IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 19, 2016, accepted August 13, 2016, date of publication September 9, 2016, date of current version October 6, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2607786

SDN and Virtualization Solutions for the Internet of Things: A Survey

NIKOS BIZANIS AND FERNANDO A. KUIPERS, (Senior Member, IEEE)
Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands
Corresponding author: F. A. Kuipers (f.a.kuipers@tudelft.nl)

ABSTRACT The imminent arrival of the Internet of Things (IoT), which consists of a vast number of devices with heterogeneous characteristics, means that future networks need a new architecture to accommodate the expected increase in data generation. Software defined networking (SDN) and network virtualization (NV) are two technologies that promise to cost-effectively provide the scale and versatility necessary for IoT services. In this paper, we survey the state of the art on the application of SDN and NV to IoT. To the best of our knowledge, we are the first to provide a comprehensive description of every possible IoT implementation aspect for the two technologies. We start by outlining the ways of combining SDN and NV. Subsequently, we present how the two technologies can be used in the mobile and cellular context, with emphasis on forthcoming 5G networks. Afterward, we move to the study of wireless sensor networks, arguably the current foremost example of an IoT network. Finally, we review some general SDN-NV-enabled IoT architectures, along with real-life deployments and use-cases. We conclude by giving directions for future research on this topic.

INDEX TERMS Internet of things (IoT), software defined networking (SDN), network virtualization (NV), network functions virtualization (NFV), 5G, wireless sensor network (WSN).

I. INTRODUCTION

With the number of devices connected to the Internet of Things (IoT) projected to expand to somewhere between 20 and 46 billion by 2020, our future will connect nearly everything, from traditional communication tools, such as laptops and smartphones, to home appliances, such as refrigerators and garage doors, and from industrial systems to actual people.

Although several solutions have been proposed and implemented to deal with a steady increase in data consumption and number of devices (e.g., the introduction of IPv6), they were not designed with *billions* of new users/devices in mind, who would join the network in a short period of time. This projected growth means that current wireless and mobile networks should evolve to become more ''intelligent,'' efficient, secure and, most importantly, extremely scalable to deal with a torrent of data communications that are extremely diverse in nature.

Software defined networking (SDN) and Network Virtualization (NV) are two of the most prominent technologies to serve as key enablers for the IoT networks of the near future. The main idea behind SDN is to separate the control plane (where the logical procedures supporting the networking protocols are executed and all the relevant decisions are taken) from the data plane (where the forwarding of packets on the most suitable interface towards the intended destination is executed). As illustrated in Figure 1, the main entity behind this separation is the controller, which communicates with the network applications through the so-called Northbound interface and translates their requirements into appropriate network decisions. The controller also communicates with the network switches that forward packets according to the controller-installed rules. This way, SDN provides increased possibilities to smartly route traffic, for example to balance the load over the network or to exploit underutilized network resources in an optimal way, thereby alleviating the burden on the network by the data onslaught of IoT.

The term network virtualization concerns a network that allows multiple service providers to form multiple separate and isolated virtual networks by sharing physical resources provided by one or more different physical network infrastructure providers. Network virtualization techniques offer reduced Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) costs by sharing the network infrastructure, improved time to market for new services and applications, as well as extreme customizability and agility by leveraging the concept of network slicing and

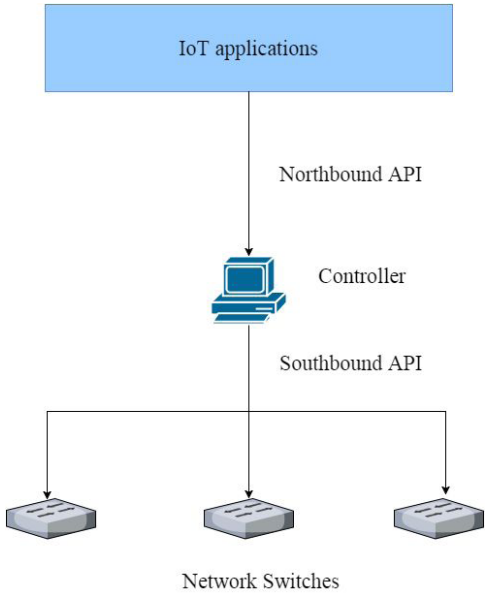

FIGURE 1. SDN layered system view.

subnet isolation. This will be a crucial feature for future IoT multi-networks, enabling diversified QoS for diverse usage scenarios and the quick introduction of new services and applications.

In the remainder of this article, we survey the literature on applying SDN and virtualization techniques to enable IoT scenarios. In Section II, the case about combining SDN and virtualization technologies is advocated, while in Section III, we distinguish the unique features of our work from existing surveys in the field. In Section IV, the implementation of SDN and virtualization technologies in wireless and mobile networks is examined. In Sections V and VI, the application of SDN and virtualization in IoT scenarios is investigated, by examining wireless sensor networks (WSNs) in the former and general architectures in the latter section. Section VII concludes the article and suggests future research directions. In the Appendix, we list the acronyms and abbreviations used throughout this survey.

II. THE COMBINATION OF SDN AND NETWORK VIRTUALIZATION

Although the notions of SDN, Network Virtualization and Network Functions Virtualization (NFV) share many common elements and the terms are often used interchangeably, they are distinct technologies that may work in combination, with SDN usually acting as an orchestrator for the slicing of networks into virtual subnets, or other related functions. The technologies, and more importantly their combination, can be used as key enablers for IoT deployments.

Sherwood et al. [1] were among the first to combine network virtualization and SDN. In their work, which aims at supporting multi-tenancy in a data center, the network is ''sliced'' into multiple logical sub-networks, each completely isolated from the others. This can be beneficial both in terms of optimizing the allocation of the available network resources and providing differentiated services to diverse users. In such a setting, each user of a virtual slice believes that it has its own dedicated network hardware, thus specific performance guarantees can be given and fine-grained service-level agreements (SLAs) can be made. Their specific architecture is called FlowVisor [1], and it consists of a virtualization layer residing between the network hardware and the software controllers, using the OpenFlow protocol to translate between them. There is one dedicated software controller for each slice, with FlowVisor acting as a referee by intercepting every message between the two layers (controllers and switches). This happens in order to make sure that each controller can only direct the forwarding elements belonging to its own slice and it does not know that it controls only a subset of the network traffic and topology.

Although the mechanisms that dictate the way that isolation is implemented are adequately discussed, the article does not specify which algorithms to use for resource allocation among slices (e.g., in case of topology isolation, how to determine the number of virtual switches dedicated to each slice). Also, it is natural that the addition of an extra hypervisor layer and the existence of multiple software controller modules add overhead and complexity to the system. This work was published in 2008 and the recent innovations in SDN and OpenFlow enable a controller, given appropriate algorithms, to perform network slicing without using a middle layer.

FlowN [2] is an alternative architecture, where in order to achieve better scalability the idea of multiple controllers (advocated in FlowVisor) is abandoned. In this scheme, virtualization in terms of containerization is proposed, so that each virtual network (VN) user will see only his network, as if it was standalone. The controller will give the user full liberty over his own custom topology, as well as an exclusive address space, by performing a mapping between the virtual address space of each VN and the global physical address space that only the controller can access. In order to distinguish and isolate the different VNs, the controller instructs the edge SDN switches to encapsulate all packets with an extra header containing a special VLAN ID field that uniquely identifies a customer VN. This way, customers are at liberty to deploy their own custom applications over their VNs, without interfering with one another. Information regarding to which virtual nodes are assigned to which physical ones, as well as the respective information about the mapping between virtual and physical links is stored in an SQL database. Using this information along with the global topology view, the controller handles the packets belonging to each VN user differently.

To deal with the issue of integrated, centralized management of heterogeneous networks, Qin et al. [3] present a tree-based hierarchical overlay network. This tree is directed by a central entity, residing at its root, which has a global view of the network by gathering network state information from

its tree descendants. Nodes are categorized based on their capabilities and their expected mobility requirements and are put on the appropriate sub-tree. The adopted algorithm tries to optimize the overall network performance, based on heterogeneous flows, separated mainly by different packet lengths and inter-arrival times.

Although it is briefly mentioned that their proposal can be extended to much more diverse packet flows, for instance to optimize the network shared by a delay-tolerant file-sharing exchange and a delay-sensitive video streaming service, there are no details on how this architecture can be practically implemented. SDN would be a natural candidate for the implementation of such a framework, with the SDN controller assuming the role of the directing entity at the root of the overlay tree structure. Also, network slicing could be used to completely isolate the different tree branches.

III. RELATED SURVEYS

The topics of SDN, virtualization, and IoT have individually received a lot of attention from the research community, but to the best of our knowledge there has not been any previous effort to survey the conjunction of the three.

In [4] and [5] one can find surveys on SDN, its innovative features, and how it can be used to facilitate future networks, while in [6] the historical path towards programmable networks is examined. Despite the fact that those articles are comprehensive in treating SDN, there is no specific mention of its applications to IoT.

Al-Fuqaha et al. [7] provide a comprehensive survey of IoT, but there is only minimal mention of virtualization and/or SDN as enabling technologies. Another recent relevant survey is given in [8], but it is very focused on virtualization for WSNs, and it does not cover more holistic IoT architectures, more complex deployment scenarios and applications, or the way that SDN can be exploited to realize them.

Two more relevant surveys can be found in [9] and [10], which study the combination of SDN and virtualization in wireless and mobile networks, but they do not target IoT applications. The work closest to this article is [11]. There, the authors survey SDN orchestration for IoT applications, but very little attention is given to virtualization, NFV, network slicing or their combination with SDN to serve as facilitating technologies for the IoT.

IV. SDN AND VIRTUALIZATION FOR MOBILE AND CELLULAR NETWORKS

Virtualization and the exploitation of a central controller for a variety of network functions are seen as key enablers for future 5G cellular networks [12].

A. ACCESS NETWORK APPROACH

Granelli et al. [13] acknowledge the benefits that the combination of SDN and virtualization can bring to wireless access networks. They study heterogeneity in such networks, in terms of interoperability issues between different wireless technologies and standards. To address this challenge they propose the use of NFV, which can lead to more efficient network management in the 5G world. As an example, they outline a technique inspired from the concept of Software Defined Radio (SDR), of creating virtual network ''slices'' to support multiple wireless protocol instances (e.g., LTE, ZigBee or WiMax), over the same hardware radio resources. This way, SDR and its related frameworks can be thought of as an extension of SDN to the wireless world. In the same manner as SDN techniques are used to declare a reconfigurable backhaul network, SDR can be used to design a radio access network (RAN) where all access points are controlled by a central entity, in order for certain functions to be optimized (radio resource allocation, load balancing, handovers). Another, somewhat more ambitious proposal is for this integrated SDN network management framework to be applied over a mixed wired-wireless scenario. This is the case for future 5G networks, where extreme traffic volumes are going to be applied both to the front- and the backhaul of the system. Although the combination of SDN with NFV in future wireless access networks is expected to support the anticipated vast increase in the number of mobile devices, the heterogeneity in devices, requirements, and usage scenarios, leaves many hurdles yet to be taken.

Dely et al. [14] use virtualization and the cloud at the front-end of a wireless network as well. A new architecture is proposed, where processing intelligence is removed from the physical access points (APs) and it is passed to the cloud. The APs only serve to receive the traffic from the radio front-end and redirect the MAC frames to an SDN-enabled switch. They do not have any ability to process or generate their own MAC frames. Then, the core network takes over to process the frames in a central infrastructure, giving to the end user the impression of a single, centralized virtual AP. All decisions on how the virtual APs are formed and organized, and which MAC frame gets forwarded to which virtual AP for processing, are taken by the SDN controller. The transfer of the whole computational procedure to a virtualized central AP makes this architecture a suitable case for SDN-orchestrated network virtualization, where cloud computing is applied to the lower layers of the wireless stack. However, there is no mention of SDR techniques in this work, and no attempt to virtualize the PHY layer. A potential cause for concern regarding this design would be that virtual APs, located deep inside the cloud, could introduce large delays, as every MAC frame must be sent back and forth to be processed.

Zaki et al. [15] move virtualization down to the PHY layer, mainly in terms of spectrum virtualization. A hypervisor middleware is introduced at a base station level, and all LTE Physical Resource Blocks (PRB), are pooled. This means that each device sees a virtual base station (BS, like eNB), which will use PRBs gathered from many physical eNBs, in order for a certain Quality of Service (QoS) metric to be fulfilled. Figure 2 provides a schematic representation of this concept. This could mean that different virtual network ''slices'' can be created in the physical layer, based on certain SLAs. One application of this concept is SLA-level differentiation for

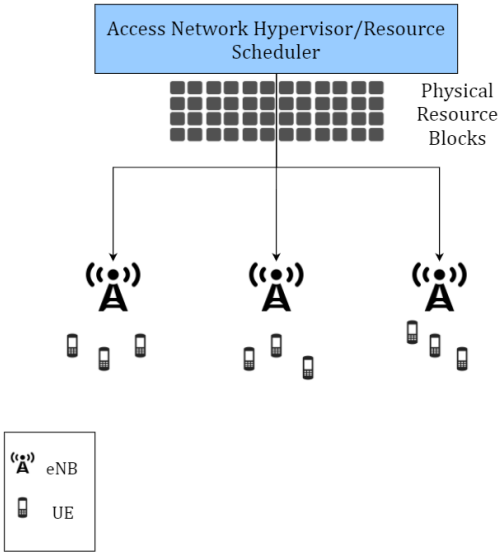

FIGURE 2. The architecture as presented in Zaki et al. [15].

different IoT usage scenarios. While a hypervisor is proposed to control the eNBs, at the time the paper was written the concept of SDN was still in its infancy and OpenFlow did not yet exist. Consequently, an extension of this concept to higher layers of the network stack is possible, by involving the core backhaul network directly, using an SDN controller to orchestrate the resource allocation.

In [16], the V-Cell design is introduced. This scheme also relies on pooling the resources of several cells in such a way that the end devices can only a see a single, central macrocell. Interestingly, the authors take it one step further by introducing the concept of no-handover zones. According to this scheme, every end-user is assigned to a virtual BS, which follows the mobility of the user along the whole system, by properly assigning resources to the user, even if those resources belong to different physical BSs. Thus, the need for handover signaling is eliminated, which can be beneficial to future wireless networks that likely have a very large number of small cells, making handovers very frequent. Thus, even more resources are freed for useful packet transmission. Still, there is no mention of any algorithm to accommodate heterogeneous QoS requirements by slicing the common radio resource pool.

Gebremariam et al. [17] apply a similar SDN technique, targeting interference mitigation, instead of mobility. The central controller gathers transmission-related information from the transmitting BSs, such as transmitted power, code rates, number of antennas used (if MIMO is applied) and the modulation and coding method used. Using the information collected, the controller has a complete, global network view and it computes an Interference Graph. In this abstraction of the network, different BS to user equipment (UE) communication links are represented by the nodes of the graph, and two nodes are connected by an edge if the respective links are within the interference range of one another. Based on this data structure, the controller can respond to every resource allocation request from a terminal with an optimal set of resources granted, so that total interference in the system is minimized.

B. CORE NETWORK APPROACH

The introduction of SDN and NFV can also bring significant benefits to the core network. Dynamic control of mobile traffic flows through the mobile packet core can be achieved, and directed to the appropriate gateways, according to the network conditions. Also, the virtualization of critical core elements can offer flexibility and cost reduction for a provider. An illustration of this concept can be seen in Figure 3, where the core network functional elements are deployed in the premises of a cloud infrastructure provider, while the access network remains under control of a traditional network provider. A network hypervisor is used at the border between the two, to translate network requirements to specific tasks in the cloud.

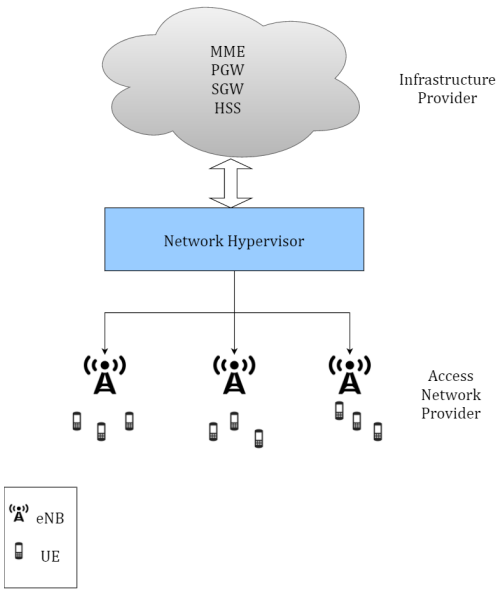

FIGURE 3. Core network virtualization.

Such an approach is taken by MobileFlow [18]. There, a special Software-Defined Mobile Network (SDMN) is outlined that virtualizes the network, under the orchestration of an SDN controller. The network infrastructure is sliced, under the supervision of the controller and the various network elements, such as routers and switches are virtualized using the concept of the virtual machine (VM). For instance, in long-term evolution (LTE) networks, entities such as the Mobility Management Entity (MME), Home Subscriber Server (HSS), Packet data network Gateway (PGW), and Serving Gateway (SGW), are fully virtualized, such that each user, or group of users thinks it has its own dedicated and isolated network hardware. In addition, flow-level forwarding is provisioned, which leads to carrier-grade service chaining. A drawback is that this structure is not based on OpenFlow,

but on a custom protocol that the authors do not describe in detail. Instead, a high-level architectural framework for the SDN-guided virtualization of a wireless network is provided. The authors do not delve deeper into how specific applications can be built to exploit the flexibility that is offered. A natural application is to explore network slicing for heterogeneous QoS requirements of IoT.

Basta et al. [19] study the benefits of virtualization for the evolved packet core (EPC) network of an LTE system. The authors investigate the effect of virtualizing the main core network entities of LTE, such as the MME, PGW and SGW, based on general-purpose hardware residing in a data center. The OpenFlow protocol is enhanced by a GPRS encapsulation module. They also study different placement configurations of that module inside the virtual network and their effect on signaling traffic load and packet delay.

A complimentary work is outlined in [20]. The authors claim that the current LTE/EPC architecture (4G) lacks elasticity, which leads to complicated management procedures, practices such as vendor lock-in, and reduced flexibility to introduce new services. They propose both SDN to achieve dynamic management of traffic flows and NFV to move the central n etwork entities, such as the MME, to commodity hardware in the cloud, thereby achieving lower OPEX. OpenFlow is used to handle the Mobile Packet Core (MPC) switches in order to manage the addition, on-line modification, and deletion of connections between the access network and the Internet. In this way, services - such as QoS provisioning or billing and accounting - can be deployed in a flexible manner. Further work must be done to extend this architecture and to provide the implementation of the specialized service modules, such as online charging or reliable routing in case of failures. For work on resilience in SDN we refer to [21] and [22].

Jin et al. [23] present another framework, where SDN is used in between the edge and the core network of a cellular system. A logically centralized controller is used to support fine-grained policies, such as firewalls, or as a transcoder for multimedia transfer. The controller handles the RAN switches, which are co-located with the BSs. In order to introduce specific service policies, it combines the marking of the packets with VLAN tags as well as MPLS labels. Although this is interesting work in the field of SDN and its application to mobile networks, the whole framework is still based on conventional protocols, i.e. standard IP routing and MPLS. The introduction of special SDN-related standardized protocols, such as OpenFlow would be a natural extension, in order for routing to be centrally controlled throughout the network.

C. SYSTEM-LEVEL APPROACH

A more holistic framework, which, unlike the articles described above, is not limited in either the RAN or the CN, can be found in [24]. The author proposes to introduce NFV as the main enabling technology for a telco carrier cloud, spanning the whole of the carriers network stack. In this scenario, operators will deploy their network, with emphasis on the core network elements, such as the MMEs, in the cloud, by renting computing space from a data-center provider.

The telco cloud is expected to provide the necessary adaptability/flexibility to accommodate novel usage scenarios. It is also expected to reduce the CAPEX of a mobile operator, as the same network physical infrastructure can be quickly re-configured to expand or shrink the carriers network on-demand, depending on the traffic load imposed. The suggested architecture is end-to-end oriented, and it aims to jointly optimize the core network and the RAN. The author divides the telco cloud framework into 5 distinct layers:

- A Physical Infrastructure layer, which contains the physical infrastructure of the data center that is used to accommodate the telco cloud.
- A Virtual Infrastructure layer, which abstracts the Physical layer from the application, serving as a hypervisor.
- A Carrier Cloud Service platform, which ensures that the virtualized core network, and the respective virtualized RAN of the same provider are interoperable.
- A Service Provider layer, which provides the necessary high-level APIs of the services that are visible to the end users. A Content Delivery Network (CDN) or a custom Firewall could be examples of such services.
- A User layer, which describes the different usage scenarios and the ways that those services can be consumed by the end user.

Although the article touches the subject of machine-type communications, which is related to the IoT concept, it does not discuss how NFV can be used to accommodate heterogeneous QoS requirements from the different use-cases. There is also no specific mention of any algorithms that can be used in order to translate a flow's QoS requirements into a specific network configuration based on virtual machines to provide isolation between those different usage scenarios.

Virtualization and SDN are also utilized in [25] to compose a novel proposal, called SoftAir, for the 5G cellular network. Some of the key aspects of the proposed architecture are:

- High network flexibility, through the introduction of NFV. This happens both in the core network, where functions such as mobility management, QoS routing and billing are moved to the cloud, as well as in the RAN, where the baseband processing units are fully virtualized and completely decoupled from the radio hardware, bringing both cost reduction and enhanced cooperation capabilities among different RANs.
- Network slicing, where multiple isolated subnetworks are dynamically allocated to different network entities, and where each entity can use its own PHY/MAC/NET layer protocols to provide customized services to their customers.
- Central SDN-enabled cloud orchestration. This includes (1) a mobility-aware control traffic management module, which selects the optimal routing paths for control traffic between the RAN and the CN, (2) a distributed

traffic classification function, which identifies the application, the stochastic features of the traffic carried, and its QoS requirements, and (3) a resource-efficient network virtualization module, which maximizes the global, network-wide throughput achieved, by respecting the individual QoS demands of every flow.

Some more concrete traffic engineering schemes that can be used to realize this architecture and to provide fine-grained QoS differentiation for various tenants with separate requirements include:

- Centrally-coordinated BS clustering, through the use of SDN at the RAN level. A group of BSs pool their transmitting antennas (RRHs) in order to act as a single antenna array, rendering this group a virtual single BS. This leads to a large-scale MU-MIMO realization [26], which ideally could lead to a single, giant BS in the network, achieving a frequency reuse of one, optimal capacity in the network, and a low level of interference.
- Throughput-optimal and QoS-aware resource provisioning. The objective here is to select for each user the most suitable combination of transmission power, modulation/coding scheme and MIMO antenna configuration so that the total system throughput is maximized, while the individual data rate and maximum tolerable delay requirements of the users are satisfied.
- A centrally-located traffic classifier. This is a module that takes as input the statistical characteristics and historical patterns (if available) of traffic flows (e.g., packet inter-arrival rates or the Hurst parameter) and groups the flows belonging to one of a set of pre-defined QoS classes, using semi-supervised machine learning algorithms.
- A new mobility management framework. A central controller is used both for location management (i.e., registration and paging) and handoff rerouting. Location management can be simplified, in terms of less signaling overhead, by the use of an SDN controller and enhanced by the exploitation of heterogeneous access networks (e.g., cellular combined with WiFi). Rerouting in case of handoff is also simplified by the use of a central controller that combines a global network topology view and per-flow QoS requirements to take optimal decisions about the establishment of the new path.

SoftAir comprises a complete system-level design, and the use of SDN and central control is proposed to encompass every aspect of a 5G cellular system, from mobility control to end-to-end QoS performance guarantees. It would be interesting to realize some of the proposed algorithms and integrate them into a single system at an experimental scale to assess their combined performance in practice.

An interesting fusion is proposed in [27], which combines the concept of Information-Centric Networking (ICN) with virtualization, and more specifically network slicing, in the context of a wireless network. ICN uses caching of information chunks in intermediate nodes of the network. This means that desirable content moves from the actual server to a location closer to the end user who wants it. The authors make the observation that in the same way that the physical network hardware (or any other kind of physical resources for that matter) is virtualized to be shared among multiple parties, the same logic can be extended to the content that circulates over the network.

They propose three different ways of slicing: Network-level slicing, which encompasses both access and core network virtualization. Flow-level slicing, an example of which is the FlowVisor framework analyzed above and where different flows get a separate slice of the network, e.g. in terms of bandwidth or time-slots assigned. Finally, and this is the novelty of this article, they outline content-level slicing, where the physical hardware of the content cache is virtualized and the content gets split into multiple slices, each given for consumption to a different user that requests it. The authors mention that by using an SDN controller, the physical network operator can perform optimization of the mapping between the physical resources available and the virtual resources granted to a requesting service. Nevertheless, they do not provide any methods or algorithms to achieve this optimization.

A similar fusion was presented in [28]. Here, the authors focus on Named Data Networking (NDN), which is a specific architecture that implements the broader ICN concept. Their approach, called NDNFlow, is based on using SDN to set-up and facilitate ICN networks. In order to do so, instead of modifying the OpenFlow specifications, they opted for the addition of a separate, parallel ICN layer, by implementing the necessary controller module. In this design, ICN flows are treated in a special way, compared to regular IP flows, as the forwarding path computation for those flows is done by the ICN module, separately from the traditional flows. This way the advantage of adding application-specific forwarding functionalities of ICN is combined with minimal upgrading at the network elements, where only a software plug-in must be installed to render them ICN-enabled.

The concept of using virtualization and the cloud to offer resource elasticity, adding the element of flexibility and on-demand consumption of network resources, is explored in [29]. It is suggested that ''when limited resources are offered for potentially unlimited use, providers must manage them elastically by scaling up and down, as needed.'' The authors outline different dimensions of cloud elasticity and state that the allocation paradigm can change depending on the particular dimension that needs to be optimized. For instance, ''quality elasticity'' is defined as the perceived QoS difference, depending on the level of resource usage. This can be beneficial in the case of applications with temporal variations, which in a static network would cause overprovisioning for the peak usage instances, with the resources staying idle and unexploited for the remainder of the time. Another dimension is ''cost elasticity,'' which enables dynamic pricing models. This can lead to a flattening of network demand, as customers will be charged more during peak times, thus

optimizing network usage and minimizing the maintenance cost for the network owner. A last, but not least, constraint is the amount of network resources at the operators disposal.

Only a high-level description of the model is given and there are no specifics on the implementation, namely on algorithms that can be used to dynamically translate the application QoS requirements into network resource requirements and decide how to perform the optimal allocation. The same holds for the rationale based on which the trade-off between the three competing dimensions/constraints (cost, QoS, available resources) will be resolved in each case.

Bari et al. [30] present a related work, where it is recognized that in order to support elastic services in an online fashion and maintain QoS guarantees at the same time, a special QoS policy enforcement framework must be in place. These frameworks usually operate on top of a DiffServ or IntServ architecture, but they do not operate in an online manner and they can only support a rather coarse granularity of QoS classes. Also, they require proprietary software on the network nodes. The authors therefore turn to SDN. A network manager can define a set of high-level policies that the outlined framework will translate to a set of Southbound SDN rules that will be installed in the forwarding elements. During the system operation the central controller gathers statistics from the network nodes and performs the necessary calculations to decide if there are QoS policy violations. If so, the network can be reconfigured, with the controller altering the routing paths, so that the SLAs will be maintained. For new requests arriving to the system, there is an admission control module that checks whether the addition of the new flow can be tolerated by the system, in terms of available resources. The advantage of using SDN for these purposes is that on the one hand we can have fine-grained, custom QoS-aware resource allocation for each individual flow and on the other hand we can also perform dynamic reconfiguration of the network in case new flows are added or existing flows change their traffic patterns.

V. SDN AND VIRTUALIZATION FOR WIRELESS SENSOR NETWORKS

One of the greater challenges to be addressed in the development of IoT in general and WSNs in particular is the great variety of participating devices. Already a multitude of different devices with different capabilities exist. This diversity manifests itself in terms of different capabilities (processor, memory and storage), different communication standards supported, as well as different types of sensor hardware. An illustration of a traditional (non-SDN-enabled) WSN multi-network deployment can be seen in Figure 4. The integration of SDN and network virtualization with the various sensor network architectures promises great benefits in a future where IoT will be prevalent. A major field for an early implementation of leveraging SDN and virtualization in the IoT can be found in WSNs, and there has been substantial research on interconnecting WSNs into a wider IoT framework [31], [32].

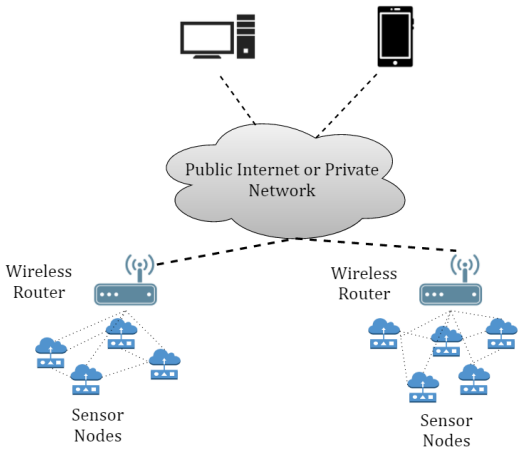

FIGURE 4. Traditional WSN architecture.

A. SDN FOR WSNs

One of the first articles that leveraged SDN and network programmability for WSNs can be found in [33]. There, a WSN consists of a BS, which contains the SDN controller, and several sensors under its command. Sensors are considered to be dumb and they do not make any routing decisions. Instead, they can only forward packets according to a flow-table in their memory, where the rules are installed by the SDN controller. The network application can then be modified by simply changing the forwarding rules at the controller, which then propagates the changes to the sensors.

In [34] the authors take a step forward by suggesting a more concrete framework. They state that application-specific WSNs are extremely rigid, in that they can only perform specific tasks and manual reconfiguration is needed if they are to be deployed in alternative ways. At the same time their management can also be hard, thus necessitating a new paradigm, where the re-tasking of general-purpose sensors will be easily performed through software. Their proposal, called SD-WSN, consists of an architecture very similar to traditional SDN. Namely, the network is clearly divided into a data plane, which includes the sensors that perform the packet forwarding, and a central controller that performs control and monitoring of the sensors (instead of switches in the traditional SDN model). The communication between the two planes is based on a modification of the OpenFlow protocol, called Sensor OpenFlow (SOF). Although no experimental validation was made in this article, it was the first concrete proposal for a synergy between the WSN and SDN worlds.

A similar attempt towards an architecture for reconfigurable WSNs, based on standardized commodity hardware, is outlined in [35]. This time the architecture is more loosely based on SDN and the OpenFlow protocol is not used. More autonomy is given to individual sensor hardware and not all network decisions are centralized. Each node has its own local low-end controller software and an embedded VM to support interoperability among heterogeneous sensor devices

and over-the-air programming for the re-tasking of sensors in accordance to the central controller commands. Thus, many routing and MAC layer decisions [36] are taken at the local level. The central controller remains only for optimization of long-term goals, and it can take decisions such as the choice of network parameters and protocols. It also possesses global knowledge about the network topology, link quality, and the requirements of the applications at the Northbound interface, to orchestrate the sensors. What is not described is the protocol through which the central controller communicates. Another concern is the suitability of low-power and limited computational capacity sensors to perform sophisticated network functions.

Galluccio et al. [37] present an alternative framework that supports stateful SDN, by implementing some degree of programmable forwarding logic in the nodes themselves. They define an SDN controller to decouple the control plane, which runs on top of the WSN, from the data plane, which is still implemented in the sensor nodes. In order to exploit the already deployed infrastructure, the SDN functions operate on top of the 802.15.4 PHY and MAC layers that are still used by the nodes for communication. Functionally, the network consists of several sensor nodes and one or more sink nodes that serve as the gateways between the low-level nodes running the data plane and the higher-level elements running the control plane. The SDN functions are separated into three distinct layers: The Forwarding layer (FWD), which processes incoming data packets according to the rules that are installed by the controller in a flow table inside each node and sends them to the next best chosen hop towards the sink, the In-Network Packet Processing layer (INPP), which runs on top of the FWD and mainly serves for data aggregation, and the Topology Discovery layer (TD), which collects information about the neighboring relationships between nodes and feeds it to the controller, so that it will have an accurate global view of the sensor interconnections. A question remains on the practicality of such stateful SDN implementations, as many of the sensors participating in the IoT are expected to be extremely limited in terms of processing and storage capabilities, and it is doubtful whether they can support such a level of programmability.

Another interesting SDN-to-WSN application is given in [38]. There, the authors argue for applying OpenFlow to Wireless Mesh Networks (WMNs), which is the most common topology found in WSN deployments. Those networks exhibit similar characteristics to traditional Wireless Mobile Networks, such as cellular or WLANs, but with the crucial difference that every node in a WMN can act as a router as well, and participate in the routing function by sending and receiving packets to/from its neighbors. Given that in many cases the sensors or other devices participating in the network are geographically dispersed, WMNs are seen as a key component for the future IoT. In the proposed architecture, the network is composed of OpenFlow-enabled mesh routers, terminal stations that connect to the routers using a standardized 802.11 interface, mesh gateways to connect the mesh network to the outside world, and the central controller. The controller installs the forwarding rules to the routers and performs the channel assignment to optimize the network operation and to minimize collisions among different node-pairs transmissions. It also handles global node addressing and mobile connectivity, in case the nodes can also move in space. Although the merging of OpenFlow with mesh networking is an interesting proposal, the authors do not provide any specific algorithms on how to leverage OpenFlow to achieve the interesting applications they outline, such as load balancing and mobility management.

A more specific example, where SDN is applied to sensor networks, can be found in [39]. There an SDN architecture is applied instead of a traditional one, in order to implement a sleep-scheduling mechanism. It is based on the so-called EC-CKN (Energy Consumed uniformly Connected K-Neighborhood) algorithm [40]. This algorithm selects the most suitable nodes to be put into sleep mode, by taking into account both the remaining power of the sensor (they usually operate on batteries) and the number of its neighbors. If a node has less than k neighbors, none of its neighbors is allowed to go into sleep mode, while if it has more than k neighbors, at least k of them must remain awake. Taking into account the remaining power is done to maximize the lifetime of the network in terms of energy, while taking into account the number of neighbors serves to ensure the connectivity of the network. The novelty of this approach is that instead of having the sensors compute the relevant algorithm parameters locally, in a distributed manner, the central controller performs all computations and communicates the decisions, on whether they must get in or out of the sleeping mode, to the sensors. The gain of this modification is that the numerous broadcast messages that the sensors had to exchange in the distributed approach are avoided. Instead, the only information exchange necessary is a brief beacon message from the node to the controller, which is directed through a pre-defined route and is not broadcasted to save energy.

B. NETWORK VIRTUALIZATION FOR WSNs

An approach that aims to exploit network virtualization to make a separation between the physical infrastructure and the applications is given in [41]. It considers a scenario where a sensor node can serve multiple sensing applications concurrently and the network contains multiple-purpose sensors that are application-agnostic. This way multiple virtual sensor networks are created over a shared physical infrastructure and this article is probably the first work where an open access API to support third-party WSN applications is envisioned. To achieve this goal, the authors opted for sensors that contain a special hardware abstraction layer, instead of deploying virtual machines to reside on top of the sensor OS inside each node (as suggested in [42]). This layer is used in each node in order to allow each separate application access to the shared hardware. When an application requires access to some component, e.g. the timer of the node, the request

is processed by a run-time layer that resides between the embedded operating system and the application, which is written in TinyOS. Once the application code is sent to the relevant nodes, network-level virtualization must be achieved and to this end an overlay network, using the Collection Tree Protocol [43], is formed to route control messages to the sensors and collect useful data from them. This way the nodes that run the same application are completely isolated from the rest of the network, even if they are not located in physical proximity. This makes each application to perceive the network as being dedicated to it only. A downside of this architecture is that it is not generic, as only applications written in the context of TinyOS are supported. Also, although the authors mention that the target nodes for each application are selected, among other criteria, according to available resources and their computing capabilities, there is no algorithm given that selects the appropriate targets in case of multiple applications requesting network resources simultaneously.

A similar system is described in [8]. There, a multi-layer architecture is proposed to enhance the efficiency of already deployed WSN hardware. This approach is based on overlay Virtual Sensor Networks (VSNs), which have been first proposed in [44]. At the bottom layer, individual sensor nodes can run several tasks concurrently. Those tasks can belong to different applications/end-users, so that the sensing capabilities of each physical node are shared, creating a virtualized sensor network. For instance, a possible scenario would be that of a sensor deployed in a residential smart home application, which will also be exploited to serve a city administration task (e.g., for a public safety project) at the same time. The homeowner can be given incentives to allow his/her hardware to be virtualized for sharing. The data gathered by the sensors are also fully aggregated and shared at a higher, network level. This creates a second abstraction layer, from which the end consumer can access the platform using a standardized interface for the communication between the end-user application and the gateway node. The whole system gives an implementation of the concept of Sensing-as-a-Service [45]. Since sensor hardware can be extremely limited in terms of processing capabilities, this design relies on special central gateway nodes (e.g., an LTE eNB).

Although plenty of technical details are given for the platform and its implementation, there is no definition of a mechanism that will discover and allocate the required physical resources [46] and which will assign the tasks to the sensors in such a way that the performance requirements of each task are fulfilled and the overall number of tasks admitted to the system gets maximized. This becomes even more imperative in the case of different applications that require separate application priorities (which translates to different QoS levels) at the sensors, and those priorities can change dynamically. A solution to this is proposed in [47], where the authors define a bipartite graph to model the problem of matching sensing tasks (missions) to sensor nodes. A bipartite

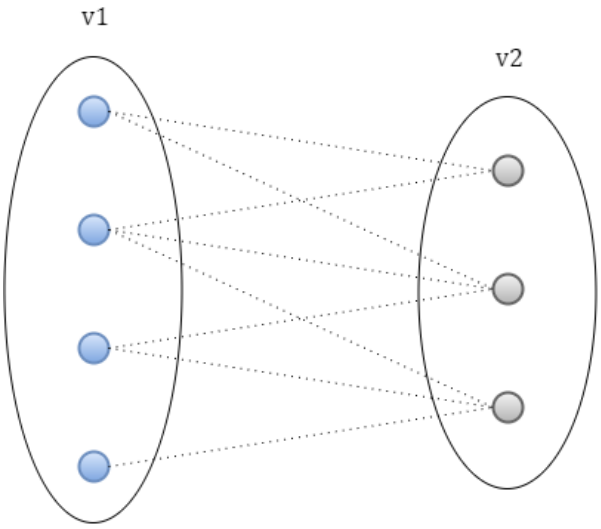

FIGURE 5. Example of a bipartite graph.

graph, as can be seen in Figure 5, is a graph whose vertices can be separated in two disjoint sets, v1 and v2, in such a way that every edge has one end in v1 and one in v2. The limitation here lies in the assumption that each node can only execute a single task, thus eliminating any notion of virtualization.

Mouradian et al. [48] add another perspective, by proposing the merging of WSNs with NFV technology. More specifically, they outline a case study for implementing NFV sensor gateways. Three different network entities are described: the WSN infrastructure provider, the virtual gateway provider, and the application provider. In this architecture the virtualized gateways serve as the intermediary between the substrate physical infrastructure and application developers, providing an abstraction of the sensor deployment details to the latter. In the case study, there are two distinct WSN deployments, comprising two different sets of sensor hardware, to provide the integration and interoperability capabilities of the NFV concept. The two most critical virtualized gateway functions proposed are protocol conversion and information model processing. The first is essential for the protocol translation between the two different infrastructure domains, as well as for the translation between the physical and the application layer. The second one is needed to convert the raw data from the sensors, which can be extremely unstructured and follow multiple different data formats to one unified pattern, such as JSON or XML.

VI. GENERALIZED IoT ARCHITECTURES AND FRAMEWORKS ENABLED BY SDN AND VIRTUALIZATION

The IoT ecosystem consists of much more than only sensor networks. The whole concept is still in its infancy, and standardization efforts are still under way, with multiple competing alliances trying to dominate for a global standard. It is thus natural that there has already been substantial research effort

towards defining suitable architectural reference models for present and future IoT deployments. The ''holy grail'' of these efforts is the emergence of a global viable solution that is both scalable in terms of the enormous number of devices that are expected to be connected in the near future and which will also provide interoperability and seamless connectivity, given the heterogeneity of the participating nodes.

The SENSEI project [49] and the Cognitive Management Framework for the IoT [50] are two examples, but they are too generic and do not capture the singularities of the various IoT use-cases. We believe that the introduction of full-customization and programmability brought by SDN and network virtualization techniques can bring the IoT to maturity.

A. SDN-BASED IoT ARCHITECTURES

An example where SDN is used to facilitate IoT networks can be found in [51], where specific service requirements are translated by a central controller into network requirements, e.g. a minimum data rate, or a maximum tolerable delay or packet loss for each separate flow. Network calculus is used to model the multi-network environment and a genetic algorithm is used to schedule flows, in order to optimize end-to-end flow performance. The paper considers routing optimization, where all flows share the same nodes/resources, but it does not consider network containerization.

In [52], scheduling is seen as the key to make wireless networks deterministic. Their technique is based on time-slotted channel hopping, controlled by a central processing element. This central element makes all decisions, both in terms of route establishment, and in terms of time/frequency slot assignment [53]. The scheme described is interesting, but it can only be applied to networks consisting of well-defined, periodic flows, such as those of an industrial wireless sensor network. The nodes in such networks also have certain limitations, such as absence of packet buffering. It is much more challenging to expand it to a more generalized IoT setting, where heterogeneous flows, with dynamic behavior, are serviced over the same network resources. Also, DetNet is used, instead of OpenFlow to communicate the controllers commands to the network devices (sensors), making it less SDN-standardized.

Another article that underlines the need to address the heterogeneity of the different IoT objects and applications can be found in [54]. The authors conclude that, although with the introduction of IPv6 the vast increase in the number of connected devices is properly addressed, the heterogeneity amongst their different requirements and capabilities still remains an open research question. To address it, they provide a rather high-level architecture of an IoT controller, which to a generic level seems an adequate framework to handle heterogeneous IoT flows. The work misses a more concrete design of the inner workings of the controller and an experimental evaluation of the proposed high-level architecture.

B. VIRTUALIZATION-BASED IoT ARCHITECTURES

Nastic et al. [55] were among the first to propose the combination of network virtualization and the telco cloud for the IoT. Rather than virtualizing the core or the access network that the IoT devices plug into, they propose to virtualize the physical hardware of the various sensors and actuators themselves, creating an IoT cloud infrastructure. A framework that is concentrated on the Northbound APIs of the SDN controller is outlined, to ease the development of IoT applications. An example is given where a fleet management platform is developed, using the provided APIs, to provide functionalities such as vehicle tracking or statistics collection. What this framework lacks is a programmable separation of the various concurrent IoT applications running inside the same cloud, each with its own characteristics and requirements.

The convergence of virtualization and the cloud with IoT is thoroughly investigated in [56]. In that article, a publisher/subscriber based model is proposed, where three entities are recognized: an IoT application provider (e.g., a wireless sensor access network operator, which monitors remote patients), an IoT application user/consumer (e.g., doctors who want access to patient data), and a central entity, which could be the core network that lies in between. Interestingly, the core network is rather seen as a cloud, which offers Sensing-as-a-Service (SaaS) access to the IoT WSN applications. The core network component to manage customer access is a central entity called the Broker, which is responsible for admitting or denying requests to the system, allocating the appropriate resources to them, and monitoring their usage throughout the session. On the other side of the network, special gateways are defined, through which data flow from the sensors to the core network, where they are properly processed to be ready for delivery to the customers. An algorithm is given to perform the matching between the relevant sensor data and the subscribers, called Statistical Group Index Matching.

Although there is mention of accommodating different QoS requirements from the various consumers and/or IoT applications, there are no specifics on how the flows from different WSN gateways can optimally co-exist in the network, in order to satisfy the consumer SLAs. In other words, this framework performs the producer-subscriber matchings and then assumes there are sufficient resources to satisfy all SLAs in the system, which might not be the case. One could imagine a solution where the Broker entity would cooperate with an SDN controller. The controller could isolate the different producer-consumer instances, making them think that they have their own network for the duration of the session, based on the application QoS requirements or the SLA with the customer. This would further enhance the virtualized SaaS nature of this network offering, where each customer gets a portion of the network on demand to connect to the sensor network.

A question that was not clearly answered is how the application gateways from the WSNs to the cloud are going to

be implemented. Kovatsch et al. [57], propose a solution by defining Actinium, a runtime container that allows low-end devices, such as sensors, to expose Web Service-like RESTful APIs that integrate them with the cloud, where their services can be consumed. JavaScript is used to implement those apps, as "heavy" languages (e.g., Java) were deemed inefficient for such devices. The Constrained Application Protocol (CoAP) is employed for the communication to and from the cloud, and consumer applications can access IoT devices using simple scripts. However, this approach centers on the Application Layer of the IoT network stack.

C. COMBINED SDN-VIRTUALIZATION IoT ARCHITECTURES

One of the first works proposing SDN-orchestrated network virtualization can be found in [58]. There, network slicing is suggested for home network management. Multiple service providers can operate over the same physical infrastructure, each getting an isolated slice of the network, directed by its own software controller. This virtual slice can consist of separated bundles of bandwidth, the forwarding table database, or the home router CPU. The different providers can be smart grid operators, HVAC providers, or a video streaming service. By guaranteeing complete isolation to each of those entities, as well as full management control over their respective subnetworks, the providers are free to optimize their service delivery and business model, as well as to share the infrastructure costs, such that the final cost for the homeowner is minimal. In order to practically implement the network, the authors propose a distributed solution, with a central hypervisor module directing multiple OpenFlow controllers, each in charge of a different network slice.

What remains somewhat unclear is which algorithms/policies to use by the controller, so that the resources are shared amongst the various use-cases. The authors propose the use of a "slicing layer" that lies between the resource request from the various applications and the network infrastructure substrate. Rather than providing an exact implementation for this layer, the authors only outlined the slicing mechanism.

Moving towards the same direction, Li et al. [59] argue that the bottleneck in developing vertical, dedicated, application-specific IoT platforms, is the lack of reusability and interoperability. Instead of each application coming along with its own set of sensing hardware, gateways and cloud computing platform, they propose a horizontal SDN-based IoT platform. The architecture is divided into four layers:

- A device layer, which contains the sensing devices and the actuators that collect and transmit the data, or perform a specific action.
- A communication layer that contains the SDN-enabled switches and gateways and which forwards data according to the commands set by the SDN controller.
- A computing layer, which contains the SDN controller(s) and the accounting/billing functions. In addition to populating the forwarding tables of the switches, the controllers are also equipped to perform processing and storage tasks. Those include topology calculation and management, network operation and maintenance procedures, and security management.
- A service layer, which is used by the IoT application developers to give high-level instructions to the controller, which in turn will translate them to specific network commands.

It is important to note that, in this architecture, the sensing devices and the sinks that gather their data are not part of the SDN system, i.e. their behavior is not defined by the SDN controller.

Li et al. [60] defined their own abstract IoT architecture, which consists of a Service layer, providing functionalities such as application event processing or data analytics, a Network layer, where the switches for the data transmissions belong, and a Sensing layer, consisting of IoT end-devices, such as sensors or actuators. This framework is combined with the well-known SDN architecture (Application, Control and Infrastructure layers) to produce a general SDN-IoT framework. This consists of an upper layer with servers providing developers with the necessary APIs for IoT applications, a middle layer, which contains a distributed network OS, commanding several physically distributed SDN controllers, a south layer, which contains the SDN-enabled network switches, and the IoT gateway, which connects them to the middle layer. In essence, this is just the classic SDN architecture, with IoT applications in mind. The authors take it one step further when they claim that, in order to achieve an IoT-optimized network, one has to design the network OS, which sits in the middle layer, using virtualization techniques. The network OS must be used in such a way that the diversity of use-cases and IoT devices is acknowledged. Although the authors do not present any specific details about how virtualization is going to be used in the middle layer, the approach of combining NFV techniques with an SDN-orchestration logic for an IoT network is interesting.

Another article, where SDN-orchestrated network virtualization techniques are proposed, is [61]. There, the routing functionality is moved out of the routers and switches of the core network and is taken over by a central controller. The authors make the case that by giving the controller, which has a global view of the routing state of the entire network, the responsibility to make all routing decisions, signaling overhead is greatly reduced, especially in the case of inter-domain routing. This serves to reduce the total number of physical routing devices required to deploy the network, thus minimizing CAPEX for the provider. Every packet is inspected at the switches and, if necessary, forwarded to the controller, which will make the routing decision. A full protocol that is used for the controller to communicate with the virtualized router module is given and a proof-of-concept prototype is presented and tested on the GEANT testbed. A drawback of this approach is the latency introduced by using a special routing module, separate from the controller. The communication between the two is based on a RESTful

API, which uses a VPN as the channel, and it occurs for every packet that needs a routing decision, thus increasing the round-trip time for a significant portion of the IP packets.

D. APPLICATIONS OF SDN-ENABLED IoT

Current and future urban environments serve as prominent application areas for IoT. The application of IoT is envisioned to optimize various sectors, including transportation, utilities and law enforcement. Thus, significant research has been devoted to developing and implementing the relevant system architecture.

For instance, Liu et al. [63] provide an example of such an SDN-orchestrated architecture. The proliferation of sensors connected to the Internet, as well as of smartphones with extended sensing capabilities [64] brings new opportunities for the acquisition of infrastructure and environmental data from big cities [65]. In order for effective and sustainable urban development to be supported through sensing applications, the authors argue that the current IoT urban deployments should have a shared physical infrastructure, where the same set of sensor nodes can support multiple applications from multiple developers, who will be able to customize the substrate network behavior through software only. To this end, SDN is an ideal candidate, with its Northbound APIs providing the necessary abstractions that developers can exploit to use the substrate sensor network. The proposed software-defined IoT architecture consists of three layers:

- A physical infrastructure layer, which contains the physical hardware, such as sensors, smartphones with sensing capacity, the base stations/access points, and the gateways that connect them to the backbone network. It is important to note that, in the SDN-IoT architecture, those nodes do not possess any intelligence and they leave the decision-making to the control layer that lies above.
- A control layer, which sits between the infrastructure and the application layer. It provides low-level management of the devices in the lowest layer and it exposes the developer APIs to the upper layer. In the urban sensing setting, it provides data aggregation, network transmission, and processing. The SDN controller lies in that plane and is responsible both for the sharing mechanism of the hardware among various applications (having a global view of the geographical sensor topology) as well as for the QoS-aware routing of the data produced in the core network towards the end-consumers.
- An application layer, which is used by developers to build IoT applications, by using the exposed APIs. Full abstraction of the physical infrastructure is provided, so that developers can work without worrying about what happens in the lower layers.

The architectural design is ambitious and plenty of practical implementation details are posed as open challenges for the future. These challenges include the translation of application requirements, both in terms of QoS and geographical sensor location, into sensor node configurations, the sharing of sensors among various competing applications that want access, the optimal and QoS-aware transmission of the data in the core network towards the end-servers, as well as an efficient distribution of data to the processing servers in the cloud [66].

In [62], the authors make the point that the current, fully centralized SDN controller architecture does not address the needs of urban-scale IoT mobile multi-networks. These environments are expected to consist of a multitude of different access networks (e.g., LTE, WiFi or ZigBee), as well as of an enormous number of connected devices, making the current single controller architectures impractical. Thus, a distributed scheme must be used, consisting of multiple physical controllers. Another consideration is mobility, as IoT devices are expected to roam frequently from one access point technology to another. In the outlined architecture, the network is divided into several geographic partitions, each having its own local controller, which has a view of its own partition only. To address scalability issues, each IoT device is assigned to a controller based on a distributed hashing algorithm, and to handle handovers there is a special coordination protocol among the separate controllers. After the initial assignment of devices to network partitions, a special task-resource matching unit, taking into account the service requirements of each flow and the condition of each network partition in terms of load, has the task of re-assigning IoT devices to access points. Although it is one of the few articles considering wireless and mobile networks, besides the access network assignment, there are no further provisions for a flow-scheduling optimization in terms of the backbone network.

A specific example of a demanding IoT application is a Vehicular Ad Hoc Network (VANET) [67]. In a VANET scenario, vehicles can both communicate with each other (V2V communication) in an ad hoc manner, and with a fixed infrastructure that consists of either roadside transceivers or cellular base stations. This can be used to introduce added-value services, such as road safety or traffic management systems. To use SDN in a VANET setting, the role of the SDN switches in the traditional scenario is performed by the vehicles and the roadside units, which receive the control messages from the central unit to perform the routing actions [68]. If the central controller is for some reason unavailable, the nodes must maintain some computational intelligence in order to be able to revert back to traditional ad hoc distributed routing protocols. That can also allow for hybrid schemes, where the controller only defines the general routing directions, by choosing the routing protocol and the relevant parameters, and then the nodes make the forwarding decisions locally. This can be especially beneficial for VANETs, which exhibit extreme fluidity, with the network conditions and the node positioning possibly changing rapidly, making it extremely impractical to maintain and constantly recompute a global network connectivity graph at the controller. That said, if

TABLE 1. Representative summary of important frameworks.

Framework	Technology	Year	Summary
FlowVisor [1]	SDN/Virtualization	2009	A virtualization layer is used to slice resources among different users, using one controller for each flow and OpenFlow for communication
FlowN [2]	SDN/Virtualization	2013	A single, shared OpenFlow controller is used to create isolated subnets for different users through containerization
CloudMAC [14]	SDN/NFV	2012	OpenFlow is used to move 802.11 MAC functions to the cloud
V-Cell [16]	SDN/Virtualization	2014	Abstracts the cellular access network resources by using a central SDN controller for the RAN
MobileFlow [18]	SDN/Virtualization/NFV	2013	A custom-protocol (non-OpenFlow) solution for carrier-grade virtualization of the mobile core network
SoftCell [23]	SDN/NFV	2013	An OpenFlow SDN framework for supporting fine-grained traffic policies in the LTE EPC
SoftAir [25]	SDN/Virtualization	2015	A system-level framework for cellular networks exploiting SDN and a network hypervisor, encompassing both access and core-network
NDNFlow [28]	SDN	2015	OpenFlow is used to facilitate ICN networks through a special NDN-enabled controller module
PolicyCop [30]	SDN	2013	A QoS policy-enforcement platform, where OpenFlow is used for traffic engineering
Sensor OpenFlow [34]	SDN	2012	An OpenFlow-enabled SDN architecture for WSNs, where the functionality of general-purpose sensor nodes can be dynamically re-configured by the controller
SDN-wise [37]	SDN	2015	A stateful SDN solution, which compared to [34] maintains some degree of forwarding computation in the sensor nodes, although SDN is still used to dictate high-level forwarding decisions
SDN-ECCKN [39]	SDN	2016	An algorithm that exploits SDN (non-OpenFlow) to perform the necessary computations for sleep/awake energy-efficient scheduling in the controller, instead of the sensor nodes.
SenShare [41]	Virtualization	2012	Another solution for a multi-purpose sensor network, where compared to [34] traditional distributed WSN protocols are used instead of SDN, in order to achieve network virtualization and subnet isolation
MINA [51]	SDN	2014	An SDN framework proposed to facilitate heterogeneous access-network IoT deployments, by using the controller to translate service requirements into low-level network requirements and schedule traffic flows abiding certain QoS constraints
SDIoT [54]	SDN	2015	A high-level description of an SDN controller designed to handle heterogeneous IoT flows
UbiFlow [62]	SDN/Virtualization	2015	In a similar fashion as in [51], this is an SDN system for IoT multi-network deployments, being novel in defining multiple controllers to isolate heterogeneous subnets and in taking mobility into account

it is feasible for the controller to orchestrate everything, great advantages can be gained, such as QoS provisioning, in case some applications have more strict network demands than others (e.g., safety systems are not delay-tolerant), or channel and transmission power selection for every node, in such a way that the total network interference gets minimized.

VII. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In this survey, we examined the fusion between SDN, network virtualization and IoT. More specifically, this survey is, to the best of our knowledge, the first one to discuss the application of both SDN, network virtualization, and their combination, as key facilitators for the deployment of IoT. It presents an overview of the proposed protocols, architectural models, algorithms and applications for the application of those technologies to IoT. Furthermore, it aims at becoming a reference point for the future efforts of both researchers and industry practitioners who would like to delve into the emerging field of IoT and its numerous envisioned applications. A representative summary of the most prominent architectural frameworks presented in this survey is given in Table 1. In the remainder of this section, we identify some key research challenges and possible future directions in applying SDN and virtualization as orchestrators for IoT.

One possible point of focus would be the optimization of SDN and OpenFlow to accommodate the peculiarities of the IoT paradigm. The expected number of participating devices is going to lead to ever-expanding flow tables at the SDN switches, as well as great overhead for the negotiations between the switches and the controller as devices enter and leave the network continuously, enabling and disabling flows in a rapid manner. It is possible that some form of ''grouping'' for similar devices must be performed to save space in the switch buffers. Also, modifications to OpenFlow are expected, to allow for the forwarding rules of specific IoT applications with certain temporal behavior to be kept in the tables for an extended amount of time.

Another consideration is the extreme heterogeneity of IoT applications and their requirements from the network. For instance, a smart vehicle application, which interconnects cars on a highway to exchange information with one another, would ideally require almost zero latency. A sensor network of an industrial plan would require in addition minimal packet loss, while a mobile video surveillance network could tolerate a modest bit error rate and latency, but it would need a much higher bandwidth than the previous two applications to operate. To satisfy those distinct QoS requirements simultaneously, SDN could offer smart routing and scheduling solutions, and virtualization in the form of

TABLE 2. Acronyms and abbreviations.

AP	Access Point
API	Application Programming Interface
BS	Base Station
CAPEX	Capital Expenditure
CDN	Content Delivery Network
CN	Core Network
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DiffServ	Differentiated Services
EC-CKN	Energy Consumed uniformly Connected K-neighborhood
EPC	Evolved Packet Core
GPRS	General Packet Radio Service
HVAC	Heating, Ventilation and Air-Conditioning
ICN	Information Centric Networking
INPP	In-Network Packet Processing
IntServ	Integrated Services
IoT	Internet of Things
LTE	Long Term Evolution
MAC	Media Access Control
MIMO	Multiple Input, Multiple Output
MME	Mobility Management Entity
MPC	Mobile Packet Core
MPLS	Multiprotocol Label Switching
MU-MIMO	Multi User Multiple Input, Multiple Output
NDN	Named Data Networking
NFV	Network Functions Virtualization
OPEX	Operational Expenditure
PGW	Packet data network Gateway
PRB	Physical Resource Block
QoS	Quality of Service
RAN	Radio Access Network
RRH	Remote Radio Head
SaaS	Sensing as a Service
SDMN	Software-Defined Mobile Network
SDN	Software-Defined Networking
SDR	Software Defined Radio
SGW	Serving Gateway
SLA	Service Level Agreement
SOF	Sensor OpenFlow
TD	Topology Discovery
UE	User Equipment
VANET	Vehicular Ad-Hoc Network
VLAN	Virtual Local Area Network
VM	Virtual Machine
VN	Virtual Network
VSN	Virtual Sensor Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network

network slicing could be used to isolate IoT use-cases with conflicting requirements.

Cellular technology is expected to be a critical tool for IoT. In 5G cellular networks of the future, SDN and NFV are expected to be on the frontline, and it is a challenge to find ways to exploit them to handle the vast increase in data traveling across both the access and the core network. The IoT era will require extended automation of network functions, via virtualization, QoS-aware differentiation of different classes of IoT traffic, and the collection and analysis of data to enable SDN to optimize the network. It is thus imperative for the future 5G architectural models to be designed having in mind the IoT data explosion.

Another advantage that SDN can bring to the numerous sensitive and mission-critical IoT applications concerns security. The challenge lies in the development of algorithms that can detect specific security threats, depending on the IoT application, at the edge of the network in the least complex and overhead-inducing way, by exploring the global view of an SDN controller. Another feature that SDN can provide is the creation of complex device-access rules. As such, the complexity of managing a multi-network, consisting of hundreds of different devices with various permission levels for different actors, can be significantly reduced from an administrator point of view. Also, virtualization can be leveraged to isolate those sensitive applications from the rest of the network to provide assurance and safety. The creation and standardization of such security protocols, interfaces, and applications specifically tailored for IoT devices, is still an open research area.

Finally, SDN, which provides centralized network management and data collection, can be used together with machine learning techniques, so that the network can become more intelligent and self-adaptive. Real-time decisions can be made depending on the type and characteristics of the traffic received. The development of such customized machine learning algorithms to serve for IoT traffic flows, both to optimize performance and security is expected to be an emerging research field, merging the worlds of networking and artificial intelligence.

APPENDIX

In Table 2 we list the acronyms and abbreviations used throughout this survey.

REFERENCES

[1] R. Sherwood *et al.*, "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep. OPENFLOW-TR-2009-1, 2009, pp. 1–13.
[2] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 20–27, Mar./Apr. 2013.
[3] Z. Qin, L. Iannario, C. Giannelli, P. Bellavista, G. Denker, and N. Venkatasubramanian, "MINA: A reflective middleware for managing dynamic multinetwork environments," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–4.
[4] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, Mar. 2014.
[5] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, 1st Quart., 2014.
[6] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *Queue*, vol. 11, no. 12, p. 20, 2013.
[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.
[8] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 553–576, 1st Quart., 2015.
[9] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Athanasios, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Netw. Appl.*, vol. 20, no. 1, pp. 4–18, Feb. 2014.
[10] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.
[11] A. L. V. Caraguay, A. B. Peral, L. I. B. López, and L. J. G. Villalba, "SDN: Evolution and opportunities in the development IoT applications," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 5, p. 735142, 2014.

[12] E. Hossain and M. Hasan, "5G cellular: Key enabling technologies and research challenges," *IEEE Instrum. Meas. Mag.*, vol. 18, no. 3, pp. 11–21, Jun. 2015.

[13] F. Granelli *et al.*, "Software defined and virtualized wireless access in future wireless networks: Scenarios and standards," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 26–34, Jun. 2015.

[14] P. Dely, J. Vestin, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "CloudMAC—An OpenFlow based architecture for 802.11 MAC layer processing in the cloud," in *Proc. IEEE Globecom Workshops*, Dec. 2012, pp. 186–191.

[15] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "A novel LTE wireless virtualization framework," in *Proc. Int. Conf. Mobile Netw. Manage.*, 2010, pp. 245–257.

[16] R. Riggio, K. Gomez, L. Goratti, R. Fedrizzi, and T. Rasheed, "V-cell: Going beyond the cell abstraction in 5G mobile networks," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–5.

[17] A. A. Gebremariam, L. Goratti, R. Riggio, D. Siracusa, T. Rasheed, and F. Granelli, "A framework for interference control in software-defined mobile radio networks," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2015, pp. 892–897.

[18] K. Pentikousis, Y. Wang, and W. Hu, "MobileFlow: Toward software-defined mobile networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 44–53, Jul. 2013.

[19] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, "A virtual SDN-enabled LTE EPC architecture: A case study for S-/P-gateways functions," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Nov. 2013, pp. 1–7.

[20] M. R. Sama, L. M. Contreras, J. Kaippallimalil, I. Akiyoshi, H. Qian, and H. Ni, "Software-defined control of the virtualized mobile packet core," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 107–115, Feb. 2015.

[21] N. L. M. van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, 2014, pp. 61–66.

[22] N. L. M. van Adrichem, F. Iqbal, and F. A. Kuipers. (2016). "Computing backup forwarding rules in software-defined networks." [Online]. Available: http://arxiv.org/abs/1605.09350

[23] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *Proc. 9th ACM Conf. Emerg. Netw. Experim. Technol.*, 2013, pp. 163–174.

[24] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 80–91, Jun. 2014.

[25] I. F. Akyildiz, P. Wang, and S.-C. Lin, "Softair: A software defined networking architecture for 5g wireless systems," *Comput. Netw.*, vol. 85, pp. 1–18, Jul. 2015.

[26] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[27] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Netw.*, vol. 29, no. 3, pp. 68–74, May/Jun. 2015.

[28] N. L. M. van Adrichem and F. A. Kuipers, "NDNFlow: Software-defined named data networking," in *Proc. 1st IEEE Conf. Netw. Softw. (NetSoft)*, Apr. 2015, pp. 1–5.

[29] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, "Principles of elastic processes," *IEEE Internet Comput.*, vol. 15, no. 5, pp. 66–71, Sep. 2011.

[30] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: An autonomic QoS policy enforcement framework for software defined networks," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Nov. 2013, pp. 1–7.

[31] C. Kruger, A. Abu-Mahfouz, and G. Hancke, "Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2015, pp. 1926–1931.

[32] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IoT gateway: Bridging-wireless sensor networks into internet of things," in *Proc. IEEE/IFIP 8th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Dec. 2010, pp. 347–352.

[33] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Proc. 27th Biennial Symp. Commun. (QBSC)*, 2014, pp. 71–75.

[34] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[35] M. Jacobsson and C. Orfanidis, "Using software-defined networking principles for wireless sensor networks," in *Proc. 11th Swedish Nat. Comput. Netw. Workshop (SNCNW)*, Karlstad, Sweden, May 2015.

[36] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, "pTunes: Runtime parameter adaptation for low-power MAC protocols," in *Proc. 11th Int. Conf. Inf. Process. Sensor Netw.*, 2012, pp. 173–184.

[37] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 513–521.

[38] P. Dely, A. Kassler, and N. Bayer, "OpenFlow for wireless mesh networks," in *Proc. 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2011, pp. 1–6.

[39] Y. Wang, H. Chen, X. Wu, and L. Shu, "An energy-efficient SDN based sleep scheduling algorithm for wsns," *J. Netw. Comput. Appl.*, vol. 59, pp. 39–45, Jan. 2016.

[40] Z. Yuan, L. Wang, L. Shu, T. Hara, and Z. Qin, "A balanced energy consumption sleep scheduling algorithm in wireless sensor networks," in *Proc. 7th Int. Wireless Commun. Mobile Comput. Conf.*, 2011, pp. 831–835.

[41] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "SenShare: Transforming sensor networks into multi-application sensing infrastructures," in *Proc. Eur. Conf. Wireless Sensor Netw.*, 2012, pp. 65–81.

[42] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," *ACM SIGPLAN Notices*, vol. 37, no. 10, pp. 85–95, 2002.

[43] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst.*, 2009, pp. 1–14.

[44] A. P. Jayasumana, Q. Han, and T. H. Illangasekare, "Virtual sensor networks—A resource efficient approach for concurrent applications," in *Proc. 4th Int. Conf. Inf. Technol. (ITNG)*, 2007, pp. 111–115.

[45] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things," *Trans. Emerg. Telecommun. Technol.*, vol. 25, no. 1, pp. 81–93, 2014.

[46] S. Tilak, K. Chiu, N. B. Abu-Ghazaleh, and T. Fountain, "Dynamic resource discovery for sensor networks," in *Proc. Int. Conf. Embedded Ubiquitous Comput.*, 2005, pp. 785–796.

[47] H. Rowaihy, M. P. Johnson, O. Liu, A. Bar-Noy, T. Brown, and T. L. Porta, "Sensor-mission assignment in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 6, no. 4, p. 36, 2010.

[48] C. Mouradian, T. Saha, J. Sahoo, R. Glitho, M. Morrow, and P. Polakos, "NFV based gateways for virtualized wireless sensor networks: A case study," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 1883–1888.

[49] V. Tsiatsis *et al.*, "The SENSEI real world Internet architecture," in *Proc. Future Internet Assembly*, 2010, pp. 247–256.

[50] P. Vlacheas *et al.*, "Enabling smart cities through a cognitive management framework for the Internet of Things," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 102–111, Jun. 2013.

[51] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.

[52] P. Thubert, M. R. Palattella, and T. Engel, "6TiSCH centralized scheduling: When SDN meet IoT," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2015, pp. 42–47.

[53] M. R. Palattella, P. Thubert, X. Vilajosana, T. Watteyne, Q. Wang, and T. Engel, "6TiSCH wireless industrial networks: Determinism meets IPv6," in *Internet of Things* (Smart Sensors, Measurement and Instrumentation,). Switzerland: Springer, Jan. 2014, pp. 111–141.

[54] Y. Jararweh, A. Mahmoud, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDIoT: A software defined based Internet of things framework," *J. Ambient Intell. Humanized Comput.*, vol. 6, no. 4, pp. 453–461, Aug. 2015.

[55] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning software-defined IoT cloud systems," in *Proc. Int. Conf. Future Internet Things Cloud (FiCloud)*, 2014, pp. 288–295.

[56] M. M. Hassan, B. Song, and E.-N. Huh, "A framework of sensor-cloud integration opportunities and challenges," in *Proc. 3rd Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2009, pp. 618–626.

[57] M. Kovatsch, M. Lanter, and S. Duquennoy, "Actinium: A restful runtime container for scriptable internet of things applications," in *Proc. 3rd Int. Conf. Internet Things (IOT)*, 2012, pp. 135–142.

[58] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *Proc. 2nd ACM SIGCOMM workshop Home Netw.*, 2011, pp. 1–6.
[59] Y. Li, X. Su, J. Riekki, T. Kanter, and R. Rahmani, "A SDN-based architecture for horizontal Internet of Things services," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.
[60] J. Li, E. Altman, and C. Touati, "A general SDN-based IoT framework with NVF implementation," *ZTE Commun.*, vol. 13, no. 3, pp. 42–45, 2015.
[61] J. Batalle, J. F. Riera, E. Escalona, and J. A. Garcia-Espin, "On the implementation of NFV over an OpenFlow infrastructure: Routing function virtualization," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, 2013, pp. 1–6.
[62] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, "UbiFlow: Mobility management in urban-scale software defined IoT," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 208–216.
[63] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, "Software-defined Internet of Things for smart urban sensing," *IEEE Commun. Mag.*, vol. 53, no. 8, pp. 55–63, Sep. 2015.
[64] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 29–35, Aug. 2014.
[65] G. P. Hancke, B. de Carvalho e Silva, and G. P. Hancke, "The role of advanced sensing in smart cities," *Sensors*, vol. 13, no. 1, pp. 393–425, 2012.
[66] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, 2013.
[67] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," in *Proc. 13th Annu. Medit. Ad Hoc Netw. Workshop (MED-HOC-NET)*, 2014, pp. 103–110.
[68] M. Zhu, J. Cao, D. Pang, Z. He, and M. Xu, "SDN-based routing for efficient message propagation in VANET," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.*, 2015, pp. 788–797.

NIKOS BIZANIS received the B.S. degree in computer science and telecommunications from the University of Athens, Greece. He is currently pursuing the M.Sc. degree with the Network Architectures and Services Group, Delft University of Technology, The Netherlands. His thesis was on the application of software-defined networking and network virtualization to the Internet-of-Things.

FERNANDO A. KUIPERS (SM'10) received the Ph.D. degree (Hons.) from the Delft University of Technology (TUDelft), Delft, The Netherlands, in 2004. He was a Visiting Scholar with Technion and Columbia University in 2009 and 2016, respectively. He is currently an Associate Professor with TUDelft, where he is involved in Internet science. His research focuses on network optimization, network resilience, quality of service, and quality of experience, and addresses problems in software-defined networking, optical networking, content distribution, and cyber-physical systems/infrastructures. He is a member of the Executive Committee of the IEEE Benelux Chapter on Communications and Vehicular Technology. His work on these subjects include distinguished papers at the IEEE Infocom 2003, Chinacom 2006, IFIP Networking 2008, the IEEE Future Multimedia Networking 2008, the IEEE International Symposium on Multimedia 2008, International Test Conference 2009, IEEE Intelligence and Security Informatics Conference 2014, and NetGames 2015.