spamming* attack, attackers place malicious QR codes to the deployment areas which cause users to be forwarded to malicious targets on the Internet [42].

### 5.2 D/DoS Attacks to the MAC Layer

MAC Layer D/DoS attacks are *link layer jamming*, *GTS*, *backoff manipulation*, *CCA manipulation*, *same nonce attack*, *node specific flooding*, *replay protection attack*, *ACK attack*, *man-in-the-middle*, *ping-pong effect*, *bootstrapping attack*, *PANID conflict* and *stenography*.

*Link layer jamming* is a type of jamming where frames are jammed instead of signals as in the physical layer [7].

IEEE 802.15.4 standard has an optional feature called as Guaranteed Time Slot (GTS) which works in beacon-enabled operational mode. GTS is intended for timely critical applications that require strict timing with channel access and transmissions. Nodes have to request and allocate time slots in order to use this feature. However, if attackers cause interference during this process (e.g., by jamming), then ordinary nodes cannot register themselves for the guaranteed time slots and thus QoS of the application gets affected. This attack is called *GTS* attack [51].

*ACK* attack consists of attackers creating interference to Acknowledgment (ACK) frames and thus causing a node to believe that its fragment was not successfully received by the receiving node [7]. By this way targeted nodes are forced to retransmit the same fragment and consume more power. QoS of the running application would be affected by it too. Moreover, it may cause the sender node believe that its next hop neighbor is filtering the messages.

Clear Channel Assessment (CCA) mechanism is used by nodes to sense the channel and find out if any other node is currently using the channel or not. This approach is commonly used to prevent collisions. However, attackers can skip CCA and access the channel, which causes collisions. By this way, delays, retransmissions and unnecessary energy usage occurs. This attack is called *CCA manipulation* [7].

*Backoff manipulation* attack compromises the backoff periods of Carrier Sense Multiple Access (CSMA)-based medium access with attackers choosing shorter backoff times instead of longer [7]. By this way, they get the chance to use the channel as much as possible and limit the other users' channel accesses.

Sequence numbers are used in the IEEE 802.15.4 standard in order to prevent malicious devices sending the previously sent fragments over and over. However, in *replay protection attacks* [7], attackers can still misuse it by sending frames with bigger sequence number than the targeted ordinary node. This would cause the receiving node drop the fragments coming from the ordinary node since it now looks like it is sending old fragments.

As we mentioned in Sect. 4, IEEE 802.15.4 PHY and Link Layer Security is a candidate security mechanism for IoT security, which promises to protect the communication between two neighbor nodes. If nodes share the same key and nonce values in the implementation of IEEE 802.15.4 PHY and Link Layer Security, then attackers may extract the keys by eavesdropping the messages which happens in the *same nonce* attack [7].

The *PANID Conflict* attack misuses the conflict resolution procedure of IEEE 802.15.4 which functions when two coordinators are placed close to each other in a deployment area and holding the same Personal Area Network ID (PANID). Malicious nodes may transmit a conflict notification message when there is actually no conflict to force the coordinator to initiate the conflict resolution process [7].

Another MAC Layer D/DoS attack is the *ping-pong effect*, where malicious nodes intentionally switch between different PANs [7]. If attackers choose to do it frequently, then they may cause packet losses, delays and extra overhead to the already limited resources.

In the *bootstrapping attack*, attackers aim to obtain useful information about a new node joining the network. In order to do so, firstly a targeted node is forced to leave the network by the attackers. Then when it tries to join the network again, attackers obtain the bootstrapping information which they may use to associate a malicious node to the network [7].

*Node specific flooding* attacks are a type of flooding attack which is applied at the MAC layer [7]. In this attack, malicious nodes send unnecessary fragments to the target node which aims to consume its resources and thus is no longer able to serve for its ordinary purpose.

The *Stenography* attack abuses the unused fields in the frame format of IEEE 802.15.4. Unused bits can be used by the attackers to carry hidden information [7].

### 5.3 D/DoS Attacks to the 6LoWPAN Layer

Hummen et al. [19] proposed two attacks, namely *fragment duplication* and *buffer reservation*, which may target the 6LoWPAN Adaptation Layer.

In *fragment duplication* attack, attackers duplicate a single fragment of a packet and thus force the receiving node to drop the fragments of the corresponding packet. In this attack, attackers abuse the approach of 6LoWPAN standard which deals with the duplicate fragments. The standard advises to drop the fragments of a packet in case of duplicates so as to get rid of the overhead of dealing with duplicates and save resources. However, malicious nodes can turn this naive mechanism into a DoS attack very easily.

In *Buffer reservation* attacks, attackers reserve the buffer space of the targeted node with incomplete packets and keep it occupied as long as possible. Since resources are limited, nodes cannot afford to spare extra buffer space for the incomplete packets of other nodes. Thus, during the time the attacker holds the buffer space, ordinary nodes' fragments cannot be accepted. Readers should note that, this behavior of 6LoWPAN is possible when 6LoWPAN is configured to forward the fragments according to the route-over approach, where all fragments of a packet are reassembled by the receiving node before being forwarded.

### 5.4 D/DoS Attacks to the Network Layer

D/DoS attacks which may target the IoT Network Layer can be divided into two categories: RPL-specific and non-RPL-specific attacks. RPL-specific attacks are *rank*, *version number*, *local repair*, *DODAG inconsistency* and *DIS* attacks. Non-RPL-specific attacks are the ones which are already known from the wireless sensor networks, and other communication networks research. Although they look old-fashioned, they are still applicable in RPL-based networks. Non-RPL-specific attacks are *sybil*, *sinkhole*, *selective forwarding*, *wormhole*, *cloneID* and *neighbor* attacks.

**RPL-Specific Attacks.** D/DoS attacks which may target RPL networks abuse the vulnerabilities of the RPL protocol design. RPL, designed by the IETF for the routing of IPv6 packets on low power and lossy networks, has vulnerabilities with the control plane security and attackers can easily misuse it. In order to secure RPL networks, the IETF advises to use cryptography-based security solutions, secure control messages and some attack-specific countermeasures (e.g., using location information, multi-path routing) [52]. However, as explained in Sect. 4.3, there are several issues to consider with security and it is highly probable that RPL-based networks will be susceptible to D/DoS attacks.

*Rank* is a very crucial parameter of the RPL protocol which represents a node's position within the DODAG. This position is a relative distance of a node from the DODAG root. The distance is determined with respect to the Objective Function and can be based on the hop count, link quality, remaining power etc. Rank is used to create an efficient DODAG according to the application needs and to set up the child-parent relationship. For optimized and loop-free

DODAGs, nodes have to follow the rules. However malicious nodes may use rank in various ways to apply D/DoS attacks. In [27] and [28], an attacker node selects the neighbor with worst rank as a preferred parent instead of choosing the one with the best rank. Thus an inefficient DODAG is created which causes delays and an increased number of control messages. In [29], the attacker intentionally skips applying the rank check which breaks the rank rule constructing the loop-free parent-child relationship.

RPL has two repair mechanisms in order to keep the DODAG healthy. One of them is the global repair operation where the entire DODAG is re-created. According to the RPL specification, only the DODAG root can initiate the global repair mechanism by incrementing the *Version Number* parameter. Every DODAG has a corresponding version number that is carried in DIO messages. When the root increments the version number, nodes in the RPL network find out the global repair operation by checking the version number in the incoming DIO messages. They exchange control messages and setup the new DODAG. However, there is no mechanism in RPL which guarantees that only the DODAG root can change the version number field. Malicious nodes can change the version number and force the entire network to set up the DODAG from scratch [11, 35]. This attack is called *Version Number* attack and it affects the network with unnecessary control messages, delays, packet losses and reduced network lifetime.

Similar to the global repair, the local repair mechanism of RPL can be the target of a D/DoS attack called *local repair* [27, 29]. Local repair is an alternative repair solution of RPL which aims to solve the local inconsistencies and issues and cost less than the global repair mechanism since it involves a smaller portion of the network. If nodes find out inconsistencies (e.g., loops, packets with wrong direction indicators), then they start the local repair mechanism which consists of exchanging control messages and re-creating the parent-child relationships and getting appropriate ranks again. However malicious nodes can start local repair when there is no need so as to misuse the resources. This type of attack is called *local repair* attack.

RPL has a data path validation mechanism, in which headers of the IPv6 data packets carry RPL flags that indicate the direction of the packet and possible inconsistencies with the rank of the previous sender/forwarder. When a node receives a packet with those flags indicating an inconsistency, it drops the packet and starts the local repair mechanism. In *DODAG inconsistency* attacks [49], attackers can set the corresponding flags of a data packet before they forward it and force the receiver node to drop the packet and start local repair.

The last D/DoS attack specific to RPL is the *DIS* attack. DIS messages are used in RPL when a new node wants to join the network and therefore asking for information about the RPL network. Attackers can send unnecessary DIS messages in *DIS* attacks [27], which causes the neighboring nodes to reset their DIO timers and send DIO messages frequently. Thus, the attacker forces nodes to generate redundant control messages and consume more power.

**Attacks not Specific to RPL.** Attacks which are not specific but still applicable to RPL are *neighbor*, *sybil*, *sinkhole*, *selective forwarding*, *wormhole* and *cloneID*.

A malicious node can apply the *neighbor* attack by retransmitting the routing control messages it hears [27]. This behavior causes neighbor nodes to think that the source of the control message is close to them and take actions accordingly. Actions could be sending control messages back, trying to select it as a preferred parent, etc. If the attacker uses a high power radio, then it may affect a large portion of the network by this way.

In *sybil* attacks, a malicious node seems to act as multiple nodes, introducing itself with multiple logical identities [54]. If there is a voting mechanism running in the IoT network (e.g., voting based security mechanisms, cluster head selection), attackers can apply sybil to change the results and thus take control of the complete network or a portion of the network.

The *CloneID* attack is similar to the sybil attack but works in a different dimension. The attacker in this case places the clones of a malicious node or normal node to the multiple positions at the network [41, 54]. This attack has similar aims as sybil and it may also be called *node replication* attack.

*Sinkhole* attacks are another type of attacks where malicious nodes advertise good routing parameters to show themselves as candidate parents. In RPL, attackers can advertise good ranks, which causes the neighbor nodes to select it as the preferred parent [41, 54, 55]. When a malicious node is selected as the preferred parent by neighbor nodes, then it can apply other attacks, such as selective forwarding.

In *selective forwarding* attacks, a malicious node inspects the incoming packets, drops the ones it is interested in and forwards the rest [41, 54]. For example, it may forward only the routing messages, whereas it may drop the data packets. Or, malicious node may filter specific packets sourced from or destined to specific addresses.

The last attack we explore in this category is the *wormhole* attack [39, 54]. In wormhole attacks, at least a couple of malicious nodes create a hidden communication channel by means of multiple radios and transfer the overheard messages transmitted at one end point to another. This may work bidirectional as well. By this way, two sets of nodes around each attacker believe that they are in the communication range of each other, which causes several issues.

## 5.5 D/DoS Attacks to the Transport and Application Layer

D/DoS attacks which may target Transport and Application Layers are *flooding*, *desynchronization*, *SYN flood*, *protocol parsing*, *processing URI*, *proxying and caching*, *risk of amplification*, *cross-protocol* and *IP address spoofing* attacks [21, 50]. The majority of the attacks mentioned here were not studied in the literature and the IETF considers them as possible threats for CoAP.

## 6 Studies that Analyze D/DoS Attacks for Internet of Things

The previous section was about the possible D/DoS attacks which can target the IoT networks. Starting from this section, we will analyze the studies for the aforementioned attacks. In this section, we will explore the works which investigate the effects of the attacks.

Sokullu et al. [51] proposed GTS attacks to IEEE 802.15.4 in 2008. In their work, they also analyzed the effects of the attack in the bandwidth utilization of Contention Free Period (CFP). They considered single and multiple attackers where attackers can either attack randomly or intelligently. They found out significant decrease in the bandwidth utilization of CFP periods due to GTS attacks.

Le et al. analyzed the rank attack in [28] in RPL networks in 2013. In this work, they applied the rank attacks with different cases where the attacker constantly applies the attack or switches between legitimate and malicious behaviors frequently. Analysis with respect to combinations of attacking cases show that if the rank attack is applied in a dense part of the network, then its effect is more detrimental. They also realized that, the number of affected nodes, number of generated DIO messages, average end-to-end delay and delivery ratio can be the indicator of such attacks.

Mayzaud et al. studied RPL version number attacks in [35] in 2014. They investigation with a single attacker in a grid topology at varying positions showed that the location of the attacker is correlated to the effects of the attack. If the attacker is located far from the DODAG root within the grid, then its effect is larger than when attacker is closer to the root.

The Version number attack is analyzed by another work [11] proposed by Aris et al. in 2016. In this study, the authors considered a factory environment consisting of varying topologies (i.e., grid and random) with different node mobilities (e.g., static and mobile nodes). A probabilistic attacker model is incorporated here. Based on the simulations, in addition to the location-effect correlation found in Mayzaud's work [35], the authors found out that the mobile attackers'

**Table 2.** Categorization of the studies that analyze the D/DoS attacks for IoT

Proposal	Target attack	Finding
Sokullu [51]	GTS (MAC Layer)	Significant bandwidth utilization decrease in CFP
Le [28]	Rank (Routing)	Dense networks are more vulnerable
Mayzaud [35]	Version Number (Routing)	<i>Attacking position-effect of the attack correlation</i>
Aris [11]	Version Number (Routing)	Mobile attackers are more detrimental and attack triples the power consumption of the network

effect can be as detrimental as the farthest attacking position in the network. They also showed that, version number attacks can increase the power consumption of the nodes by more than a factor of two.

Table 2 categorizes the studies which analyze the effects of the D/DoS attacks for IoT. When we review the studies in this section, we realize that researchers focused on the IoT-specific attacks rather than the attacks which we are already familiar with from the Wireless Sensor Networks research (i.e., selective forwarding, wormhole, sinkhole, etc.). In addition to this, three of the studies found out correlations with the success of the attack and the attack settings. Such findings can be extremely useful in defending IoT networks against the attackers and designing better detection and mitigation systems which consider these findings. In Table 1, we had provided a categorization of the D/DoS attacks for IoT and it is clear that many attacks have not been implemented and analyzed in a similar manner.

## 7 Mitigation Systems and Protocol Security Solutions for Internet of Things

Mitigation systems are proposed by researchers in order to minimize the effects of the attacks. Such systems are far from being a complete security solution but still can increase the strength of the system against attackers. In this context, existing protocols are enriched with additional features by the designers which can mitigate the detrimental effects of the D/DoS attacks. On the other hand, protocol security solutions referred here consist of mechanisms which aim to secure a communication protocol or a specific part of it. In this section, readers can find the studies which either propose a security solution or mitigate the effect of the attacks.

Dvir et al. proposed VeRA [16], a security solution for the crucial version number and rank parameters carried in DIO messages in 2011. Their solution makes use of hash chains and message authentication codes in order to securely exchange these RPL parameters in DIO messages.

Weekly et al. [55] evaluated the defense techniques for sinkhole attacks in RPL in 2012. They compared a reduced implementation of VeRA to their novel technique called as Parent Failover. Parent Failover uses Unheard Node Set which includes the IDs of the nodes that the BR did not hear from. Each node blacklists its parent if it sees itself in the list in this technique.

Wallgren et al. [54] proposed implementations of routing attacks (i.e., selective forwarding, sinkhole, hello flood, wormhole, sybil) which are not specific to RPL. They did not analyze the effects of the attacks. However, they made comments on possible mitigation/detection mechanisms against such attacks. Their mitigation ideas include usage of geographical location information, incorporation of cryptography schemes, using multiple routes and/or RPL instances and keeping track of the number of nodes within the network. Although the authors suggest to use such mechanisms against the corresponding attacks, they did not implement the mitigation mechanisms and analyze the performance of them.



Hummen et al. [19] proposed two novel attacks to 6LoWPAN adaptation layer, which are fragment duplication attack and buffer reservation attack. They also proposed two novel mitigation mechanisms against these attacks. For fragment duplication attacks, the authors proposed hash chain structures which create a binding for fragments of a packet to the first fragment of the corresponding packet. In order to mitigate the effects of buffer reservation attacks, they suggested to split the reassembly buffer into fragment-sized slots and let multiple fragments belonging to different packets use it. They merged split buffer approach with a fragment discard mechanism in case of overloaded buffer conditions.

In 2014, Sehgal et al. [49] proposed a mitigation study which targets DODAG inconsistency attacks. According to the authors, RPL uses a threshold to mitigate the effects of such an attack. In RPL, a node receiving a data packet with flags indicating an inconsistency drops the packet and resets its trickle timer. A node can do this until reaching a threshold. After this threshold it does not reset the trickle timer any more. This proposal changes the constant threshold of RPL to an adaptive threshold to mitigate the effects of the attack better.

Another mitigation technique for DODAG inconsistency attacks was proposed by Mayzaud et al. [33] in 2015. It is an improved version of the mitigation technique proposed in Sehgal's work [49]. In the former study, packets with 'R' flags set were counted, whereas in this study, the number of trickle timer resets are counted. Based on this, a node either drops the packets and resets trickle timer, or forwards the packets with modifying the R and O flags to the normal state.

**Table 3.** Categorization of the mitigation systems and protocol security solutions

Proposal	Target attack	Mitigation/Security mechanism
VeRA [16]	Rank and Version Number (Routing)	Hash chains and Message Authentication Codes
Weekly [55]	Sinkhole (Routing)	Reduced VeRA and Unheard Node Set
Wallgren [54]	Routing attacks not specific to RPL	Geographical Location Info., Cryptography, Multiple Paths and Instances, Cardinality of the Network
Hummen [19]	Fragment Duplication and Buffer Reservation (MAC)	Content Chaining Using Hash Chains, Split Buffer with Fragment Discard
Sehgal [49]	DODAG Inconsistency (Routing)	Adaptive Threshold for Inconsistency Situations
Mayzaud [33]	DODAG Inconsistency (Routing)	Adaptive Threshold for Inconsistency Situations
Ramani [43]	CloneID (Routing), General DoS	Distributed Firewall

In 2016, Ramani proposed a two-way firewall [43] for low power and lossy networks. The two-way firewall analyzes the traffic destined to the 6LoWPAN network and traffic leaving from the network. The proposed firewall was tested against the CloneID and simple DoS attacks. The main module of the firewall works on the BR and becomes active when packets destined to the CoAP and DTLS ports are captured. Packets are parsed into incoming and outgoing packets and their IP addresses and ports are verified. After this check, information related to the packet is saved and checked against the protocol rules. Erroneous packets are dropped here. Also the nodes in the 6LoWPAN network are equipped with mini-firewall modules which inform the main firewall about their behavior.

Table 3 categorizes the Mitigation Systems and Protocol Security Solutions for IoT. Mitigation mechanisms against routing attacks constitute the majority of the studies in this section. Researchers targeted both RPL-specific attacks and other routing attacks which can be applied to RPL as well. Considering the resource limitations in IoT networks, we can see that three proposals use hash functions as lightweight solutions.

## 8 Intrusion Detection Systems for Internet of Things

In this section, we will survey the literature for intrusion detection systems proposed against D/DoS attacks for IoT networks. This section is organized as follows: Firstly, we will briefly give some background information about Intrusion Detection Systems (IDS). After that, we will analyze the IDSes proposed for IoT.

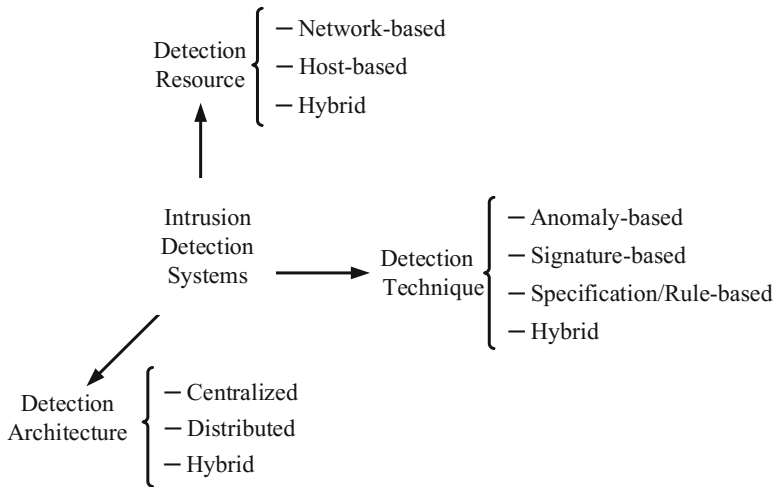
### 8.1 Intrusion Detection Systems

Intrusion Detection Systems serve as a strong line of defense for computer networks against the attackers. Without IDS, the puzzle of a *secure* network is incomplete. As explained in Sect. 4.3, despite having cryptography-based solutions, attacks are still possible and IDS comes into the picture here, where it monitors and analyzes the traffic, data, behavior or resources and tries to protect the network from attackers.

Intrusion Detection Systems can be explored from various points of view. Figure 5 shows a 3D Cartesian Plane of IDSes, where axes depict important categories which may be helpful to classify the IDSes. Although not shown, there may be other dimensions in this figure, such as operation frequency and targeted attacks.

**Intrusion Detection Systems: Detection Techniques.** IDSes can be divided into four classes based on the detection technique. These are *anomaly-based*, *signature-based*, *specification/rule-based* and *hybrid* systems where the former two are the most popular ones.

*Anomaly-based* systems learn the behavior of the system when there is no attack and create a *profile*. Deviations from the profile show possible anomalies. Anomaly-based IDSes can detect the new attacks since attacks are expected



**Fig. 5.** Intrusion detection systems

to cause deviations from the ordinary behavior. However, they can create false alarms and incorrectly classify legitimate connections as intrusion attempts. In addition to this, anomaly-based techniques are generally believed to be more complex and to use more resources than the other detection techniques.

*Signature-based* systems aim to detect intrusions by making use of attack signatures/patterns. Typically signatures are stored in a database and IDS tries to match them when analyzing the connections, packets or resources. If the database does not have a signature for an attack, which happens in case of new attacks, such systems cannot detect it. Otherwise they promise high detection rates for the known attacks and they do not suffer from false alarms. If we use signature-based techniques, we have to consider how to deal with new attacks since our IDS will probably skip them. Also we have to think about the storage cost of the signatures.

*Specification/rule based* systems require specifications of the protocols/systems and create rules based on the specifications. These rules separate legitimate connections from the malicious ones. In such systems creation of the specification and coverage of the created rules are important issues which affect the performance of the IDS.

*Hybrid* intrusion detection systems consider advantages and disadvantages of the previous three detection techniques and aim to benefit from multiple of them at the same time. Of course such a decision may be costly in terms of the available resources.

**Intrusion Detection Systems: Detection Resources.** Intrusion Detection Systems can be divided into three categories in terms of the resources they use for detection. These are *network-based*, *host-based* and *hybrid* detection systems.

*Network-based* IDSes use the incoming and outgoing monitoring traffic to/from the network in addition to the internal traffic to detect the intrusions. Network-based IDSes can have a global view of the network and use it to boost the detection performance. However, such systems lack the information about the individual resource consumptions and logs of the nodes within the network which may be crucial for the detection of specific attacks.

*Host-based* intrusion detection systems consider the traffic only coming to and leaving from the host. Such systems monitor the resources and logs of the hosts as well which may provide hints about attacks. Since they work locally, they cannot have a global knowledge about the state of other nodes or the network which can be very useful to increase the performance of the IDS.

*Hybrid* IDSes combine the strengths of network-based and host-based systems that benefit from both network and node resources.

**Intrusion Detection Systems: Detection Architecture.** Architecture of Intrusion Detection Systems can be *centralized*, *distributed* or *hybrid*.

*Centralized* IDSes place the intrusion detection to a central location and all of the monitoring information has to be collected here. One of the main reasons to select a central point for intrusion detection can be the available resources. As mentioned previously, anomaly detection techniques can be resource-hungry and it may not be feasible to place them on every node due to resource-constraints. Therefore, a resource-rich node, such as border router, can accommodate the intrusion detection system. However, centralized systems come along with communication overhead since monitoring data has to be carried all the way to the central location. If malicious nodes prevent monitoring data from reaching to the centralized IDS, then they may achieve to mislead the IDS.

In *Distributed* IDSes, intrusion detection runs locally at every node in the network. In order to afford an IDS at every node, designers have to tailor the detection technique or algorithm according to the available resources. This approach clearly does not have any communication overhead, however the IDS has only local information to analyze in order to detect the intrusions.

*Hybrid* IDSes again harmonize both of the detection architectures and try to benefit from each of them as much as possible. In such systems, IDS is divided into modules and these modules are distributed along the network. Modules at every node can apply intrusion detection to a certain extend, may share less information (in comparison with centralized IDSes) with the centralized module and thus both reduce the communication overhead and enjoy the rich resources of the centralized module.

## 8.2 Intrusion Detection Systems Proposed for Internet of Things

Cho et al. [15] proposed a botnet detection mechanism for 6LoWPAN-based networks in 2009. They assumed that the nodes in the IoT network use TCP transport layer protocol. Nearly seven years before the Mirai botnet, this study considered how IoT networks can be used as a botnet for DDoS attacks.

The authors thought that, if there exists a malicious node on the forwarding path, then it can forge the packets and direct them to the target victim address based on the command of the bot master. A detection mechanism is placed at the 6LoWPAN gateway node which analyzes the TCP control fields, average packet lengths and number of connections to detect the botnets. The idea is based on hypothesis that the ordinary IoT traffic should be very homogeneous and botnet would cause significant deviations on the traffic.

Le et al. proposed a specification-based IDS [29] for IoT in 2011. It targets rank and local repair attacks. Their work assumes that a monitoring network is set up at the start of the network with minimum number of trustful monitoring nodes which has full coverage of the RPL network and has capability to do additional monitoring jobs. In this context, it has a distributed architecture and it is a network-based IDS. Each Monitoring Node stores the IDs, ranks and preferred parents of neighboring nodes. MNs accommodate a Finite State Machine (FSM) of RPL with normal and anomaly states to detect the attacks. If a MN cannot decide whether a node is an attacker or not, then it can ask the other MNs. This IDS was not implemented and the authors did not specify the format of the communication between the monitoring nodes.

Misra et al. [36] proposed a learning automata based IDS for DDoS attacks in IoT. When there is an attack taking place, packets belonging to the malicious entities need to be sampled and dropped. This study aims to optimize this sampling rate by means of Learning Automata (LA). Firstly DDoS attacks are detected at each IoT node in the network based on the serving capacity thresholds. When the source of the attack is identified, all of the nodes are informed about it. In the next step, each node samples the attack packets and drops them. This is when the LA solution comes to the scene. Sampling rate of the attack packets are optimized by means of the LA.

SVELTE IDS [41] was proposed by Raza et al. in 2013. It targets sinkhole and selective forwarding attacks. It has a hybrid architecture where it places lightweight IDS modules (i.e., 6LoWPAN mapper client and mini firewall module) at the resource-constrained devices and the main IDS (i.e., 6LoWPAN mapper, intrusion detection module and distributed mini firewall) at the resource-rich Border Router (BR). 6LoWPAN Mapper at the BR periodically sends requests to the mapper clients at nodes. Mapper clients reply with their ID, rank, their parent ID, IDs of neighbors and their ranks. Based on the collected information about the RPL DODAG, SVELTE compares ranks to find out inconsistencies. It also compares the elements of the white-list and elements of current RPL DODAG and uses nodes' message transmission times to find out the filtered nodes. The mini firewall module is used to filter outsider attackers. In addition to these, nodes change their parents with respect to packet losses encountered. In terms of the detection resources used, we can classify SVELTE into network-based IDSes. We can also put it into the category of specification/rule-based IDSes.

In the same year with SVELTE, Kasinathan et al. proposed a centralized and network-based IDS [21] and its demo [20] for 6LoWPAN-based IoT networks. The motivation of this study is the drawback of centralized IDSes which

suffer from internal attackers. As mentioned in Sect. 8.1, internal attackers may prevent monitoring data from reaching the centralized IDS. This is due to the fact that monitoring data is sent through the shared wireless medium, which can be interfered by attackers. In this IDS architecture, the authors place monitoring probes to the IoT network which have wired connections to the centralized module. Evaluation of their architecture was done via a very simple scenario where they used an open source signature-based IDS. A monitoring probe sends monitoring data under the UDP flood attack.

Amaral et al. [6] proposed a network-based IDS for IPv6 enabled WSNs. In the proposed scheme, watchdogs which employ network-based IDS are deployed in specific positions within the network. These nodes listen their neighbors and perform monitoring of exchanged packets. IDS modules at each watchdogs use rules to detect the intrusion attempts. These rules are transmitted to watchdogs through a dedicated channel. In order to dynamically configure the watchdogs, the authors used policy programming approach.

In 2015, Pongle et al. proposed an IDS [39] for wormhole attacks. Their IDS has a hybrid architecture similar to SVELTE. Main IDS is located at the BR and lightweight modules are located at the nodes. This study assumes that the nodes are static and the location of each node is known by the BR at the beginning. The main IDS collects neighbor information from nodes and uses it to find out the suspected nodes whose distance is found to be more than the transmission range of a node. The probable attacker is detected by the IDS based on the collected Received Signal Strength Indicator (RSSI) measurements related to the suspected nodes. In terms of the detection resource, we can consider this study as a network-based IDS. And from the detection technique point of view, it can be counted as a specification/rule-based IDS.

Another IDS proposed in 2015 was INTI [14] which targets sinkhole attacks. INTI consists of four modules. The first module is responsible for the cluster formation, which converts the RPL network to a cluster-based network. The second module monitors the routing operations. The third module is the attacker detection module, where reputation and trust parameters are determined by means of Beta distribution. Each node sends its status information to its leader node, which in turn determines the trust and reputation values. Threshold values on these parameters define whether a node is an attacker or not. The fourth module isolates the attacker by broadcasting its information. INTI can be classified as an anomaly-based IDS with distributed IDS architecture. In terms of the detection resources, we can put it into the category of hybrid IDSes.

Sedjelmaci et al. proposed an anomaly-detection technique [48] for low power and lossy networks in 2016. Unlike the other IDSes targeting specific attacks and aiming to detect them, this study focuses on the optimization of running times of detection systems. The motivation of the study is derived from the fact that anomaly-based systems require more resources compared to signature-based systems. If our system can afford to be a hybrid system, having both of the detection systems, then we have to optimize the running time of the anomaly-detection module in order to lengthen the lifetime of the network. The authors

choose game-theory in this study for the optimization of the running time of the anomaly detection system. They claim that, thanks to the game-theory, anomaly detection runs only when a new attack is expected to occur. Anomaly detection runs only during such time intervals and create attack signatures. The signature-based system in turn puts this signature to its database and runs more often than the anomaly-detection system.

Mayzaud et al. proposed a detection system [32] for version number attacks in 2016. Their system uses the monitoring architecture which was proposed in the authors' earlier work [34]. Their monitoring system makes use of the multiple instance support of RPL protocol. It consists of special monitoring nodes with long range communication radios. These nodes are assumed to be covering the whole network and can send the monitoring information to the DODAG root using the second RPL instance that was setup as the monitoring network. In the proposed IDS, monitoring nodes eavesdrop the communication around them and send the addresses of their neighbors and addresses of the nodes who sends DIO messages with incremented version numbers to the root. The root detects the malicious nodes by means of the collected monitoring information. However, the proposed technique suffers from high false positives. This IDS can be counted as a network-based IDS with centralized detection architecture. We can also categorize it as a specification/rule-based IDS.

Another IDS proposed in 2016 was Saeed et al.'s work [45]. This study focuses on the attacks targeting a smart building/home environment where readings of sensors are sent to the server via a base station. In this study, the focus is on the attacks that target the base station. These attacks include software-based attacks and other attacks (i.e., performance degradation attacks, attacks to the integrity of the data). The anomaly-based with a centralized architecture is located at the base station. It consists of two layers. The first layer is responsible for analyzing the behavior of the system and detecting anomalies. It uses Random Neural Networks to create the profile and detect the anomalies. The second part is responsible from the software-based attacks. It comes up with a tagging mechanism to pointer variables. Accesses with the pointer are aimed to be limited with respect to the tag boundaries.

Le et al. proposed a specification-based IDS [27] which is based on their previous work [29]. Their IDS targets rank, sinkhole, local repair, neighbor and DIS attacks. Firstly the proposal obtains an RPL specification via analysis of the trace files of extensive simulations of RPL networks without any attacker. After the analysis of the traces for each node, states, transitions and statistics of each state are obtained. These are merged to obtain a final FSM of RPL which helps them to find out instability states and required statistics. This study organizes the network in a clustered fashion. The IDS is placed at each Cluster Head (CH). CH sends requests to cluster members periodically. Members reply with neighbor lists, rank and parent information. For each member, CH stores RPL related information. CH runs five mechanisms within the concept of IDS. These mechanisms are understanding the illegitimate DIS messages and checks for fake DIO messages, rank inconsistencies and rules, and instability of the network. CH makes use of three thresholds to find out the attackers. These are number

of DIS state and instability state visits, and number of faults. This IDS can be counted as network-based IDS with a distributed architecture.

Aris and Oktug proposed a novel IDS design [9] in 2017 which is an anomaly-based IDS with hybrid architecture. In this study lightweight monitoring modules are placed at each IoT nodes and the main IDS is placed at the BR. Monitoring modules send RPL-related information and resource information of the node. The main IDS module periodically collects the monitoring information and also works as a firewall, where it can analyze the incoming and outgoing traffic from and to the Internet. In this study, each IDS module working on different RPL networks can share suspicious events information with each other. Each IDS works autonomously and detects anomalies using the monitoring information of 6LoWPAN network, firewall information and suspicious events information. When anomalies are detected, nodes within the network are informed via white-lists, whereas other IDSes are informed via the exported suspicious events information. This anomaly-based IDS is a hybrid IDS in terms of architecture and the detection resources used.

Table 4 categorizes the IDSes for IoT. This table shows that majority of the systems are specification/rule-based. It clearly shows that, researchers focused on the protocols (i.e., RPL) rather than a common approach of creating a profile of normal behavior. This observation is also related to signature-based systems being rarely proposed for IoT. The reason may be due to the hardness of creating the signatures for the aforementioned attacks in IoT environments. In terms of the detection architecture, we can see that researchers consider every possible architecture and there is no outperforming option here. When we analyze the detection resources used, most of the studies are network-based IDSes. This shows that, node resources and logs are not yet used frequently by IoT security

**Table 4.** Categorization of intrusion detection systems for IoT

Proposal	Det. arch	Det. technique	Det. resource
Cho [15]	Centralized	Anomaly-based	Network-based
Le [29]	Distributed	Specification/Rule-based	Network-based
Misra [36]	Distributed	Specification/Rule-based	Network-based
SVELTE [41]	Hybrid	Specification/Rule-based	Network-based
Kasinathan [21]	Centralized	Signature-based	Network-based
Amaral [6]	Distributed	Specification/Rule-based	Network-based
Pongle [39]	Hybrid	Specification/Rule-based	Network-based
INTI [14]	Distributed	Anomaly-based	Hybrid
Sedjelmaci [48]	Distributed	Hybrid	Host-based
Mayzaud [32]	Centralized	Specification/rule-based	Network-based
Saeed [45]	Centralized	Anomaly-based	Network-based
Le [27]	Distributed	Specification/rule-based	Network-based
Aris [9]	Hybrid	Anomaly-based	Hybrid



**Table 5.** Target attacks & Implementation environments of intrusion detection systems for IoT

Proposal	Target attacks	Implem. env.
Cho [15]	Botnets	Custom Simulation
Le [29]	Rank, Local Repair	Not-implemented
Misra [36]	General DoS	Custom Simulation
SVELTE [41]	Sinkhole, Selective-forwarding	Contiki Cooja
Kasinathan [21]	UDP flooding	Project Testbed
Amaral [6]	General DoS	Project Testbed
Pongle [39]	Wormhole	Contiki Cooja
INTI [14]	Sinkhole	Contiki Cooja
Sedjelmaci [48]	General DoS	TinyOS TOSSIM
Mayzaud [32]	Version Number	Contiki Cooja
Saeed [45]	Software-based attacks, Integrity attacks, Flooding and other	Prototype impl.
Le [27]	Rank, Sinkhole, Local Repair, Neighbor, DIS	Contiki Cooja

researchers. This may be due to the already limited resources of the nodes which may already be used 100% (e.g., RAM) or no space to store logs. But it is interesting to see that no proposal considers to use the deviation of the power consumption as an intrusion attempt.

Table 5 compares the studies in terms of target attacks and implementation environments. The majority of the attacks targeted by IDSes for IoT are routing attacks as shown in the table. A big portion of the studies focus only on a single attack, whereas only a few studies consider multiple attacks. When we consider these attacks, nearly all of them are insider attacks. This means, IoT security researchers in this concept are not thinking of the threats sourced from the Internet yet. In addition to this, only one study targets software-based attacks. However, we know that embedded system developers choose programming in C language, which may open software-based vulnerabilities to the attackers targeting IoT. In terms of the implementation environment, Contiki Cooja is the environment selected by most of the researchers.

## 9 Discussion and Open Issues

In this study, we provided an extensive overview of Internet of Things security in order to ensure reliable Internet of Services for the future. Of course there may be other studies which were left unmentioned unintentionally. Considering the limitations, attacks, cryptography-based security solutions and studies in the literature, there are still several issues to research in order to reach a secure IoT environment.

One of the major points to consider is the usability and user experience when providing security to IoT environments. We have to consider users and provide user-friendly schemes which will not disturb the satisfaction of users while promising security. This is directly related to the success and acceptance of our solutions. Otherwise, our efforts will be in vain, making the attackers' job easier.

As we have mentioned in Sect. 5.2, some of the MAC layer attacks make use of jamming attack to reach their aim. If we find a solution against jamming attacks, then this may makes it easier to mitigate the effects of such attacks.

IDSes proposed for IoT use thresholds to decide whether a node/connection is malicious nor not. However, considering the proposals, thresholds seem to be set intuitively, not based on a scientific technique. This approach clearly limits the applicability and reproducibility of the proposed mechanism. The way we set thresholds may be an important issue to think about when designing IDSes.

Assumptions of the studies are another point to re-consider. Some studies assume that there is a monitoring network covering the whole network with a minimum number of nodes and was setup at the beginning and is ready to use. Such assumptions have to be supported with deployment scenarios, otherwise it may not be realistic to have such assumptions.

Anomaly-based and also specification-based IDSes typically require an attack-free period where the underlying system will able to understand the normal operating conditions and create a profile accordingly. However, this may not be possible for real-life deployments. In addition to this, if our deployment includes thousands of nodes, then ensuring such a period may not even be feasible.

Most of the studies target only a small number of attacks as mentioned in another study [60]. Researchers have to target a broader range of attacks or propose systems which have the capability to be extended to detect other attacks too.

In terms of the types of attacks, most of the studies focus only on insider attacks, whereas outsider attacks from the Internet have to be researched and analyzed. When we consider Table 1, we can see that attacks above the network layer were not studied extensively. This clearly shows that transport and application layers of IoT may be vulnerable to attacks and IoT will be mentioned a lot within news of DDoS attacks.

Another issue with IoT security research is related to reproducibility and comparability of the studies. When we have a look at the studies, most of the authors keep the source codes of their implementations closed. In addition to this, IoT security research does not have datasets which can be used by the researchers as a common performance evaluation benchmark although testbeds that are publicly available exist. It would enrich the IoT security research if more researchers share their implementations with public and organizations provide datasets which can be used for evaluation purposes.

## 10 Conclusion

In this study while we aim to provide a comprehensive overview of security of IoT for reliable IoS, we incorporated the points of view that include unique characteristics of IoT environments and how they affect security, architectural components

of IoT and their relation to the standardized protocol stack, cryptography-based solutions and their detailed comparisons in addition to considerations on issues (i.e., implementation flaws, users and usability, physical security of the devices and trade offs), taxonomy of D/DoS attacks for IoT, analysis of the studies which analyze the effects of the attacks based on the attacks and findings, examination of mitigation systems and protocol security solutions with respect to mitigation mechanisms and targeted attacks, categorization of D/DoS attacks according to detection architecture, detection technique, detection resources as well as targeted attacks and implementation environments.

Although we can think that cryptography will be enough for us, various issues open our networks to D/DoS attacks. D/DoS attacks are clearly threats not only for availability but also for reliability of future Internet of Services. There are various attacks and literature has several studies to secure the IoT networks against these attacks. When we consider the efforts, we cannot say that IoT security is over now. Clearly, there is still a lot to research and consider.

Although majority of studies examined in this work target 6LoWPAN networks, security of emerging communication technologies such as LoRaWAN, NB-IoT, Thread and many others needs attention of researchers.

Based on our analysis, we can say that a plethora of research exists for routing layer D/DoS attacks, whereas we can not see studies targeting the application layer of IoT. Therefore, security of the application layer considering the attacks and use-cases needs research. In addition to this, most of the studies do not focus on a broad range of attacks, but only a few. There is a need for proposals which are capable of targeting more attacks for IoT security research.

Only a few papers consider IoT to be used as an attacking tool for D/DoS attacks by malicious entities. However, the predicted number of devices in IoT networks is in the order of billions and IoT applications will be weaved into the fabric of our daily lives. It will be very easy for attackers to target. Therefore, there is a serious need for studies which address this issue.

Nevertheless, while researchers will focus into the mentioned issues as future research, they will face with several challenges including resource limitations, heterogeneity of devices and applications, usability and security awareness, management and cost.

**Acknowledgments.** This study was supported by COST Action IC1304 with STSM reference ECOST-STSM-IC1304-010217-081547. It was also supported by 2211C - Domestic Doctoral Scholarship Program Intended for Priority Areas, No. 1649B031503218 of the Scientific and Technological Research Council of Turkey (TUBITAK).

## References

1. IEEE Standard for Local and Metropolitan Area Networks - Part 15.4: Low Rate Wireless Personal Area Networks. IEEE Std. 802.15.4-2011 (2011)
2. DDoS on Dyn Impacts Twitter, Spotify, Reddit (2016). <https://krebsonsecurity.com/2016/10/ddos-on-dyn-impacts-twitter-spotify-reddit/>. Accessed 7 Dec 2016

3. Digital Attack Map Top daily DDoS attacks worldwide (2018). <http://www.digitalattackmap.com/>. Accessed 19 Jan 2018
4. Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M.L., Stransky, C.: How Internet resources might be helping you develop faster but less securely. *IEEE Secur. Priv.* **15**(2), 50–60 (2017). <https://doi.org/10.1109/MSP.2017.24>
5. Adat, V., Gupta, B.B.: Security in Internet of Things: issues, challenges, taxonomy, and architecture. *Telecommun. Syst.* (2017). <https://doi.org/10.1007/s11235-017-0345-9>
6. Amaral, J.P., Oliveira, L.M., Rodrigues, J.J.P.C., Han, G., Shu, L.: Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks. In: 2014 IEEE International Conference on Communications (ICC), pp. 1796–1801 (2014)
7. Amin, Y.M., Abdel-Hamid, A.T.: A comprehensive taxonomy and analysis of IEEE 802.15.4 attacks. *J. Electr. Comput. Eng.*, 1–12 (2016). <https://doi.org/10.1155/2016/7165952>
8. Arce, I., Clark-Fisher, K., Daswani, N., DelGrosso, J., Dhillon, D., Kern, C., Kohno, T., Landwehr, C., McGraw, G., Schoenfeld, B., et al.: Avoiding the top 10 software security design flaws. Technical report, IEEE Computer Societys Center for Secure Design (CSD) (2014)
9. Aris, A., Oktug, S.F.: Poster: state of the art ids design for IoT. In: International Conference on Embedded Wireless Systems and Networks (EWSN 2017) (2017)
10. Aris, A., Oktug, S.F., Yalcin, S.B.O.: Internet-of-Things security: denial of service attacks. In: 2015 23th Signal Processing and Communications Applications Conference (SIU), pp. 903–906 (2015). <https://doi.org/10.1109/SIU.2015.7129976>
11. Aris, A., Oktug, S.F., Yalcin, S.B.O.: RPL version number attacks: in-depth study. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 776–779 (2016). <https://doi.org/10.1109/NOMS.2016.7502897>
12. Bormann, C., Hoffman, P.: Concise Binary Object Representation (CBOR). RFC 7049 (Proposed Standard) (2013). <https://doi.org/10.17487/RFC7049>. <https://www.rfc-editor.org/rfc/rfc7049.txt>
13. Brandt, A., Buron, J., Porcu, G.: Home Automation Routing Requirements in Low-Power and Lossy Networks. RFC 5826 (Informational) (2010). <http://www.ietf.org/rfc/rfc5826.txt>
14. Cervantes, C., Poplade, D., Nogueira, M., Santos, A.: Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 606–611 (2015)
15. Cho, E.J., Kim, J.H., Hong, C.S.: Attack model and detection scheme for botnet on 6LoWPAN. In: Hong, C.S., Tonouchi, T., Ma, Y., Chao, C.-S. (eds.) APNOMS 2009. LNCS, vol. 5787, pp. 515–518. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-04492-2\\_66](https://doi.org/10.1007/978-3-642-04492-2_66)
16. Dvir, A., Holczer, T., Buttyan, L.: VeRA - version number and rank authentication in RPL. In: 2011 IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS), pp. 709–714 (2011)
17. Evans, D.: The Internet of Things how the next evolution of the internet is changing everything (2011). [http://www.cisco.com/web/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf)
18. Hui, J., Thubert, P.: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard) (2011)

19. Hummen, R., Hiller, J., Wirtz, H., Henze, M., Shafagh, H., Wehrle, K.: 6LoWPAN fragmentation attacks and mitigation mechanisms. In: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2013, pp. 55–66 (2013)
20. Kasinathan, P., Costamagna, G., Khaleel, H., Pastrone, C., Spirito, M.A.: DEMO: an IDS framework for internet of things empowered by 6LoWPAN. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, pp. 1337–1340 (2013)
21. Kasinathan, P., Pastrone, C., Spirito, M., Vinkovits, M.: Denial-of-service detection in 6LoWPAN based Internet of Things. In: 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 600–607 (2013)
22. Kent, S., Seo, K.: Security architecture for the internet protocol. RFC 4301 (Proposed Standard) (2005). <https://doi.org/10.17487/RFC4301>. <https://www.rfc-editor.org/rfc/rfc4301.txt>. Updated by RFCs 6040, 7619
23. Kettimuthu, R., Muthukrishnan, S.: Is bluetooth suitable for large-scale sensor networks? In: ICWN, pp. 448–454. Citeseer (2005)
24. Khan, R., Khan, S.U., Zaheer, R., Khan, S.: Future internet: the Internet of Things architecture, possible applications and key challenges. In: 2012 10th International Conference on Frontiers of Information Technology, pp. 257–260 (2012). <https://doi.org/10.1109/FIT.2012.53>
25. Krco, S.: Bluetooth based wireless sensor networks—implementation issues and solutions (2002). <http://www.telfor.org.yu/radovi/4019.pdf>
26. Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach, 6th edn. Pearson (2012)
27. Le, A., Loo, J., Chai, K.K., Aiash, M.: A specification-based IDS for detecting attacks on RPL-based network topology. Information **7**(2) (2016). <https://doi.org/10.3390/info7020025>. <http://www.mdpi.com/2078-2489/7/2/25>
28. Le, A., Loo, J., Lasebae, A., Vinel, A., Chen, Y., Chai, M.: The impact of rank attack on network topology of routing protocol for low-power and lossy networks. IEEE Sens. J. **13**(10), 3685–3692 (2013). <https://doi.org/10.1109/JSEN.2013.2266399>
29. Le, A., Loo, J., Luo, Y., Lasebae, A.: Specification-based IDS for securing RPL from topology attacks. In: 2011 IFIP Wireless Days (WD), pp. 1–3 (2011). <https://doi.org/10.1109/WD.2011.6098218>
30. Levis, P., Clausen, T., Hui, J., Gnawali, O., Ko, J.: The Trickle algorithm. RFC 6206 (Proposed Standard) (2011). <https://doi.org/10.17487/RFC6206>. <https://www.rfc-editor.org/rfc/rfc6206.txt>
31. Martocci, J., Mil, P.D., Riou, N., Vermeylen, W.: Building automation routing requirements in low-power and lossy networks. RFC 5867 (Informational) (2010). <http://www.ietf.org/rfc/rfc5867.txt>
32. Mayzaud, A., Badonnel, R., Chrisment, I.: Detecting version number attacks using a distributed monitoring architecture. In: Proceedings of IEEE/IFIP/In Association with ACM SIGCOMM International Conference on Network and Service Management (CNSM 2016), pp. 127–135 (2016)
33. Mayzaud, A., Sehgal, A., Badonnel, R., Chrisment, I., Schnwlder, J.: Mitigation of topological inconsistency attacks in RPL-based low-power lossy networks. Int. J. Netw. Manag. **25**(5), 320–339 (2015). <https://doi.org/10.1002/nem.1898>
34. Mayzaud, A., Sehgal, A., Badonnel, R., Chrisment, I., Schnwlder, J.: Using the RPL protocol for supporting passive monitoring in the Internet of Things. In:

- NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 366–374 (2016). <https://doi.org/10.1109/NOMS.2016.7502833>
35. Mayzaud, A., Sehgal, A., Badonnel, R., Chrisment, I., Schönwälder, J.: A study of RPL DODAG version attacks. In: Sperotto, A., Doyen, G., Latré, S., Charalambides, M., Stiller, B. (eds.) AIMS 2014. LNCS, vol. 8508, pp. 92–104. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-43862-6\\_12](https://doi.org/10.1007/978-3-662-43862-6_12)
  36. Misra, S., Krishna, P.V., Agarwal, H., Saxena, A., Obaidat, M.S.: A learning automata based solution for preventing distributed denial of service in Internet of Things. In: Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, ITHINGSCPSCOM 2011, pp. 114–122 (2011)
  37. Palattella, M., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L., Boggia, G., Dohler, M.: Standardized protocol stack for the internet of (Important) things. IEEE Commun. Surv. Tutor. **15**(3), 1389–1406 (2013)
  38. Pister, K., Thubert, P., Dwars, S., Phinney, T.: Industrial routing requirements in low-power and lossy networks. RFC 5673 (Informational) (2009). <http://www.ietf.org/rfc/rfc5673.txt>
  39. Pongle, P., Chavan, G.: Article: real time intrusion and wormhole attack detection in Internet of Things. Int. J. Comput. Appl. **121**(9), 1–9 (2015)
  40. Raza, S., Duquenois, S., Chung, T., Yazar, D., Voigt, T., Roedig, U.: Securing communication in 6LoWPAN with compressed IPsec. In: 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), pp. 1–8 (2011). <https://doi.org/10.1109/DCOSS.2011.5982177>
  41. Raza, S., Wallgren, L., Voigt, T.: SVELTE: real-time intrusion detection in the Internet of Things. Ad Hoc Netw. **11**(8), 2661–2674 (2013)
  42. Razzak, F.: Spamming the Internet of Things: a possibility and its probable solution. Proc. Comput. Sci. **10**, 658–665 (2012). <https://doi.org/10.1016/j.procs.2012.06.084>. <http://www.sciencedirect.com/science/article/pii/S1877050912004413>
  43. Renuka Venkata Ramani, C.: Two way Firewall for Internet of Things. Master's thesis. KTH, School of Electrical Engineering (EES) (2016)
  44. Rescorla, E., Modadugu, N.: Datagram transport layer security. RFC 4347 (Proposed Standard) (2006). <https://doi.org/10.17487/RFC4347>. <https://www.rfc-editor.org/rfc/rfc4347.txt>. Obsoleted by RFC 6347, updated by RFCs 5746, 7507
  45. Saeed, A., Ahmadinia, A., Javed, A., Larijani, H.: Intelligent intrusion detection in low-power IoTs. ACM Trans. Internet Technol. **16**(4), 27:1–27:25 (2016). <https://doi.org/10.1145/2990499>. <http://doi.acm.org/10.1145/2990499>
  46. Samaila, M.G., Neto, M., Fernandes, D.A.B., Freire, M.M., Inácio, P.R.M.: Security challenges of the Internet of Things. In: Batalla, J.M., Mastorakis, G., Mavromoustakis, C.X., Pallis, E. (eds.) Beyond the Internet of Things. IT, pp. 53–82. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-50758-3\\_3](https://doi.org/10.1007/978-3-319-50758-3_3)
  47. Schneier, B.: Stop trying to fix the user. IEEE Secur. Priv. **14**(5), 96 (2016). <https://doi.org/10.1109/MSP.2016.101>
  48. Sedjelmaci, H., Senouci, S.M., Al-Bahri, M.: A lightweight anomaly detection technique for low-resource IoT devices: a game-theoretic methodology. In: 2016 IEEE International Conference on Communications, ICC 2016 (2016). <https://doi.org/10.1109/ICC.2016.7510811>
  49. Sehgal, A., Mayzaud, A., Badonnel, R., Chrisment, I., Schönwälder, J.: Addressing DODAG inconsistency attacks in RPL networks. In: Global Information Infrastructure and Networking Symposium (GIIS 2014), pp. 1–8 (2014)
  50. Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard) (2014)

51. Sokullu, R., Dagdeviren, O., Korkmaz, I.: On the IEEE 802.15.4 MAC Layer Attacks: GTS Attack. In: Second International Conference on Sensor Technologies and Applications, SENSORCOMM 2008, pp. 673–678 (2008)
52. Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., Richardson, M.: A security threat analysis for the routing protocol for low-power and Lossy Networks (RPLs). RFC 7416 (Informational) (2015). <http://www.ietf.org/rfc/rfc7416.txt>
53. UC Berkeley Center for Long-Term Cybersecurity: Cybersecurity Futures 2020. Technical report (2016)
54. Wallgren, L., Raza, S., Voigt, T.: Routing attacks and countermeasures in the RPL-based internet of things. *Int. J. Distrib. Sens. Netw.* **2013**, 11 (2013)
55. Weekly, K., Pister, K.: Evaluating sinkhole defense techniques in RPL networks. In: 2012 20th IEEE International Conference on Network Protocols (ICNP), pp. 1–6 (2012)
56. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard) (2012)
57. Wu, M., Lu, T.J., Ling, F.Y., Sun, J., Du, H.Y.: Research on the architecture of Internet of Things. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. V5–484–V5–487 (2010). <https://doi.org/10.1109/ICACTE.2010.5579493>
58. Yang, Y., Wu, L., Yin, G., Li, L., Zhao, H.: A survey on security and privacy issues in Internet-of-Things. *IEEE Internet of Things J.* **4**(5), 1250–1258 (2017). <https://doi.org/10.1109/JIOT.2017.2694844>
59. Yang, Z., Yue, Y., Yang, Y., Peng, Y., Wang, X., Liu, W.: Study and application on the architecture and key technologies for IoT. In: 2011 International Conference on Multimedia Technology, pp. 747–751 (2011). <https://doi.org/10.1109/ICMT.2011.6002149>
60. Zarpelo, B.B., Miani, R.S., Kawakani, C.T., de Alvarenga, S.C.: A survey of intrusion detection in Internet of Things. *J. Netw. Comput. Appl.* **84**, 25 – 37(2017). <https://doi.org/10.1016/j.jnca.2017.02.009>. <http://www.sciencedirect.com/science/article/pii/S1084804517300802>

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.





The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.







# TCP Performance over Current Cellular Access: A Comprehensive Analysis

Eneko Atxutegi<sup>1</sup>, Åke Arvidsson<sup>2</sup>, Fidel Liberal<sup>1</sup>,  
Karl-Johan Grinnemo<sup>3</sup>, and Anna Brunstrom<sup>3</sup>

<sup>1</sup> University of the Basque Country (UPV/EHU), Bilbao, Spain  
{eneko.atxutegi,fidel.liberal}@ehu.eus

<sup>2</sup> Kristianstad University, Kristianstad, Sweden  
ake.arvidsson@hkr.se

<sup>3</sup> Karlstad University, Karlstad, Sweden  
{karl-johan.grinnemo,anna.brunstrom}@kau.se

**Abstract.** Mobile Internet usage has increased significantly over the last decade and it is expected to grow to almost 4 billion users by 2020. Even after the great effort dedicated to improving the performance, there still exist unresolved questions and problems regarding the interaction between TCP and mobile broadband technologies such as LTE. This chapter presents a thorough investigation of the behavior of distinct TCP implementation under various network conditions in different LTE deployments including to which extent TCP is capable of adapting to the rapid variability of mobile networks under different network loads, with distinct flow types, during start-up phase and in mobile scenarios at different speeds. Loss-based algorithms tend to completely fill the queue, creating huge standing queues and inducing packet losses both under stillness and mobility circumstances. On the other side delay-based variants are capable of limiting the standing queue size and decreasing the amount of packets that are dropped in the eNodeB, but under some circumstances they are not able to reach the maximum capacity. Similarly, under mobility in which the radio conditions are more challenging for TCP, the loss-based TCP implementations offer better throughput and are able to better utilize available resources than the delay-based variants do. Finally, CUBIC under highly variable circumstances usually enters congestion avoidance phase prematurely, provoking a slower and longer start-up phase due to the use of Hybrid Slow-Start mechanism. Therefore, CUBIC is unable to efficiently utilize radio resources during shorter transmission sessions.

**Keywords:** TCP adaptability · LTE · Flow size · Slow-Start  
Mobility

## 1 Introduction

Mobile Internet usage has increased significantly over the last decade, growing almost 18-fold over the past 5 years and more than half a million new mobile devices and connections in 2016 [1]. The following years are expected to be



equally promising with 4G traffic reaching quotas of more than three-quarters of the total mobile traffic by 2021. The growth expectation is not only related to the traffic volume itself but also to the average speed. To continue this growth and to meet user expectations, all the involved stakeholders have a common interest in fast downloads, quick responses, high utilization and few packet losses.

Since a large part of mobile Internet comprises TCP flows, the performance of TCP over cellular networks has become an important research topic. Even though in the last three decades many different TCP implementations have been developed [2] each of them targeting a different Congestion Control Algorithm (CCA), there still exists room for improvement in terms of achieved throughput and resulting delay over highly variable mobile networks.

Previous studies and proposals have reported their results regarding the interaction effects between mobile networks and TCP [3–5] and tried to define suitable CCAs for mobile networks [6]. However, none of them have extensively study the implication of a wide range of TCP implementations in a variety of static and moving scenarios. This chapter complements and extends previous works on mobile networks by studying and evaluating the behavior of a selection of TCP variants with different packet sizes, network loads, during start-up and mobility with different speeds, i.e. scenarios that are considered challenging for TCP. In order to appropriately study the different sources capable of impacting the final performance, the chapter suggests a bottom-up scenario with respect to complexity starting with static conditions so as to understand the responsiveness of TCP under distinct network status and load combinations and finishing with a variety of mobility scenarios.

The chapter is organized as follows. Section 2 covers related work. In Sect. 3, a brief overview of the studied TCP variants is provided and the LTE testbeds are described. Next, in Sect. 4, we explain the methodology regarding the performed measurements and the studied scenarios. The findings and results from our work are presented in Sect. 5. Finally, Sect. 6 concludes the chapter with a summary and a discussion of future work.

## 2 Related Work

TCP and LTE cellular access have been deeply studied throughout the last years. Most of the studies have either research the TCP side or mobile network side. However, a significant amount of researchers have been attracted by the interaction between TCP and LTE.

One of the first basis of such interaction is the impact that radio retransmissions have into the delay increment and how they therefore degrade the achieved goodput [7, 8]. It has been proven that the number of simultaneously active User Equipments (UEs) towards a common eNodeB has a huge impact on the effective available bandwidth due to radio resources being shared. Thus, the work [9] found that sudden increases in background traffic load have an important effect in the Round-Trip Time (RTT) increment. This cross-traffic effect severely influences the network playground for TCP, provoking sudden changes in the network

conditions and making TCP struggle while following the fluctuations in the available capacity. This chapter compiles a more detailed treatment of the effects of buffering in the radio access part of LTE by also considering the performance of high-speed/long-delay variants of TCP in these kinds of networks.

The so-called bufferbloat effect has also a huge impact into the performance of TCP over LTE [10]. The bufferbloat effect is possible due to the configuration of long queues both in the end-nodes and intermediate nodes, which can accumulate a great number of packets without any drop. However, that excessive packet buffering in a single queue in the end-to-end network path, causes a great latency increase and therefore, throughput degradation. Our work does not merely focus on bufferbloat, but considers the implications of different TCP variants in queue build-up under certain network conditions.

Considering that many flows in Internet are short, it is important to verify the efficiency of TCP to carry out such transmissions over cellular networks, it has been demonstrated [11] that under some network conditions TCP fails to correctly utilize the available capacity and therefore, the flows last longer than necessary. The current work complements such works and analyzes the impact that different flow sizes have in the performance outcome of different TCP flavors. To this end, our work not only focuses on the stationary phases of TCP but also on its behavior during start-up due to its significant impact in short flows performance. In particular, we study the Hybrid Slow-Start scheme [12], and evaluate how it operates in LTE networks in comparison with the Standard Slow-Start scheme.

Other studies have measured TCP over live LTE networks. Apart from the classic metrics of TCP throughput and RTT in [4] they also measured the delay caused by mobile devices going from idle to connected state. In [13], measurement trials were carried out over the cellular access of four Swedish operators and the diurnal variation of TCP throughput and delay were analyzed. [14,15] studies did similar TCP measurements, however, they did not consider daily variations. None of these live measurements took into account the impact of speed in the performance of TCP, or the behavior on different types of CCAs. So, to the best of our knowledge, our work both complements and extends these works through the study and evaluation of the behavior of common TCP variants in LTE networks under mobility with different speeds.

There are only a few works that have considered the impact of different speeds on the performance of TCP over LTE networks. Even though some works [5] have studied different speeds, the primary metrics were more related to the radio part with spectral efficiency and share of resource blocks among the UEs. Even though the utilization of such radio resources was studied, one or two simple variants of TCP were utilized in a multi-user resource share, leading to TCP micro-effects masking. Also, in [3] the impact of speed on TCP in LTE was studied. The work focused on uplink and downlink throughput, RTTs and also considered time-of-day variations. Still, they did not consider how the CCA factor into the TCP performance at different velocities. Our work serves to cover all the options and extends the previous studies with multiple mobility patterns, different speeds and a wide range of CCAs.

### 3 Research Environment

In order to compare the behavior of TCP in LTE networks, we first choose the TCP variants AND identify the LTE working parameters. This section first describes the most important features of the selected CCAs and later presents the LTE setup.

#### 3.1 TCP Variants

TCP variants fall into three categories according to the CCA mechanism used: loss-based, delay-based and combined loss- and delay-based. Along this chapter, the analysis starts with five CCAs and, with every measurement phase, we will reduce the group, avoiding the repetitive usage of TCP solutions that do not work well in mobile networks. A brief overview of the TCP variants is given below together with the classification of CCAs in Table 1.

**Table 1.** Selected TCP CCAs and their category

CCA category	Selected TCP CCA
Loss-based	TCP NewReno
	TCP CUBIC
Delay-based	TCP CDG
Hybrid with bandwidth estimation	Westwood+
Hybrid without bandwidth estimation	Illinois

- (i) TCP NewReno [16] employs the well-known additive increase multiplicative decrease (AIMD) mechanism that is common to most CCAs. During the Slow-Start period the *cwnd* increases by one packet per acknowledgment (ACK) reception until it reaches the value of *ssthresh*. Afterwards, the *cwnd* enters the congestion avoidance phase, with an increment of one packet per RTT period (standard synchronization with RTT or RTT-synchronized). If a 3-duplicate ACKs (3DUPACK) are received or a time-out occurs, the CCA deducts that some link is congested. After 3DUPACK, NewReno establishes the *cwnd* to the half (basic back-off) and the new *ssthresh* to previous *cwnd*. However, if a time-out occurs the *cwnd* will be decreased to one packet. NewReno is essential in the measurements since it represents the base TCP behavior.
- (ii) TCP CUBIC [17] employs a different mechanism compared with AIMD based on a cubical function. After a decrease of the *cwnd*, the *cwnd* ramps up in a concave shape, until it achieves the value that the *cwnd* had before the reduction. Afterwards, CUBIC increases its growth rate and ramps-up in a convex shape. CUBIC uses Hybrid Slow-Start [12] mechanism in the

sender instead of the Standard Slow-Start phase. Hybrid Slow-Start aims at finding the proper exit point for standard Slow-Start in order to avoid massive packet losses. The detection of such an exit point is based on the measurements of ACK trains and RTT delay samples. The TCP CUBIC implementation has been selected for the analysis due to its widespread use due to the fact that it currently is the default CCA in Linux servers, whose market share comprises the 67% of world-wide servers (as stated by W3Techs [18]).

- (iii) TCP CAIA delay gradient (CDG) [19] modifies the TCP sender to use RTT gradients as a congestion indicator. CDG also calculates the state of the bottleneck queue so that packet losses are treated as congestion signals only when the queue is full. Finally, CDG also uses Hybrid Slow-Start but with a more strict configuration than CUBIC. The selection of TCP CDG has been based on its novel use of delay gradients in the AIMD mechanism and to evaluate the actual usefulness of such a different feature in mobile networks.
- (iv) TCP Westwood+ [20] is capable of estimating the available bandwidth and minimum RTT (RTTmin) by measuring ACK inter-arrival times. The estimations are used to decide the new *cwnd* after a congestion episode of 3DUPACK. With timeouts the *ssthresh* is calculated in accordance to the estimations and the *cwnd* is set to 1 segment. TCP Westwood+ has been selected in this study for its hybrid behavior using loss-based mechanisms together with delay-awareness.
- (v) TCP Illinois [21] controls the AIMD mechanism by the estimated queuing delay and buffer size. In a normal situation when no queuing delay is detected, the *cwnd* is increased by 10 packets per RTT. If estimated delay starts increasing, the increment of *cwnd* will be gradually lowering until the minimum value of 0.3 packets per RTT is reached. When the RTT is considered as high as compared to the baseline RTT, the loss is considered as buffer overflow, whereas in low RTT the loss counts as packet corruption. Developed to perform efficiently within high speed networks, its loss-based and delay-awareness make a perfect candidate for our study.

### 3.2 LTE Setup

In order to evaluate the performance of LTE three different environments have been used: simulation, emulation and controlled deployment. Most of the work described in this chapter has been carried out over the simulated environment and for comparison purposes the findings and results have been correlated with the behavior in the other two deployments. Since the configuration and explanation of the simulated environment is comprised of many parameters and in order to help the reader understand the setup, Table 2 gathers the most important information about the simulation environment regarding the configuration parameters and experiment-related conditions.

As the simulated environment, ns-3 simulator with the LTE capabilities of LENA module is used. This module also allows to create standard-based fading

traces that can be applied to the channel between the UE and the eNodeB. Since ns-3 does not exactly use the available TCP implementations in the Linux kernel, we used Direct Code Execution (DCE) Cradle [22] to be able to run real TCP implementations in ns-3. In order to simulate the distance to the server, the propagation delay between the fixed remote host and the Packet Data Network Gateway (PGW) was set to 40 ms. In the Radio Link Control (RLC) layer we selected the Acknowledged Mode (AM) in order to resemble the most commonly deployed configuration in real-world. We modified the mechanism to be able to support a limitation in terms of packets, establishing in our setup a common packet buffer size in the eNodeB of 750 packets. Regarding the radio resources, the eNodeB was configured to have a standard value of 100 available physical resource blocks (PRB). We simulated the frequency band 7 (2600 MHz), one of the most commonly used commercial LTE frequency bands (in Europe).

Background flows are used to load the network with multiple short TCP connections, similar to the behavior of real networks. The same TCP variant is used for both background and foreground traffic in order not to be affected by issues of TCP friendliness. The amount of data transferred in a background connection as well as the inter-arrival time between two connections were drawn from uniform random distributions.

The controlled testbed aims at providing a measurement platform with the ability to measure TCP in more realistic radio conditions in order to confirm or reject the findings and assumptions made in simulated environment in relation to the behavior of TCP over LTE. We have used the iMinds'/iMEC's LTE facility (LTE w-iLab.t [23]) in Zwijnaarde, Ghent. Apart from the provisioning of all the agents involved in LTE, the deployment allows ad-hoc mobility patterns while

**Table 2.** Simulation parameters

Simulation environment	
Simulator	ns-3 LENA LTE model
Linux Kernel	4.3 (DCE)
CCA	NewReno/CUBIC/Illinois/CDG/Westwood+
Parameter	Value
One-way delay PGW-Server	40 ms
MAC scheduler	Proportional fair
AMC model	MiError
Number of PRBs	100
LTE band	7 (2600 MHz)
RLC mode	AM
RLC transmission queue	750 PDUs
Pathloss model	FriisPropagationLossModel
Fading models	EVA60/EVA200

experimenting. It is important to underline that in this environment, the LTE transmissions are done over the air, thus allowing a proper study of TCP and LTE events. Even though the movement is real, the space limitation could limit the employed speed.

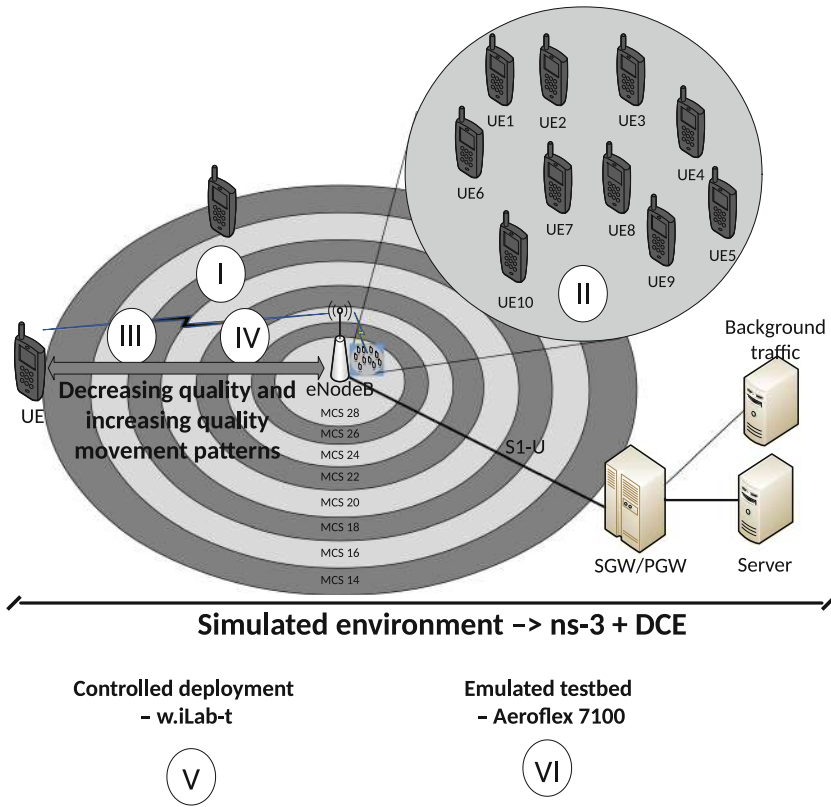
The emulated testbed targets the validation of simulated results of TCP under mobility circumstances. To this end, a LTE emulator or LTE-in-a-box (Aeroflex 7100) has been used. This emulator is capable of creating the LTE radio signal and all the necessary LTE protocol events to support the attachment and registration of any LTE device through a radiofrequency cable or over the air. The tests have been completed with an smartphone, a couple of servers and a controller to synchronize the experiments and all the equipment involved during the assessments. Since the UE in the emulated testbed is not able to physically move, the controller would continuously manipulate the baseline Signal-to-interference-plus-noise ratio (SINR) levels and Aeroflex would apply the corresponding fading pattern so as to model actual movement.

## 4 Methodology Description

The intrinsic operation mode of LTE (i.e. resource sharing, scheduling, HARQ mechanisms) results in a constant change in the available capacity. Even considering single-UE scenarios, different positions and fadings would lead to have a different SINR and it would therefore report a distinct Channel Quality Indicator (CQI) to the eNodeB. Thus, the eNodeB would assign a different available capacity for the channel of the UE through the Modulation and Coding Scheme (MCS) and transport block size (tbSize). Due such fluctuations in the radio side, the *cwnd* will be continuously evolving in order to obtain a resulting goodput as close as possible to the available capacity. The relative progressions of both parameters (available capacity and achieved capacity) play a fundamental role in the final performance.

In this section, the applied methodology will be presented. Figure 1 shows the different scenarios that have been used in the analysis of the effects between TCP's different CCAs and LTE. The methodology and reasoning of each scenario is explained below.

(I) Implication of cross-traffic and responsiveness of TCP: The static scenario aims at providing insights of the evolution and responsiveness of TCP under different background traffics (I point in Fig. 1). There are three main goals with this scenario: the comparative study of TCP behavior with and without a loaded cell, the analysis of TCP focusing on short flows and the responsiveness comparison of TCP variants with a sudden capacity increase and decrease. Several metrics are gathered at different nodes along the path. At the source, TCP state information such as *cwnd* and *ssthresh* is saved. At the eNodeB, the transmission buffer length, the drop count and the Packet Data Convergence Protocol (PDCP) delay (i.e., the time it takes for a PDCP Protocol Data Unit -PDU- to go from the eNodeB to the UE), are logged. Finally, in the UE, the goodput is recorded. Since it is measured at the application level, packet losses and/or reordering may result in goodput spikes.



**Fig. 1.** Scenarios in use.

(II) Start-up performance: The aim of this scenario is to analyze the impact of different CCAs' Slow-Start phases, such as the above mentioned standard Slow-Start and Hybrid Slow-Start, and determine their adequacy or inadequacy in broadband mobile networks. To that purpose, we deployed 10 static and scattered UEs (II point in Fig. 1) in good radio conditions (CQI 15) so as to study the start-up performance in a simplified multi-user scenario and set some basis for the understanding of the following scenarios. In the server, the *cwnd*, RTT, outstanding data and goodput has been collected.

(III & IV) Cell outwards/inwards movement resulting on decreasing/increasing available capacity: The decreasing quality movement scenario evaluates the behavior of TCP with a constantly worsening channel quality on average (III point in Fig. 1). The idea is to assess the CCA's adaptability in a continuous capacity reduction (on average) environment and the impact of UE's speed on the final performance. To help simulate different speeds, two Extended Vehicular A Model (EVA) fading patterns are applies: one for the velocity of 60 km/h (common limitation in rural roads) and one for 200 km/h (common maximum speed

in high-speed trains). Apart from the usual metrics in the evaluation of CCAs, the main metric for simulated mobility-based scenarios is the relation between the available capacity (extracted from the `tbSize`) and the achieved goodput. On the other hand, the increasing quality movement represents the behavior of TCP on a constantly improving channel quality (IV point in Fig. 1). Therefore, these simulations aim at evaluating the CCA's adaptability under different UE's speeds in a continuous increasing capacity (on average) environment.

(V) Correlation of TCP behavior in deployments as similar as possible to live commercial LTE networks: The scenario (V point in Fig. 1) aims at providing a measurement platform with the ability to measure TCP in more realistic radio conditions in order to confirm or reject the findings and assumptions made in simulated environment in relation to the behavior of TCP over LTE. Since the equipment in the scenario is fully real (see description in [23]), the scheduling, queuing and the rest of the features that could have an impact on delay are realistic and represent more clearly what would happen in live scenarios, helping in the verification of findings.

(VI) Emulated support to correlate mobility-based scenarios: Since the previous scenario is limited in terms of speed, the emulated testbed (VI point in Fig. 1) due to the utilization of real UEs and the ability to emulate movement, is capable of confirming and clarifying performance trade-offs that in simulated environment could be blurry. In order to better understand the evolution of different performance-related parameters, in the server, the *cwnd*, RTT, outstanding data and goodput have been collected.

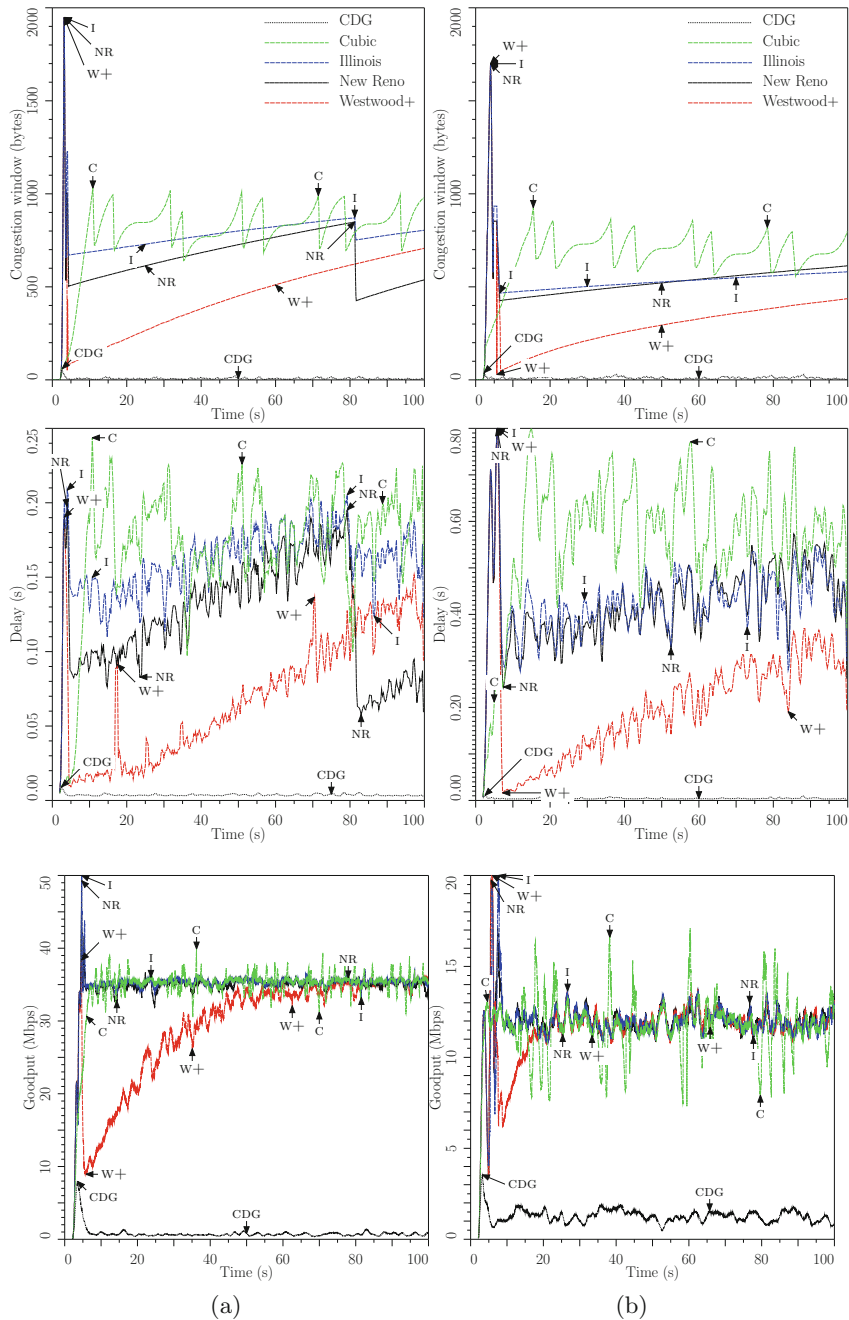
## 5 Analysis of the Interactions Observed in Different Scenarios

This section is divided in five main parts: implication of cross-traffic and responsiveness of TCP (with scenario I), the start-up performance (with scenario II), both decreasing quality and increasing quality movement scenarios (with scenario III and IV), the correlation of findings in the controlled deployment (with scenario V) and finally, the correlation of findings regarding mobility scenarios' over emulated testbed (with scenario VI).

### 5.1 Cross-Traffic Impact and Responsiveness of TCP

This subsection is responsible for covering different kind of traffic loads and behaviors while the UE is static. The location of the UE among the different measurements is the same and thus, the results are comparable. The subsection is divided in three main experiments: the comparison between a single-UE without cross-traffic and a loaded network, the impact of short flows and finally, sudden increase and decrease of the available capacity.





**Fig. 2.** Performance comparison of the selected CCAs: (a) Base single flow behavior; (b) Single flow behavior over loaded network.

Base Behavior and Behavior in a Loaded Network

According to the selected position, the UE has a maximum throughput around the half of the total maximum (35 Mbps). Different experimental trials are carried out with and without background traffic to study the responsiveness of TCP and infer whether the background traffic has the same impact among the CCAs or not. In order to make easier the reading, Table 3 gathers the most important point of the following explanation.

The three subfigures on the left of Fig. 2 depict the results regarding the scenario with no background traffic. The differences between the loss-based TCP variants and delay-based ones are remarkable even in such a simplified scenario. Loss-based implementations manage to achieve the maximum capacity and create a long standing queue delay (up to 250 ms), whereas delay-based variants, such as CDG, keep the delay controlled but fail while trying to reach full resource utilization. In the case of Westwood+, it is clear that the applied back-off after Slow-Start is very drastic and due to this, it takes longer to ramp-up. Illinois minimally reduces the *cwnd*, causing huge standing queue delay comparing with more conservative implementations like NewReno. In the case of CUBIC, it suffer for the deficient behavior of Hybrid Slow-Start. The mechanism exits to the congestion avoidance phase in an early stage and therefore reduces its growth pace far from the maximum achievable capacity, severely impacting in the time it takes to converge.

The three subfigures on the right of Fig. 2 show the outcome for the same scenario but with background traffic. The total target load of the background

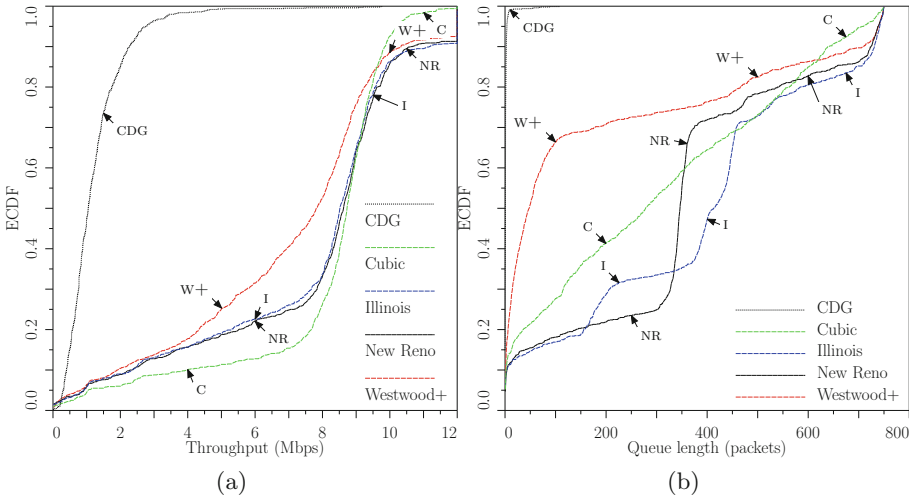
Table 3. Findings wrap-up in base behavior and behavior in a loaded network

CCA	Conditions	Behavior
CUBIC	Base behavior	Slightly suffers for the deficient behavior of Hybrid Slow-Start
	Loaded network	No impact of Hybrid Slow-Start
NewReno	Base behavior	Easily achieves maximum capacity
	Loaded network	Similar behavior but with higher delay and more unstable goodput
Illinois	Base behavior	Easily achieves maximum capacity. However, it creates a huge standing queue
	Loaded network	Very similar to NewReno but with slightly higher delay
CDG	Base behavior	Keeps the delay controlled but fails while trying to reach full resource utilization
	Loaded network	The differences with loss-based CCAs are reduced
Westwood+	Base behavior	Very aggressive back-off that impacts the time needed to ramp-up
	Loaded network	The impact of the back-off application is minimized

traffic is set to the 50% of the link capacity. The capacity reduction minimizes the performance gap between loss-based and delay-based variants and still, the more capacity a CCA gets, the harder impact it inflicts in terms of queuing delay (Illinois as an example). Big differences appear comparing with the base example without background traffic, mostly related to a significant increment in the queuing delay and the reduction of the gap in terms of capacity to reflect the differences amongst the CCAs. RTT-clocked CCAs suffer due to a lengthen of the time between implementation decisions. In contrast, CUBIC behaves better because it does not suffer for RTT increase. The scenario itself due to its reduction in the available capacity cushions the underperformance of Hybrid Slow-Start.

**Short Flows Study**

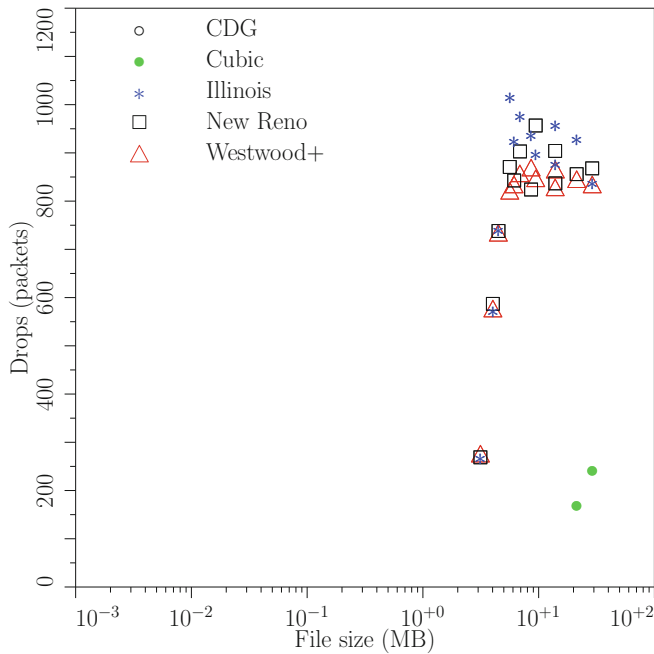
Live measurements have shown that many flows over Internet are small (90% of downstreams carry no more than 35.9 KB of data [4]). Therefore it is important to assess the impact that such load distribution has in final performance. In order to do so, the previous foreground TCP flow must be replaced by a succession of short flows following an exponential distribution regarding their amount of data.



**Fig. 3.** Throughput and queue size ECDF at 700 m

The Fig. 3 represent as an Empirical Cumulative Distribution Function (ECDF) the results obtained regarding the achieved throughput and standing queue size. In Fig. 4 the size of the flows and the number of induced drops are correlated.

In Fig. 3a, it is clear that the achieved throughput is very similar among most the CCAs and their differences really appear regarding the amount of enqueued packets in Fig. 3b. The delay-based variant, CDG, successfully limits



**Fig. 4.** eNodeB drops at 700 m

the enqueued packets while loss-based implementations overshoot causing a great standing queue. Due to the detected behavior of Westwood+ in the beginning of the transmissions and the short duration of the flows, it prompts little ability to inject packets in the eNodeB. In contrast, NewReno, Illinois and CUBIC happen to be the average solutions. If we compare two deficient solutions such as CUBIC and Westwood+, we clearly see that even with short flows and therefore quick transmission duration, the premature exit from Slow-Start for the former performs better than the excessive back-off of the latter.

Considering the reported findings, Fig. 4 shows the number of packets that have exceeded the queue size with each flow size, being therefore dropped. It is clear that the more aggressive the CCA is, the more packet are dropped by the eNodeB. Illinois for instance has a more aggressive behavior in congestion avoidance phase. It enqueues more packets and gets more packets dropped. As a result Illinois suffers on average 100 more dropped packets than any other TCP candidate. Once again, the behavior of Hybrid Slow-Start is clearly shown. If we avoid the fact that the transmissions with Hybrid Slow-Start take slightly more time to be completed, it only suffers drops with longer transmissions and when it has congestion events, the number of losses are very few. With loss-based AIMD mechanisms, the drop packets metric appears to be directly related to the aggressiveness and back-off strategy. NewReno and Westwood+ have quite similar results (Table 4).

**Table 4.** Findings wrap-up in short flows study

CCA	Behavior
CUBIC	Thanks to the underperformance of Hybrid Slow-Start, it only suffers drops with longer transmissions and when it has congestion events, the number of losses are very few
NewReno	The average solution
Illinois	Very aggressive behavior that results in 100 more dropped packets on average
CDG	Successfully limits the enqueued packets
Westwood+	In the beginning of the transmissions and the short duration of the flows, it prompts little ability to inject packets due to the aggressive back-off

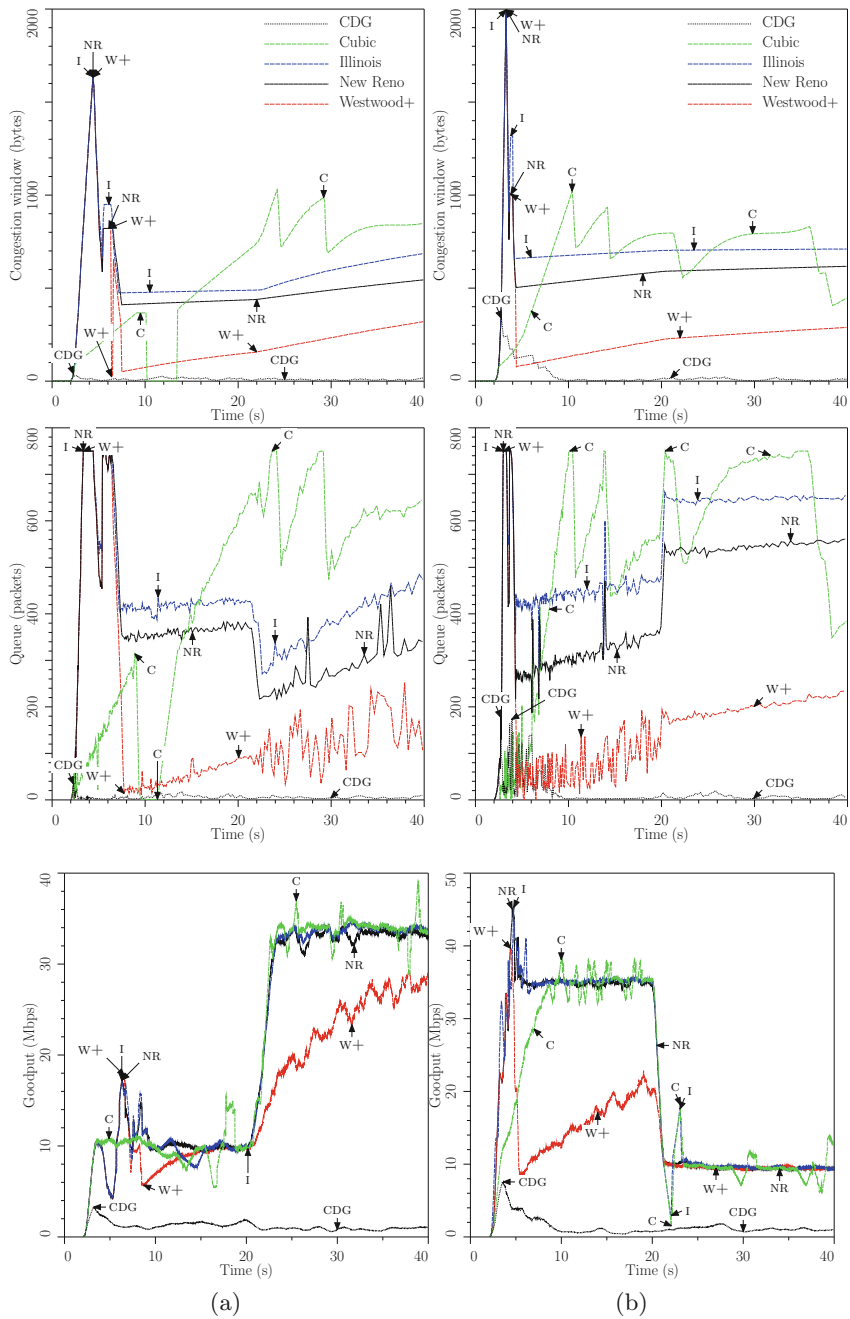
**Sudden Increase and Decrease of the Available Capacity**

Once the main features of the CCAs have been detected in loaded scenarios in comparison with the base behavior as well as the impact of different short flows on the drop rate, it is important to study the responsiveness of CCAs in big and sudden capacity changes. To this end, two type of simulations are carried out: with the background traffic being stopped at 20s of the test and with the background traffic being started at 20s of the test. In order to make easier the reading, Table 5 gathers the most important point of the following explanation.

On the one hand, the left part of Fig. 5 shows the results regarding the scenario with a sudden capacity increase. In general, as soon as the capacity increases, the queue size is lowered due to a release of previously enqueued packets. It is clear that loss-based CCAs quickly respond to an additional bandwidth assignment. However, Westwood+ still suffers from the excessive reduction of the *cwnd* after the Slow-Start phase. During the congestion avoidance phase, its AIMD mechanism is very conservative and the enqueued packets tend to be almost 0, therefore with a new and greater achievable capacity, the adaptation ability of the CCA is very weak. In the case of delay-based variants, since they mainly focus on reducing the delay over path, they usually fail to increase their pace and thus, the new available capacity is wasted.

On the other hand, the right part of Fig. 5 depicts the case in which the background traffic is activated at 20s. Due to the sudden reduction of available capacity, the queue size suffer an instant increment because of the relation between the same number of incoming packets to the eNodeB and the drastic reduction of outgoing ones. The Fig. 5 clearly shows that all CCAs but CDG are able to successfully react to the capacity reduction. However, in some cases such as CUBIC, the CCA takes more time to stabilize to the new pace.

These simulations reflect that most CCAs, even delay-based implementations, are capable of reducing their throughput when sudden available capacity decreases happen but delay-based variants struggle to adapt their pace to bandwidth increases.



**Fig. 5.** Performance comparison of the selected CCAs: (a) Sudden capacity increase; (b) Sudden capacity decrease.

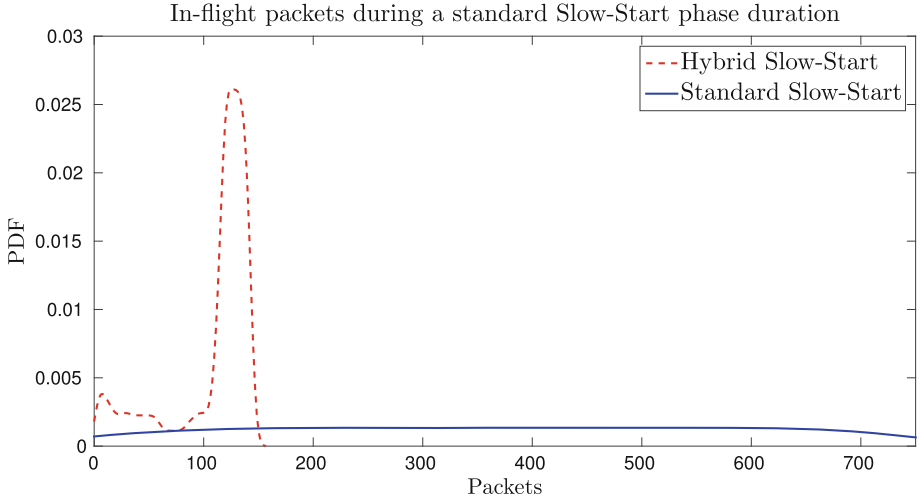
**Table 5.** Findings wrap-up in sudden increase and decrease of the available capacity

CCA	Conditions	Behavior
CUBIC	Inc. available cap.	Good performance without the impact of Hybrid Slow-Start due to the low available capacity at the beginning of the transmission
	Dec. available cap.	Impact of Hybrid Slow-Start in the beginning. Aggressive behavior in congestion avoidance phase that leads to an instant huge increment of queue size while reducing the available capacity
NewReno	Inc. available cap.	Good responsiveness and average delay impact
	Dec. available cap.	Average loss-based solution that suffers and instant standing queue increase while reducing the available capacity
Illinois	Inc. available cap.	Good responsiveness and greater induced delay than NewReno
	Dec. available cap.	Its aggressiveness is harmful in this scenario and takes some time to stabilize the goodput
CDG	Inc. available cap.	Fails to increase its pace and thus, the new available capacity is wasted
	Dec. available cap.	Bad performance in terms of goodput but full control of the delay that is always close to the baseline delay
Westwood+	Inc. available cap.	Its AIMD mechanism is very conservative and the enqueued packets tend to be very few, being not capable of responding to a sudden greater capacity assignment
	Dec. available cap.	The combination of its dynamics (with a slow ramp-up ability) and the available capacity reduction happen to get the best performance due to the achievement of the maximum goodput and the lowest impact in terms of delay

## 5.2 Start-Up Performance

In very simplified scenarios, we have seen that the behavior of Standard Slow-Start and Hybrid Slow-Start differs leading in some occasions to a successful avoidance of massive losses with Hybrid Slow-Start. However, the LTE cells are usually more crowded and therefore the UEs could inflict more delay as cross-traffic that could impact Hybrid Slow-Start. The target is to assess whether the internal mechanisms of Hybrid Slow-Start could provoke an early exit from the standard ramp-up, following to a slow increment of the *cwnd* and therefore a significant underutilization of radio resources or not.

We first measured the convergence behavior of the Standard Slow-Start, recording the packets that are in-flight at every moment. Later, we assessed



**Fig. 6.** Hybrid Slow-Start impact in mobile networks: injected packets during a standard Slow-Start period.

the same for Hybrid Slow-Start. Figure 6 shows the probability density function (PDF) of the number of injected packets for both mechanisms in the time Standard Slow-Start takes to converge. This is, we would compare in the fastest convergence period of time, the ability of both methods to put packets in-flight.

Figure 6 shows the behavior of Standard Slow-Start has a equal distribution of packets in flight, whereas Hybrid has an imbalanced distribution presumably formed by the period of time in which Hybrid Slow-Start has ramped-up as Standard Slow-Start and the period after detecting a delay variation and behaving under the incremental pace of congestion avoidance phase. The distribution represents the huge difference between both methods regarding the ability to inject packets which leads to a extrapolation of the time needed to converge or achieve the maximum capacity from the beginning of the transmission.

It is clear that not only in simplified scenarios, but also in multi-UE measurements, Hybrid Slow-Start suffers due to the detection of delay increment and the early trigger of exit condition from fast ramp-up. So, under some delay variability circumstances Hybrid Slow-Start slows-down the ramp-up of TCP. In some situations, this effect could underutilize the available radio resources and lengthens the time needed to converge, directly impacting on the quality experienced by users (QoE).

### 5.3 Mobility Performance

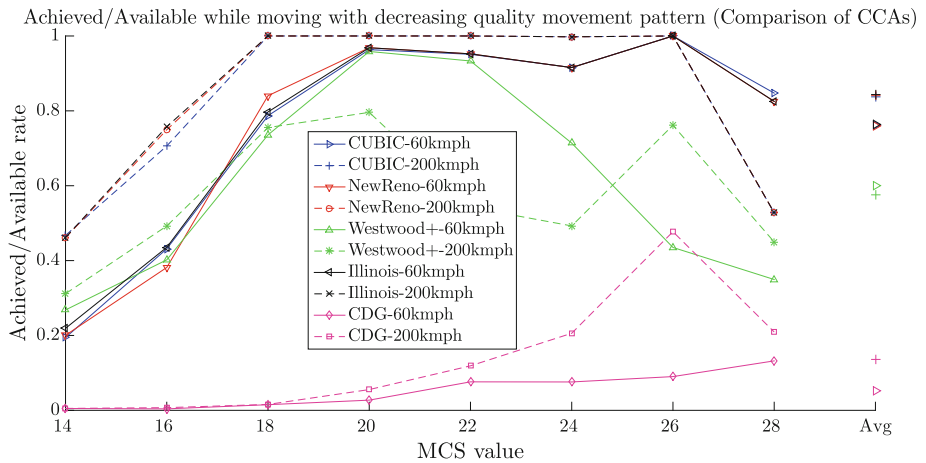
This subsection covers the analysis of decreasing quality and increasing quality movement for the selection of CCAs. Even though it has been proven in Subsect. 5.1 that some CCAs fail in mobile networks (Westwood+ and CDG), they have been kept for comparison and confirmation purposes. In order to measure the ability



or inability of distinct TCP implementation to take advantage of radio resources, the results will be presented as the portion of `tbSize` that has been actually utilized every Transmission Time Interval (TTI). In other words, since the TTI is commonly configured in 1 ms, the portion of `tbSize` will show how many bits are used for the UE every millisecond. Considering that different MCS values lead to have distinct available capacity and therefore a different achievable throughput, the analysis is divided in MCS ranges.

**Decreasing Quality Movement**

The decreasing quality movement scenario stands for the continuous movement evolution of a certain UE from the eNodeB to a further location. In other words, on average the obtained SINR due to the distance from the UE to the eNodeB and the fading will have a tendency to be worse. So will be the reported CQI and the assigned MCS (instead of worse, it is a tendency to become a more robust modulation). In such a transition, the CCA will need to adapt to the different available capacities. Figure 7 shows the difference between the available capacity and the achieved capacity for different CCAs under distinct speeds, all classified by average MCS.



**Fig. 7.** Achieved/Available capacity at different speeds for different TCP variants (decreasing quality movement).

Figure 7 clearly depicts three main areas:

Slow-Start phase: Located in the coverage zone associated to MCS 28, during the transmission establishment and first ramp-up, the *cwnd* is not great enough to take full advantage of available radio resources. Considering that Standard Slow-Start converges very fast, the MCS 28 area also takes the first back-off application. For that reason Westwood+ or NewReno among others do not report the same result. Since the distance associated with a MCS is covered a lot faster

at 200 km/h, the *cwnd* has no time to grow quickly enough and therefore the impact of ramp-up is more significant for the scenario at 200 km/h, prompting a lower value of achieved/available for this speed.

“Bufferbloat” area: While in the area between MCS 26 to 18-20, the CCAs are able to take advantage of already enqueued packets in the eNodeB (*bufferbloat* effect). However the effect itself has a drawback in relation to the inflicted delay. This feature is more present in the examples at 60 km/h. The time spent in each MCS area makes it possible to the TCP variant to inject packets throughout a longer time, getting loss packets and requiring to recover from them under high-delay conditions and therefore, not allowing the CCA to achieve maximum capacity.

Queue draining zone: Regardless the speed, it is an area in which the radio conditions are not good enough to maintain a full utilization of resources. Even though the average MCS values are between 18 to 14, fading conditions force the eNodeB to operate with very low MCS values (achieving sometimes MCS 4 and 6) in some drastic fades. With each sudden fade, it is easier to receive more robust modulations, leading the packets to need stronger segmentation. As a side effect, both the queue size of the eNodeB and the delay increase. The recovery of losses in such network conditions is also a harmful process for TCP that leads to queue starvation events. When it comes to faster UE scenario, the eNodeB is able to lengthen the utilization of previously enqueued packet to further positions, therefore, the draining effect is slower or at least happens in further positions.

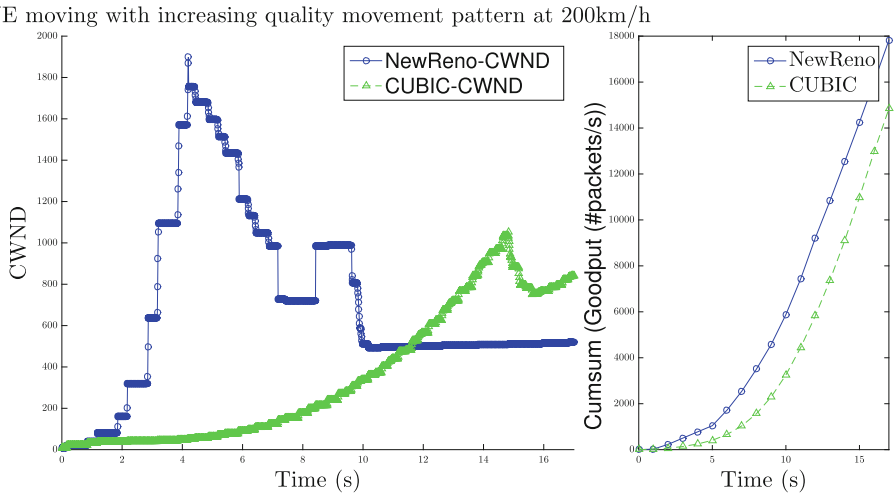
Figure 7 shows that in decreasing quality movement the differences in loss-based CCAs are minimum, getting more credit of aggressiveness at 60 km/h and RTT-synchronization at 200 km/h (NewReno and Illinois over CUBIC). Once again and even in a scenario that moves towards worse radio position, the delay-based variants have demonstrated to be unable to cope with the delay variability of LTE. CDG maintains a RTT close to the baseline RTT but underutilizes most of the assigned bandwidth. In the case of Westwood+, even though it is a scenario that helps get the maximum capacity to the weak AIMD mechanisms due to its continuous achievable capacity reduce, it takes very long time to achieve such a task at 60 km/h and it is not capable of doing so at 200 km/h.

### Increasing Quality Movement

Once analyzed the decreasing quality movement and the behavior of different CCAs under distinct speeds, it is necessary to study the increasing quality movement in a constant evolution of the channel quality to better positions. Considering the findings in decreasing quality movement, it is important to determine whether the different methods of Slow-Start equally struggle under challenging radio condition or not and analyze whether the aggressiveness of TCP overshoots sufficient packets to serve a continuous greater capacity or not.

Trying to better explain the effects of this scenario in the beginning of the transmission and the relation between *cwnd* evolution and achieved goodput, Fig. 8 represents the relation between them. The graphs have been split for the

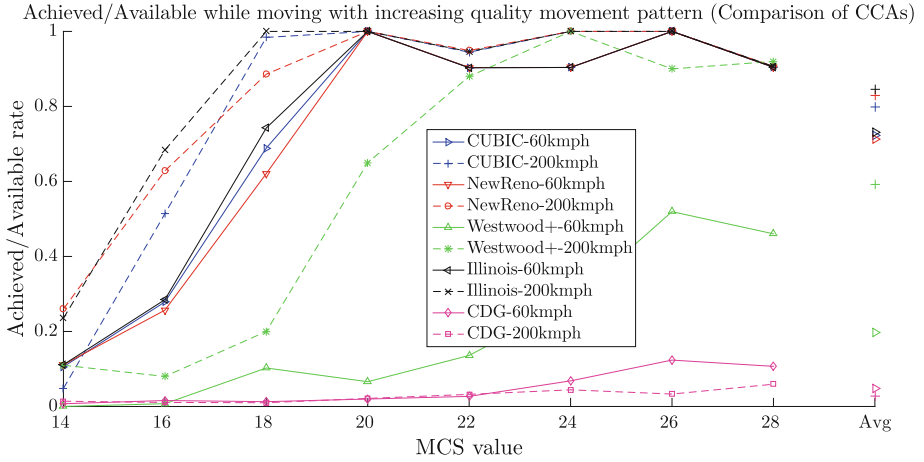
better understanding in two blocks: the result in relation to the *cwnd* evolution is on the left and goodput's cumulative sum on the right. Figure 8 depicts the behavior difference between CUBIC with Hybrid Slow-Start and NewReno with Standard Slow-Start. It is clear that the network conditions are challenging because even in Standard Slow-Start the shape of the *cwnd* is very stepped. In such conditions in which the delay variability is also a hard drawback to tackle, the Hybrid Slow-Start mechanism detects an increment in the delay that is considered enough to trigger an early exit to congestion avoidance phase. The resultant cumulative goodput of both CCAs is represented on the right where the graphs shows a big outcome gap between both methods. Once again the underperformance of Hybrid Slow-Start is shown. Besides, in this case the early exit of fast ramp-up is provoked in a single-UE scenario in which the movement and fading are the only sources that vary the delay.



**Fig. 8.** NewReno vs. CUBIC in increasing quality scenarios at 200 km/h.

Considering the explained effect regarding how Hybrid Slow-Start could affect the performance, we will now proceed to study the performance differences under different speeds between NewReno, CUBIC, Westwood+, Illinois and CDG, classified by average MCS levels (see Fig. 9). At a first glance, the figure looks very similar to Fig. 7, but some differences are present. The behavior of such scenario is divided in two areas.

Ramp-up phase: The hardest radio conditions for the channel are present from MCS 14 to 18. In such a challenging conditions the CCAs initialize the transmission and employ the selected Slow-Start method in a try to ramp-up and convergence as fast as possible without inducing a bursty loss event. As seen beforehand, at 200 km/h the performances of Standard Sow-Start and Hybrid



**Fig. 9.** Achieved/Available capacity at different speeds for different TCP variants (increasing quality movement).

Slow-Start are completely different, leading to a better utilization of the network resources in the case of Standard Slow-Start. Besides, in MCS 14 in some occasions it is not only present the Slow-Start phase but part of the first back-off and application of congestion phase as well. The growth limitation is comparatively very similar for 60 km/h and 200 km/h during this phase and establishes an undodgeable boundary for loss recovery. However, in faster scenarios the time spent in weakest radio conditions is less and the impact of such challenging conditions is less significant in the final outcome. Apart from that, in the case of Standard Slow-Start, at 200 km/h the first loss event will happen in better radio conditions than for 60 km/h and therefore, the ability to recover the lost packets is greater at 200 km/h.

Stationary area: Throughout MCS 20 to 28, TCP is able to take close to full advantage of available capacity. However, it has to be mentioned that, due to that transition speed and applied back-offs while recovering from losses, the CCAs are not able to rise sufficiently the *cwnd*, causing some channel underutilization.

Even though, in general, the CCAs follow the identified phases, there are some differences among the CCAs that result in a distinct outcome for the same network conditions (see the wrap-up Table 6).

Different Slow-Start methods affect the availability to take full advantage of radio resources, dividing the performance in two major groups. (1) Among the CCAs with same Slow-Start phase, some differences appear in MCS 14 due to the different AIMD policy applied when a loss is detected. As stated before, the higher speed, the comparatively longer Slow-Start phase and therefore, a decrease of the loss recovery effect due to the recovery taking place in better radio conditions. (2) For Hybrid Slow-Start mechanism, a difference between CUBIC and CDG appear regarding the delay sensitivity to quit fast ramp-up

**Table 6.** Findings wrap-up in mobility scenarios

CCA	Behavior
CUBIC	It only suffers the impact of Hybrid Slow-Start in the very beginning of the transmission in increasing quality movement pattern under high speed (see Fig. 8)
NewReno	Very good performance in terms of achieved available capacity in simplified single-user mobility scenarios
Illinois	The best results due to the combination of the delay-awareness and the aggressiveness
CDG	It has demonstrated very weak performance over cellular access under mobility in terms of bandwidth utilization
Westwood+	Only able to reach full utilization after a long ramp-up period in decreasing quality movement at 60 km/h and in increasing quality movement at 200 km/h

(as stated in Subsect. 3.1). In relation to the effect of speed, the faster the UE moves, the higher delay variability and therefore quicker skip to a slow increase phase, suffering more wasted bandwidth utilization at 200 km/h.

Westwood+ and CDG have been proven to be not adequate for mobile networks. The former is able to reach full utilization after a long ramp-up period in decreasing quality movement at 60 km/h and in increasing quality movement at 200 km/h. The time spent is due to a poor available bandwidth estimation and consequent drastic back-off policy. At 200 km/h in decreasing quality scenario the CCA does not allow sufficient time so as to achieve the maximum bandwidth. On the contrary, in increasing quality movement, the fastest scenario allows the CCA get the maximum capacity. The latter has demonstrated very weak performance over cellular access. It has to be underlined that the main objective of the CCA regarding the control of end-to-end delay is fulfilled, however, regardless the speed and scenario, the CCA has not been able to rise to the 10% of the available capacity, consequently leading to a 90% of resource underutilization. Therefore CDG is not suitable for mobile network as is configured now.

The group formed by NewReno, CUBIC and Illinois have shown a very successful performance regardless the speed and movement pattern. As stated in previous explanation, the shortening of the challenging periods could make a difference in terms of greater achieved capacity. On average (see average values on the right) at 60 km/h 3 CCAs are very similar and it is only under 200 km/h speed circumstances when CUBIC performs poorly due to Hybrid Slow-Start and Illinois get a slight advantage of its delay-awareness to make the most of using available resources.

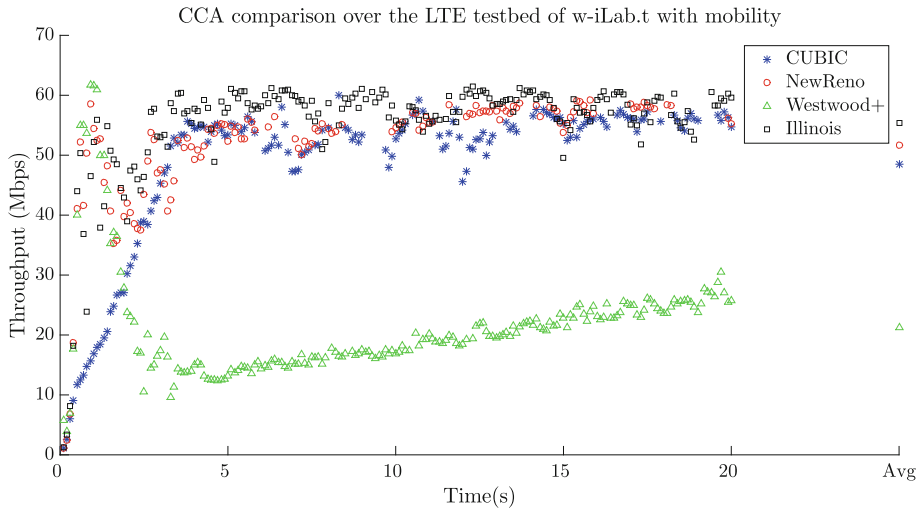
All the gathered results are consistent with the findings regarding decreasing quality movement (in Subsubsection. 5.3), the performance of different Slow-Start methods (in Subsect. 5.2) and the preliminary analysis in regards to the impact of different cross-traffic in the performance of CCAs. However, since the results

have been obtained in a single LTE deployment, it is important to determine to which extent our findings could be extrapolated as a general-purpose behavior of CCAs and whether the results are biased towards the simulated/emulated testbed or not.

### 5.4 Correlation of TCP Behavior over w-iLab.t LTE Testbed

The current subsection aims at representing and explaining the behavior of a selection of CCAs over the controlled LTE testbed called w-iLab.t. Since the deployment is formed with completely real equipment (i.e. UEs, eNodeBs, femtocells, servers), the internal mechanisms of LTE and the interaction with TCP are closer to real-world behavior and therefore the variability is presumably higher comparing with simulated environment. Thus, such testbed allows carrying out experiments that represent the performance of the reality in a smaller scale. CDG was removed from the comparison set for its incompatibility with mobile networks. Westwood+ is kept in the selection of CCAs to confirm or deny the underperformance under more variable circumstances.

We configured three different paths to be followed by the robots with decreasing quality and increasing quality movements. The location of those movement patterns were located in different places of the femtocell, having a pattern close to the eNodeB, another one close to the spacial limits of the testbed and a third one in the middle of the previous two. After ten experiments over the different configurations/patterns, we gathered the following average throughput values for CUBIC, NewReno, Westwood+ and Illinois.



**Fig. 10.** CCA comparison over w-iLab-t under mobility circumstances.

Figure 10 shows that the previous findings in ns-3 were accurate enough to explain the possible effect of CCAs in other LTE deployments. In fact, some deficiencies such as the ones regarding Hybrid Slow-Start and Westwood+ are more harmful than in simulation environment, causing a greater gap between the available capacity and the achieved one.

In general three are the most important features to be underlined. First, the drastic back-off application of Westwood+ leads the CCA to be incapable of achieving the maximum capacity even within 20 s of transmission. Looking at the growth tendency, the CCA may well take around 1 min to convergence which is an unacceptable value in order to provide a good service to the UEs. Second, the underperformance of Hybrid Slow-Start is more remarkable in this testbed and the results prompt a convergence time around 4.5 s. The performance difference with Standard Slow-Start (present in NewReno and Illinois) could be cushioned if the transmission is long enough (average value of CUBIC is close to NewReno or Illinois in 20 s transmission). However, the impact in short-lived flows would be more notable. Third, the performance of NewReno and Illinois are very similar and the only distinction appear due to the greater aggressiveness of Illinois for its delay-awareness. Nevertheless, the utilized femtocells give a very good channel quality regardless the mobility pattern, movement patterns or speed. Thus, the “signal quality rings” that are present in real-world could not be represented. Therefore, in order to better understand the performance tradeoff of CUBIC, NewReno and Illinois in congestion avoidance phase during mobility circumstances, an additional analysis was demanded.

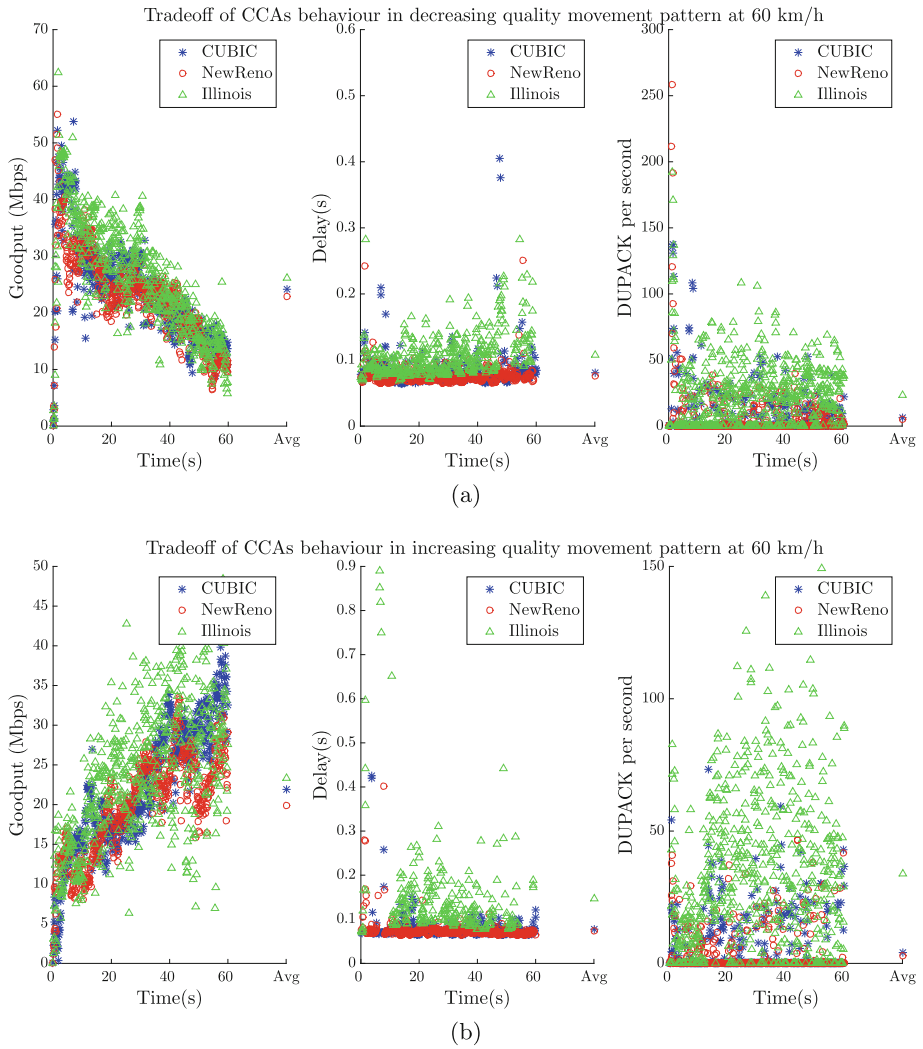
### 5.5 Performance Tradeoff of Selected TCP Variants Under Mobility in Emulated Testbed

Once the previous findings regarding the behavior of CCAs have been demonstrated in a controlled testbed, this subsection covers the comparison analysis of most adequate TCP flavors (CUBIC, NewReno and Illinois) over emulated testbed with mobile scenarios of decreasing quality and increasing quality movement. The previous scenarios have shown that CUBIC, NewReno and Illinois have a very close outcome. Therefore, this subsection will serve not only as a confirmation step of the findings in another testbed but to also carry out experiments in mobility circumstances with a realistic representation of “signal quality rings”.

The testbed itself is not able to emulate movement due to the fixed position of the UE attached to a radio cable. Nonetheless, a computer that plays the role of a experiment controlled, is capable of establishing the baseline SINR at any moment. Besides, the lte-in-a-box called Aeroflex 7100 applies a fading pattern to such a variable baseline SINR, modelling this way the effect of movement with a static UE.

To help decide the best timing for different baseline SINR values, averaged SINR traces obtained from ns-3 with a UE moving in decreasing quality and increasing quality movement patterns at 60 km/h are used. In order to give more realism to the experiments, the EVA60 fading model in Aeroflex 7100 are applied.

We have decided to only use the scenarios at 60 km/h due to the result equality in ns-3. At 200 km/h the differences among CCAs were noticeable. Therefore, these experiments add additional information to the previous inconclusive outcomes and gives more insight regarding the differences among the selected CC. Figure 11 depicts the average goodput, end-to-end delay and duplicated ACK (DUPACK) events per second as a sign of congestion for decreasing quality movement at the top and for increasing quality movement at the bottom.



**Fig. 11.** Performance tradeoff of CCAs in the emulated testbed at 60 km/h: (a) Decreasing quality movement; (b) Increasing quality movement.



In this case, the differences among the CCAs are noticeable for both movement patterns. The goodput results do not prompt any new feature and classify the performance of the selected TCP implementations from better to worse as Illinois, CUBIC and NewReno. This outcome equally applies for decreasing quality and increasing quality movement, getting slightly more difference in increasing quality movement due to the continuous capacity increase and the availability to cushion overshoots. The simplest way to proceed would be to say that Illinois is the best amongst the CCAs. However, depending on the performance objective, the decision could be another one. The reasons are manifold. First, even though the goodput performance is better for Illinois, it induces delay and the consequent packet losses are a way larger than in the examples of CUBIC and NewReno. Second, if we compare the overall performance of CUBIC and NewReno, we see that in spite of the delay and DUPACK events being very similar, CUBIC makes the most in terms of goodput. Therefore, trying to avoid massive packet losses and delay infliction, the selection of CUBIC would be more desirable in this simple comparison. Third, for comparison purposes, since the objective of this scenario was the understanding of congestion avoidance phases and the adaptability to mobile LTE scenarios, the Hybrid Slow-Start mechanism was disabled. Taking into account this detail and depending on the requirements of the application, the selection of NewReno could not be discarded. To conclude this tradeoff study, it is clear that Illinois, CUBIC and NewReno have very similar results, but it cannot be easily decided whether one is better than the other because each of them has its “bright side” and drawback.

## 6 Conclusion

This chapter has tried to shed some light in the explanation of CCAs adaptability to different mobile network situation including the implication of different type of cross-traffics, the start-up phase and mobile UEs with increasing quality and decreasing quality movement patterns. The chapter has also included different LTE deployments so as to confirm and clarify the obtained result in the simulated environment. Table 7 wrap-ups the detected findings and confirmations of CCAs behavior under distinct circumstances.

Simple static experimentation with different background traffic profiles and behaviors has demonstrated that loss-based TCP mechanisms reach the maximum capacity quicker than delay-based variants. The former achieves greater throughputs but fails limiting the standing queue size and therefore inflict severe delays. The latter is able to keep the end-to-end delay close to the baseline delay value but struggle to ramp-up or speed up its injection pace, wasting this way a great amount of the available radio resources.

Different scenarios have shown the huge impact that Hybrid Slow-Start mechanism has under some delay variability circumstances. Having in mind that the delay’s instability is one of the main features in mobile networks, Hybrid Slow-Start is capable of slowing down the start-up phase leading to a bad resource utilization. Taking into account the widespread usage of CUBIC due to the

**Table 7.** Findings wrap-up

CCA	Simulated env.	Controlled testbed	Emulated testbed
CUBIC	It suffers from its delay sensitivity in Hybrid Slow-Start phase, being very harmful and provoking mobile network capabilities underutilization	Confirmed behavior of Hybrid Slow-Start with even greater impact. Long transmission would suffer such effect but it would be more significant in short-lived ones	In simplified mobility scenarios, the cubical congestion avoidance phase allows a good available capacity utilization while the delay is lower than with Illinois (closest CCA in terms of goodput)
NewReno	I has responded very positively to different network situation, showing that it is still a good TCP candidate to be utilized in certain situations. Its speed weaknesses in fixed networks could result in a valuable feature in mobile networks	Confirmation of the good performance	Some precise mobility circumstances have shown a deficient performance of NewReno leading to resource underutilization and may well indicate which mobile network circumstances are not suitable for the protocol
Illinois	Very similar to the performance of NewReno with bigger impact in delay due to its greater aggressiveness. Such aggressiveness allows performing slightly better in scenarios that require rapid adaptability (under mobility)	Overall performance of Illinois has been demonstrated, showing in close-to-the reality scenarios better performance than NewReno in terms of achieved throughput	Under mobility circumstances, a slight gap increment in the outcome of Illinois and NewReno has been found. The results may indicate that under more realistic conditions the breach will be even greater
CDG	It has demonstrated very weak performance with all scenarios over mobile networks in terms of bandwidth utilization. However it has shown a good control of the delay keeping it close to the baseline delay	–	–
Westwood+	Found a problem with a drastic back-off application that is capable of provoking underutilization of the radio resources under certain network situations	Confirmation of the findings noticing even greater impact of the deficiency. The closer to real-world, the poorer assessment of the available capacity and therefore, the more deficient the application of the back-off policy	–

presence of it by default in most Web servers, the problem is even worse. Even though, long transmissions suffer the impact of the underperformance of Hybrid Slow-Start, the effect is greater in the case of short-lived flows.

Regarding the mobility scenarios, two have been studied. In decreasing quality movement most CCAs are able to achieve the maximum capacity during good radio conditions and they lengthen the utilization of previously enqueued packets while running towards worse channel qualities. At higher speeds, the already enqueued packets are driven to further positions comparing with lower speeds, improving the average capacity utilization. In increasing quality movement, regardless the speed, the transmission initialization and first ramp-up happens in very challenging radio conditions, requiring CCAs availability to scale, recover from losses and AIMD mechanisms' suitability to make the most of available capacity.

In relation to the specific features of each CCAs' adaptability, several findings have to be mentioned: (1) CUBIC suffers from its delay sensitivity in Hybrid Slow-Start phase, being very harmful and provoking mobile network capabilities underutilization. (2) CDG keeps the delay close to the baseline delay value but is incapable of growing its pace in order to utilize greater capacities. In its current state is not suitable for mobile networks and it could more suitable for wired networks where the delay variation is not that abrupt. However, the delay boundaries of the protocol may well be adapted to cellular networks' constraints. (3) Westwood+ has shown to be incapable to properly estimate the available bandwidth, leading to big *cwnd* reductions and the necessity to grow-up from very low values and very weak AIMD incremental pace. The adaptation of the estimation is required in order to make it suitable for mobile networks. (4) NewReno and Illinois have demonstrated to beat the other CCAs (apart from CUBIC in some situations) under different loads, traffic patterns, mobility and speed contexts. Even though in simulated environment the only detected difference has appeared in increasing quality scenario in which the delay-awareness and greater aggressiveness has given to Illinois the best performance regarding the use of available capacity, in emulated testbed the differences have been also present in decreasing quality movement. Since the emulated testbed has shown a slight gap increment in the outcome of Illinois and NewReno, the results may indicate that under more realistic conditions the breach will be even greater.

The feature-based findings have been confirmed over the LTE deployment of w-iLab.t and the performance tradeoff of the best CCAs has been explained under mobility circumstances in order to give insights regarding the appropriate selection depending on the application requirements. This chapter has given an overview of the behavior of the different TCP mechanisms in a LTE network under different circumstances. This work might be of value as a validation of the performance of different CCAs and as an indication of fruitful directions for the improvement of TCP congestion control over cellular networks.

Some knowledge from the network would help TCP decide the best strategy in accordance to the network conditions. The envisioned scenario is aligned with the main features of mobile edge computing (MEC) management that would

allow removing as much end-to-end TCP variant dependency as possible. In the same way, other initiatives such as QUIC [24] that propose transport services in the user-space of the operating system with TCP-alike CCAs on top of UDP (UDP as a substrate) could take advantage of this comprehensive analysis in order to select the most appropriate TCP candidate (i.e. depending on multi-criteria that considers both network state and application requirements) in each network conditions and enable such TCP-alike implementation.

**Acknowledgments.** This chapter has been possible thanks to the Cost Action IC1304 through the STSMs entitled “Evaluation of QoE-optimized transport protocols on cellular access” and “Evaluation of modern transport protocols over iMinds LTE facilities”. The work has been partially funded by the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 644399 (MONROE) through the open call project MaRiL and by the Spanish Ministerio de Economía y Competitividad (MINECO) under grant TEC2016-80090-C2-2-R (5RANVIR). The views expressed are solely those of the author(s). The authors would like to thank Rémi Robert for the useful discussions on the experiments and his work and dedication in the simulated environment.

## References

1. Cisco: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021. Technical report, Cisco (2017)
2. Callegari, C., et al.: Behavior analysis of TCP linux variants. *Comput. Netw.* **56**(1), 462–476 (2012)
3. Huang, J., et al.: A close examination of performance and power characteristics of 4G LTE networks. In: *MobiSys 2012*, pp. 225–238. ACM, New York (2012)
4. Huang, J., et al.: An in-depth study of LTE: effect of network protocol and application behavior on performance. *SIGCOMM Comput. Commun. Rev.* **43**(4), 363–374 (2013)
5. Merz, R., et al.: Performance of LTE in a high-velocity environment: a measurement study. In: *AllThingsCellular 2014*, pp. 47–52. ACM, New York (2014)
6. Johansson, I.: Congestion control for 4G and 5G access. Internet-Draft draft-johansson-cc-for-4g-5g-02. IETF Secretariat, July 2016. <http://www.ietf.org/internet-drafts/draft-johansson-cc-for-4g-5g-02.txt>
7. Alfredsson, S., et al.: Cross-layer analysis of TCP performance in a 4G system. In: *SoftCOM*, pp. 1–6, September 2007
8. Park, H.S., et al.: TCP performance issues in LTE networks. In: *ICTC 2011*, pp. 493–496 (2011)
9. Nguyen, B., et al.: Towards understanding TCP performance on LTE/EPC mobile networks. In: *Proceedings of AllThingsCellular 2014*, pp. 41–46. ACM, New York (2014)
10. Alfredsson, S., et al.: Impact of TCP congestion control on bufferbloat in cellular networks. In: *WoWMoM 2013*, June 2013 (2013)
11. Garcia, J., et al.: A measurement based study of TCP protocol efficiency in cellular networks. In: *Proceedings of WiOpt 2014*, pp. 131–136 (2014)
12. Ha, S., Rhee, I.: Taming the elephants: new TCP slow start. *Comput. Netw.* **55**(9), 2092–2110 (2011)

13. Garcia, J., et al.: Examining TCP short flow performance in cellular networks through active and passive measurements. In: *AllThingsCellular 2015*, pp. 7–12. ACM, New York (2015)
14. Chen, Y.C., et al.: Measuring cellular networks: characterizing 3G, 4G, and path diversity. In: *Annual Conference of International Technology Alliance*, pp. 1–6, December 2010
15. Wylie-Green, M.P., et al.: Throughput, capacity, handover and latency performance in a 3GPP LTE FDD field trial. In: *GLOBECOM 2010*, pp. 1–6 (2010)
16. Henderson, T., et al.: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 6582 (Proposed Standard), April 2012
17. Ha, S., et al.: CUBIC: a new TCP-friendly high-speed TCP variant. *SIGOPS Oper. Syst. Rev.* **42**(5), 64–74 (2008)
18. WeTechs: Usage Statistics and Market Share of Operating Systems for Websites (2017)
19. Hayes, D.A., Armitage, G.: Revisiting TCP congestion control using delay gradients. In: Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C. (eds.) *NETWORKING 2011*. LNCS, vol. 6641, pp. 328–341. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20798-3\\_25](https://doi.org/10.1007/978-3-642-20798-3_25)
20. Mascolo, S., et al.: TCP westwood: bandwidth estimation for enhanced transport over wireless links. In: *MobiCom 2001*, pp. 287–297. ACM, New York (2001)
21. Liu, S., et al.: TCP-Illinois: a loss and delay-based congestion control algorithm for high-speed networks. In: *valuetools 2006*. ACM, New York (2006)
22. Tazaki, H., et al.: Direct code execution: revisiting library OS architecture for reproducible network experiments. In: *CoNEXT 2013*, pp. 217–228. ACM, New York (2013)
23. Bouckaert, S., Vandenberghe, W., Jooris, B., Moerman, I., Demeester, P.: The w-iLab.t testbed. In: Magedanz, T., Gavras, A., Thanh, N.H., Chase, J.S. (eds.) *TridentCom 2010*. LNCS, vol. 46, pp. 145–154. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-17851-1\\_11](https://doi.org/10.1007/978-3-642-17851-1_11)
24. Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J., et al.: The QUIC transport protocol: design and internet-scale deployment. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 183–196. ACM (2017)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



## Author Index

- Aalto, Samuli 241  
Ariş, Ahmet 337  
Arvidsson, Åke 1, 371  
Atxutegi, Eneko 371
- Barakovic Husic, Jasmina 49  
Barakovic, Sabina 49  
Beben, Andrzej 269  
Blanco, Bego 241  
Bosman, Joost W. 269  
Brunstrom, Anna 371  
Bruzgiene, Rasa 104  
Burakowski, Wojciech 1, 269
- Cardellini, Valeria 182, 212
- Dobrijevic, Ognjen 49
- Galinac Grbac, Tihana 1, 182, 212  
Ganchev, Ivan 1, 81, 151  
Gribaudo, Marco 241  
Grinnemo, Karl-Johan 371
- Haddad, Yoram 1, 23  
Hasslinger, Gerhard 269  
Hoßfeld, Tobias 1, 23, 128
- Jarschel, Michael 23
- Kassler, Andreas 212  
Kathiravelu, Pradeeban 212  
Kecskemeti, Gabor 313  
Kertesz, Attila 269, 313  
Key, Peter 1
- Lassila, Pasi 1, 241  
Latre, Steven 269  
Leitner, Philipp 1  
Li, Zhi 104  
Liberal, Fidel 1, 241, 371  
Liotou, Eirini 23, 49, 128  
Lo Presti, Francesco 212
- Markus, Andras 313  
Marotta, Antonio 212  
Marques, Andre 313  
Melvin, Hugh 1, 23, 104  
Metzger, Florian 23, 128  
Moldovan, Christian 128
- Nardelli, Matteo 182, 212
- Oktuğ, Sema F. 337
- Passas, Nikos 128  
Pernici, Barbara 241  
Pflanzner, Tamas 269  
Pocta, Peter 23, 49, 104  
Poryazov, Stoyan 81, 151  
Poullie, Patrick Gwydion 269
- Sainz, Javier 241  
Saranova, Emiliya 81, 151  
Schatz, Raimund 49  
Schwarzmann, Susanna 49  
Siris, Vasilios A. 23  
Skorin-Kapov, Lea 23, 49, 104  
Sosnowski, Maciej 269  
Spinnewyn, Bart 269  
Stiller, Burkhard 269
- Tanković, Nikola 182  
Truong, Hong-Linh 182  
Tsolkas, Dimitris 128  
Tutschku, Kurt 1
- van den Berg, Hans 1, 269  
van der Mei, Rob 1, 269  
Veiga, Luís 212  
Voigt, Thiemo 337
- Wac, Katarzyna 1
- Zgank, Andrej 23, 104  
Zinner, Thomas 49