IET Software

Special Issue: Software Architecture for the Web of Things (SAWoT)

IET Journals
The Institution of Engineering and Technology

Architecting the Web of Things for the fog computing era

ISSN 1751-8806
Received on 5th December 2017
Revised 7th March 2018
Accepted on 29th March 2018
E-First on 10th May 2018
doi: 10.1049/iet-sen.2017.0350
www.ietdl.org

Niko Mäkitalo[1] ✉, Francesco Nocera[2], Marina Mongiello[2], Stefano Bistarelli[3]

[1]*Department of Computer Science, University of Helsinki, Gustaf Hällströmin Katu 2b, 00014 Helsinki, Finland*
[2]*Department of Electrical & Information Engineering (DEI), Polytechnic University of Bari, Via E. Orabona 4, 70125 Bari, Italy*
[3]*Department of Mathematics and Computer Science, University of Perugia, Via Vanvitelli 1, 06123 Perugia, Italy*
✉ *E-mail: niko.makitalo@helsinki.fi*

Abstract: Fog computing paradigm is emerging after a decade's dominance of cloud-based system design and architecture. Now, instead of centralising the computation and coordination to remote services, these are deployed and distributed to all over physical surroundings and network nodes, including cloud services, smart gateways, and network edge devices. At the moment, the majority of the Internet of things (IoT) systems and software has built on top of open Web-based technologies. The authors assume that with the ever-growing number and heterogeneity of connected devices, it becomes ever-more crucial to have open standards that support interoperability and enable interactions. They review the current technological space for architecting Web technology-based IoT software in the coming era of fog computing. They focus on fundamental research challenges and discuss the emerging issues.

1 Introduction

The software has become an essential part of every aspect of the society and our daily life. This is indicated by many recent trends in our society – smart homes, smart traffic, smart cities, (autonomous) robots, autonomous connected vehicles and so on – all contain an ever-increasing amount of software. These trends also indicate that the single device computing era is coming to an end. However, the development and software architectures have not changed much during past ten years: At the moment, cloud-based services are the de facto way how the software is constructed, and then provided for the end users in the form of Web and native apps. This approach, however, cannot accommodate all the needs that come with the multi-device era where programmable objects are everywhere and require efficient and real-time coordination and distributed computations. This has led to a situation where computation and coordination require new types of software architectures and programming models.

Internet of things (IoT) has been one key research topics in computer science for some time. IoT has no single definition, and hence it is often referred as an approach for connecting all the physical things to the Internet, or, as an extension to mobile computing. Thus, IoT can be seen to have multiple research subfields, like the Internet of Industrial Things, the Internet of Vehicles, or the Internet of People. Web of Things (WoT) is one of these subfields and a general term used for describing all the approaches to connecting physical things to the World Wide Web [1–4]. In the coming years, people use more and more various types of Web-enabled client devices, and data is stored simultaneously on numerous devices and cloud-based services. Hence, it is the devices together with people and services which form the modern computing environment. The expectations towards interoperability will dramatically raise, which will imply significant changes for software architecture as well since the development is evolving from traditional client–server architectures to decentralised architectures. Thus, cloud computing is complemented with two new computing paradigms: *edge computing* (computation solely on the device-end) and *fog computing* (computation everywhere on the network level).

Fog computing is an emerging paradigm that was presented by Cisco in 2012. It promises to be an evolution of cloud-based systems and is primarily targeted for the IoT [5]. While edge computing is solely about computations on the network edge devices, the goal of fog computing is to enable exploiting computation and data resources across cloud services, edge devices, as well as intelligent network nodes. If today the *cloud* is the most used abstraction and environment to handle remote applications, the *fog* then offers the advantage of better supporting new computer applications in our connected world. For example, autonomous driving cars, remote monitoring systems for patients, drones for home delivery, the adaptive lighting of streets and homes can all benefit from fog computing. All this by leveraging the pervasive computing infrastructure that consists of *ad hoc* processors, smart routers, and personal devices such as smartphones for computations. This approach allows reducing bandwidth consumption in IoT environments, exploiting a distributed structure that is quite similar to that used in peer-to-peer communications. In some cases, the fog can also be seen as a parallel network to the public since it can allow access to resources and computing power without passing through a public Internet connection [6].

Open standards and Web technologies provide tools and a platform for implementing applications in more vendor-neutral ways – in contrast to native apps that can run only on one platform. However, even though the communication is built-in to the Web, the interactions still happen in the same way as with native apps. Also, the Web browser essentially is an app itself and only offers a sandbox for interacting with other entities. Thus, pure Web technology-based software partly suffers from the same, and even more limitations than native software do suffer. Despite these limitations, however, Web technologies can still have advantages over native apps [7], and be used for enabling co-operation and interactions between the devices and users. Moreover, Web technologies can teach a lot about standardisation for enabling vendor-neutral interactions for the required architectures. For this reason, the objective of this article is to study the future research avenues for architecting WoT software, and especially on the fog computing era. The open standards and Web technology-based standards will then play a key role in future to enable free communication, interoperability, and interactions between the entities in the fog.

This article is structured as follows: In Section 2, we describe the key enabling technologies (KET) for building WoT software at the moment and in the near future. In Section 3, we outline some

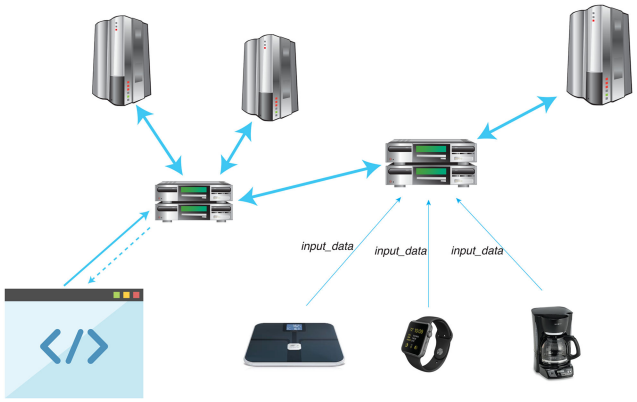

Fig. 1 *Generic cloud-based WoT system architecture*

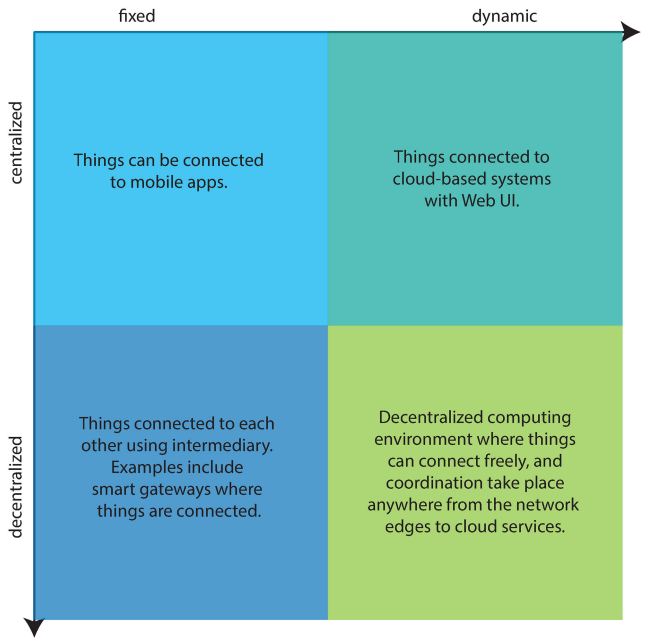

Fig. 2 *Web technologies support leveraging the full potential of the fog*

motivation and research questions for the WoT research. Section 4 introduces some research themes that are focused on WoT system and software research. In Section 5, we discuss how the current technologies and approaches can respond to the research questions. We also discuss about the possible threats to the validity of our research. Finally, Section 6 draws some conclusions.

2 Design space: towards fog computing

WoT system architecture at the moment typically builds on a cloud-based, centralised approach where the physical objects are connected to an Internet service, sending data there and possibly getting some actuation instructions back. (We have illustrated this kind of abstract-level example of traditional WoT architecture in Fig. 1.) It has been discussed, however, that the network is becoming the bottleneck in cloud computing [8], and relying on such solutions may not be fast enough for the increasing number of mission-critical applications that employ the physical objects [6].

Due to the fast development of ICT technology smaller and more powerful chips have become cheap and can now be embedded to everywhere. In addition to the everyday physical objects to become programmable, this also drives the network to become more and more programmable with the smarter routers and eventually with 5G technologies. The following describes how some key technologies foster the transfer from moving from centralised cloud-based WoT solutions towards WoT that operate everywhere on the network. In Fig. 2, we present an illustration of an abstract WoT architecture for the Fog where the computation and coordination are distributed and can take place dynamically on various nodes.

2.1 Dynamic and decentralised computation and coordination infrastructures

While the traditional WoT applications appear to run in the browser, the actual computation and coordination have typically taken place on a centralised service. The top-right corner in Fig. 3 represents these traditional approaches. One key idea of fog computing is to exploit the benefits the modern, dynamic computing environment offers. With non-Web-based approaches, this goal becomes challenging, however. Fortunately, Web technologies foster for the leveraging of the edge, fog, and cloud, since these technologies typically work everywhere, and hence support harnessing the heterogeneous computational resources. In other words, it is possible (or it is becoming possible) to partition and modularise the software since the Web-based (typically JavaScript) components can be moved and executed on the servers, edge devices as well as on many intelligent computation nodes in between the edge and cloud. This further fosters the emergence of the fog computing, as depicted in bottom-right in Fig. 3.

In practice, however, the distinction between fog and edge is not always clear since the mobile devices today offer excellent support for personal area networks, namely with Bluetooth. With these, and other network gateways becoming increasingly smart and programmable, this allows bringing the intelligence to the network level. The processing of the data can be taken care by a smart gateway or by a mobile device has many benefits over the cloud-based computation and coordination. For instance, coordination at the network edges help to reduce the communication lag and allows device coordination in situations when there is no Internet connection, or the quality of the connection is terrible. The more local coordination can also support functional safety since if one device fails to perform some operation, other devices are there to replace it.

2.1.1 Container technology: One of the challenges while moving from monolith architecture towards more distributed and decentralised microservice architecture is the management and deployment of the software constructs. This challenge is typically responded with *container technology*, with its (almost de facto) implementation Docker. Its primary objective is to make the microservices easily portable and configurable, and enable them running in isolation from the host computer.

Docker technology is not tied to any other specific programming language or implementation technology, which from the WoT perspective becomes very useful. In contrast, in a monolith architecture, it is often challenging mix other services and libraries implemented with other technologies and programming languages than the remainder of the system. The container-based services communicate with messages, and each service is independent implementation. This allows freely mixing various technologies and improves the interoperability of the different systems.

2.1.2 Serverless computing: Another alternative for the traditional server-based software is serverless computing, which the central idea is to free the software developer from maintaining a server. Instead, the developer implements functions that are deployed and executed on a cloud service, and the costs of using this service are based on pay-per-execution. Presently, the most used and well-known examples of the serverless computing include AWS Lambda, Google Cloud Functions, and Azure Functions, but others are continually emerging. As the number of serverless computing service providers increases, it may become challenging to decide which provider to use. Fortunately, the open source Serverless Framework (https://serverless.com) helps in the task by providing a homogenised interface to take benefit of these services.

From WoT development perspective, the serverless computing approaches give a solution for decentralising the computation easily. Think for example if a physical object has the insufficient computing power, it could then invoke a method remotely.

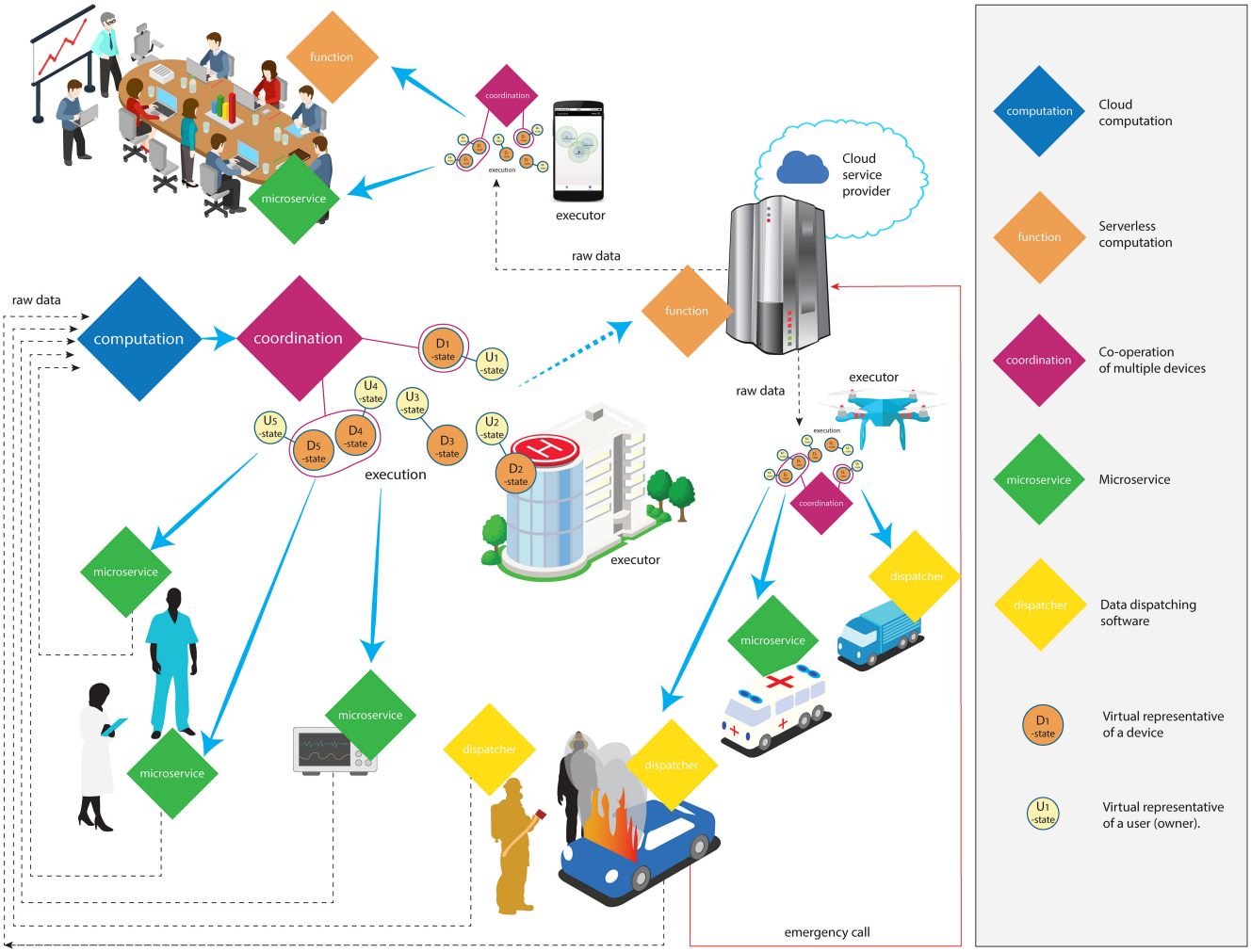

Fig. 3 *Illustration of open Web technology-based computation and coordination in the fog*

Similarly, some computations can be offloaded from the browser and performed remotely.

2.1.3 From cloud services to the edge devices: From WoT perspective, the development is often fostered by the emerging APIs of modern browsers since these enable more sophisticated architectures. The browser APIs may, for example, allow connecting the devices directly to the browser. Such connectivity further supports tasks required for coordinating the physical objects and performing computations close to the data sources [9].

From WoT point of view, it is also fortunate that now approaches allowing dynamic computations have emerged. For instance, Apple provides JavaScriptCore framework for iOS, macOS, tvOS (https://developer.apple.com/documentation/javascriptcore) or Android LiquidCore framework (https://github.com/LiquidPlayer/LiquidCore) allow the computation and coordination to be performed by some of the edge devices [10].

Moreover, although most of the serverless computing approaches (discussed above) are yet targeted to run in the cloud, some evidence exists that the cloud providers have realised the potential of the edge devices, and now offer their solutions for fog computing: Amazon's Greengrass software (https://aws.amazon.com/greengrass/) allows running AWS Lambda functions on the users' devices. It appears that the computation and coordination are again coming closer to network edges and prevents dispatching loads of useless sensor data to the cloud which eats up bandwidth and can be costly.

2.2 Connectivity and communication

Connectivity technologies can roughly be categorised into two groups: technologies that require infrastructure's support for communication, and technologies which do not require a separate intermediary technology. These both have pros and cons, as we discuss next.

2.2.1 Infrastructure-based communication: By communication requiring intermediary technology for enabling relaying the messages between the communicating entities, we refer to infrastructure-based communication. The concrete infrastructure typically is either wireless or wired local area network (W/LAN), or wide area network (WAN), which is more or less a synonym for the Internet. Currently, the Internet-based communication is yet the most typical way to implement the communication for a WoT system, and the protocol used for the task is often hypertext transfer protocol (HTTP). In many cases, the WoT system architecture follows representational state transfer (REST) design principles [11] (although this indeed is not tied to the used communication). Other much-used protocols at the moment are message queuing telemetry transport (MQTT) and constrained application protocol (CoAP) which both enable lightweight communication for more constrained devices and more direct messaging between the entities. Especially MQTT (in addition to REST) at the moment has gained particular popularity for the Internet of Things systems. However, using MQTT or CoAP requires a broker for relaying the messages. Moreover, although HTTP protocol may not be the best fit anymore, in the recent 2.0 version (HTTP/2) offers support for two-way communication [12]. The downside yet is that one entity acts as the server and the others as clients, which initialise the connection.

2.2.2 Device-to-device communication: In the fog computing, there is a growing demand for more direct communication technologies with support for bi-directional communication between the entities. The distinction between edge and fog

computing is often not very clear [13], since the modern mobile devices offer excellent connectivity, and thus act as gateways for many peripheral devices. At the moment, for example, personal and body area networks (BANs) are formed mainly with Bluetooth and Bluetooth low-energy (Smart) technologies between the things and the mobile phone. In general, smart gateway technology uses ZigBee or Z-Wave to communicate with the smart home electronics, and these gateways are then typically connected LAN and sometimes to WAN for external access. Another device-to-device connectivity technology is WiFi direct, which allows using the existing WiFi technology to form connections between the supported devices. In practice, WiFi direct means that one of the devices says a mobile phone, forms a group and acts as its leader, and other devices then connect to this established group.

3 Motivation

The objective of this article is to study the future research avenues for architecting WoT software, and especially on the fog computing era. Our primary research question is:

What are the present and future WoT research themes?

We set the scope of our study and answer the primary research question with four supportive and more focused research questions. While these research questions are targeted to WoT research, they are at least partially intertwined with the IoT research. Thus, while discussing them, we also refer the two interchangeably. The primary research question is:

- *RQ1*: How to use data and hardware resources for perception and interaction?
- *RQ2*: What are the current building blocks for WoT, and who is providing them?
- *RQ3*: Is interoperability with other systems supported, and how can this aspect be improved?
- *RQ4*: What are the current security and privacy issues, and can these threats be covered?

In the remainder of this article, we use these research questions for discussing the current state of the WoT research, as well as outlining some future research ideas. To answer the primary research question, we start by outlining some of the current research themes in the next section. Then, we continue towards a comprehensive discussion on the other research questions (RQ1–RQ4).

4 Research themes for WoT

WoT has been studied extensively since 2011 [3, 4]. Since WoT is an umbrella term for multi-device IoT systems that implemented with Web technologies, many research areas are closely related. Here we outline some of the areas that we find to be the most important ones in future research.

4.1 Liquid user experience

Liquid software refers to the approaches in which applications and data can seamlessly flow from one device to another, allowing the users to roam freely across all the computing devices that they have [14]. The goal is that users of liquid software do not need to worry about data copying, manual synchronisation of device settings, application installation, or other burdensome device management tasks. Instead, things should work with minimal effort. From the software development perspective, liquid software should dynamically adapt to the almost infinite set of devices that are available to run it.

From the usability of WoT perspective, liquid software research studies important paradigm: How the applications can roam from one device to another, following the user from everywhere. The studies aspects include for example state management and synchronisation, and various types of user interfaces [14]. While this paradigm is barely about beginning to work with full-fledged computing devices there is much research to be carried out in the context of constrained devices. (The most advanced is Apple's Continuity (http://www.apple.com/macos/continuity/).

4.2 Complex event processing

Complex event processing (CEP) refers to real-time analysis and filtering of large amounts of data as explained by Boubeta-Puig *et al.* [15]. The aim is at detecting meaningful events to be used either directly by the end users, or more commonly, by other systems. According to Boubeta-Puig *et al.*, so-called event patterns are used for this task, which are conditions to be met for detecting the interesting situations. The data streams leveraged for the processing can, for example, come from logistics, communication networks, social networks, health and wellness services and so on. Many example CEP systems can be found in [16]. Many issues yet must be solved to make CEP applicable to commercial systems. These issues include for example fault tolerance and scalability aspects [17, 18]. Also, query languages should be studied since these could help utilising the processed events [19]. Despite of these, many new data management and optimisation techniques for processing data has emerged [20–23].

In the context of WoT and fog computing, complex events and processing them becomes increasingly important as this has genuinely potential to augment people in their lives. Using Web technologies bring both, opportunities and challenges. The opportunities include that the data can be processed – or pre-processed – closer its origin, say on an intelligent network node for instance. However, such support is limited, which brings us to the challenges. While JavaScript adoption has been vast, there yet is plenty of research to be carried on this area to process the data anywhere in the Fog efficiently: Presently, JavaScript's primary function is to enable data visualisations instead of the actual processing (e.g. with the popular D3.js library).

4.3 Microservice architectural style

The microservices architecture emphasises dividing the system into small and lightweight services. Whether considering microservices as a new architectural style; as an implementation of service-oriented architecture (SOA) [24, 25]; or as an evolution of the traditional SOA style [26], the approach is one of the most recent avenues towards more flexible installations and executions. The motivation for this transition comes from the fact that continually maintaining a complex monolithic architecture has resulted in difficulties in keeping up in pace with new development approaches such as DevOps, calling for deployment several times a day.

In [27], microservice architecture is defined as a distributed application where all modules are microservices. The microservices that can be implemented, tested and executed individually, help to manage the development. Some other benefits include increased agility and developer productivity, and also scalability, reliability, and maintainability of the whole system. However, the benefits come with challenges. For instance, discovering services over the network is often challenging and may introduce new weak point to the system. Also, security and privacy management often come more challenging when the architecture is highly decentralised. Other challenges include optimising communication and performance. So finally, benchmarking and testing the system as a whole may become challenging as well. Despite the many challenges, the system can often benefit from the most advantages [28, 29].

From WoT research perspective, this transition at the moment affects to the backend of the system. However, as has been discussed, the physical objects in our surroundings are becoming more and more capable of performing computations. Thus, some of the physical objects may then run some microservices. The benefit is that deployments can be automated and made continuous. From the perspective of fog computing, the benefit of microservices is that the data can stay close to its source, and as a result, the lag in the communication reduces. Moreover, the privacy of the users may improve if all the data does not need to be transferred over to a service provided by an Internet company that benefits from the user-produced data.

5 Discussion

In this section, we continue the discussion on the remaining four research questions (RQ1–RQ4). In the end of this section, we discuss about threats to the validity of our study.

5.1 RQ1: How to use data and hardware resources for perception and interaction?

Physical devices (ever-more often constrained and embedded devices) are producing vast amounts of data with their sensing capabilities. Presently, this data is typically dispatched further without processing its raw form. In future, this data can be expected to act as one of the critical enablers for many WoT scenarios, for instance helping people as well as other devices to make more well-informed decisions [30]. However, the raw data itself does not support making these decisions, and there are mainly four reasons for this [31]: *First*, the data is distributed to various locations and must be acquired someway. *Second*, the amount of data is very large (much steeper any human could ever interpret). *Third*, the data is often noisy and contains false values. *Fourth*, the data in its raw form (e.g. temperature values) is not very useful, but instead must be processed to more abstract-level information (e.g. weather conditions). Hence, to leverage this data we need new tools for collecting, processing, analysing, describing, and then, finally, for using it in the decision-making processes [30].

The hardware resources can be used for interacting and giving 'the output' to the physical world. In essence, this means sending some actuation instructions over the network. While now the network is typically the Internet, in fog computing context heterogeneous networks play the ever-more important role, giving specific benefits like reduced dependency on high-quality connection, shorter communication lag, and some fault tolerance. Like utilising the data, also the actuation requires abstractions: For example, commanding each servo motor with an appropriate amount of power would not work. In the context of WoT, the most famous approaches come from Guinard and Trifa in their many publications and books [32].

In essence, there are two kinds of programming models for the fog computing: *sense-process-actuate* models, and *stream processing* models [6]. Of these two, the stream processing model has been the typical one and has previously been used in many IoT and cloud computing approaches. In the fog computing context, the stream processing is still a popular model as it is more straightforward to implement. The complex event processing (CEP) approaches often belong to this category, although the processing typically takes places on the edge devices [20–23]. The sense-process-actuate model, on the other hand, often requires more high-level abstractions when used with heterogeneous devices. Until now there only have been few such models in the context of fog computing [13], and especially in the context WoT and fog computing. For this reason, many have simply used APIs for the task, which makes the programming less effective and affects to the maintainability aspect as well.

We believe that higher abstraction models will start to emerge sooner than later, which is indicated by the recent developments in fog computing. Mahmud *et al.* for example discuss challenges regarding structural, service and security-related issues [33]. Also, a taxonomy of fog computing classifies and analyses the existing works based on their approaches towards addressing the challenges, proposing some promising research directions to pursue in the future. Also, Wen *et al.* provide an overview of the core issues, challenges and future research directions in fog-enabled orchestration for IoT services in [34]. Additionally, they also present experiences of an orchestration scenario as a workflow across all layers of fog architecture. The reported experiences are based on their own fog orchestrator.

The work by Byers [35] discusses some of the more significant architectural requirements for the critical IoT networks in the context of exemplary use cases, and how fog computing techniques can help fulfil them. A wide variety of potential IoT use cases, serving many critical vertical market switch several selected use cases in each market that have requirements to which fog techniques are potentially applicable, such as agriculture, health, transportation, smart cities/smart buildings and so on. The work provides useful guidelines to make more informed decisions in the architecture, partitioning, design, and deployment of fog computing in IoT networks.

In [36], Rahmani *et al.* present a fog-assisted system architecture capable of coping with many challenges in ubiquitous healthcare systems such as mobility, energy efficiency, scalability, and reliability issues. According to the Rahmani *et al.*, successful implementation of smart e-health gateways can enable massive deployment of ubiquitous health monitoring systems, especially in clinical environments. The work presents a prototype of a smart e-health gateway called UT-GATE and implements an IoT-based early warning score health monitoring to practically show the efficiency and relevance of the system on addressing a medical case study.

An example from some of the authors of this paper for building concrete fog computing applications was released lately. In [37], we present action-oriented programming model for developing and coordinating interactions between entities operating in the Fog. In addition, in the paper, we highlight the pain points of the traditional mobile app and cloud computing development in contrast to fog computing.

5.2 RQ2: what are the current building blocks for WoT, and who is providing them?

At the moment, the world is full of closed and open APIs [9], and these are most basic building blocks of the WoT development as well. Nearly, all the new IoT devices have an API for communicating and accessing it via cloud service. In some case also the gateways offer an API that can be used directly communicating with the IoT devices when operating inside the same LAN. These APIs typically follow at least some of the REST design principles [11]. Other building blocks are various protocols (e.g. HTTP and WebSocket) that then enable the actual communication, as was described above. Moreover, in addition to the things offering their APIs via cloud services, some things also embed Web servers in them [38, 39] and then provide APIs. However, this approach has been changing after introducing new two-way communication protocols which are typically more light-weight and work even when the topology of the network changes or the device loses its IP address. For the user interface, the current responsive HTML/CSS frameworks (Bootstrap and Foundation) together with the front-end JavaScript frameworks (e.g. React.js, Angular.js, Vue.js) are great building blocks. For other more constrained devices, however, the building block for user interfaces are yet very limited, and for this reason, we need new ways for implementing user interfaces.

From fog computing perspective, Docker technology is one of the most critical building blocks. It allows building microservices that then can be deployed and maintained in various locations, often in automated ways. Other technologies that help to leverage the fog computing's potential are the current serverless computing frameworks (e.g. AWS Lambda, Google Cloud Functions etc.) that can be used in a homogenised way with the open source serverless framework.

Despite these current building blocks, the development is at the moment is yet forced to focus a lot on the connection and communication-related aspects, rather than the actual application logic. The actual development tools (e.g. Node-RED, Meshblu etc.) poorly fit the fog computing or are such research-oriented that their long-term support cannot be expected. Fortunately, while looking at the numbers of published papers, the IoT and WoT development are timely topics in the research community, and new approaches can be expected to emerge.

At some point, it may even become possible to leverage the existing social structures for sharing building blocks and open APIs, as some researchers have been proposing [3]. Also, posting one's creations to social networks (e.g. Facebook, Linkedin, Twitter etc.) which share them automatically with trusted or otherwise relevant people sounds promising idea. This would also mean that we would not need to create the social networks for the sharing from scratch. Sharing could also enable advertising one's skills

(e.g. a student seeking first job). In social media, however, some things go viral in these networks, meaning they get an enormous amount of popularity, in the good as well as in the bad sense.

Although sharing one's creations or sensor data directly with friends sounds like a great idea, we, however, think that yet we are pretty far from this, as there barely is any programming models or tools (in the context of fog computing and WoT). Moreover, the danger also is that this would introduce too many new security and privacy threats, as this is the case already with the things created by the industry as we discuss later.

5.3 RQ3: Is interoperability with other systems supported, and how can this aspect be improved?

Often standardisation is considered to be the key to interoperability. The World Wide Web is one of the most successful technologies in interoperability wise, and now it is going towards harnessing physical objects to augment us in our daily lives. For this reason, it is vital that the World Wide Web Consortium (W3C) is drafting an abstract WoT architecture [40]. In the draft, they mention the objective of this work to be that the WoT is intended to enable interoperability across IoT platforms and application domains. The fundamental idea is to maximise using the existing and emerging tools to be used on for building new IoT scenarios.

As part of the same work, W3C is also drafting some other standards: WoT Thing Description [41], WoT Binding Templates [42], and WoT Scripting API [43]. Of these, the WoT Thing Description is the primary building block, which role is to describe an interface of a thing (WoT Interface) so that other things can then interact with other services and things. The role of WoT Binding Template is to enable binding the interface with multiple protocols. A thing may use WoT Scripting API internally, which means that the application logic can be done using JavaScript. The above is supposed to simplify the development significantly and enable moving the developed components fluidly. In fog computing, this would become very useful and could be used in various use cases.

The work by W3C has a lot in common with Evrythng's Web Thing Model approach. While Evrythng's approach has been developed already since 2014, the W3C member submission was submitted on 2017 [44, 45]. The W3C approach can be considered as one of the leading approaches. However, there are a number of other hypermedia API-level abstractions for constructing WoT applications and for improving the interoperability that come from consortiums as well as from individual authors. These with include: JSON hypertext application language (JSON-HAL) [46]; media types for hypertext sensor markup (HSML) [47]; constrained RESTful application language [48]; and Web Thing API by Mozilla [49].

One of the main purposes of the all above approaches (except JSON HAL) is to offer bindings for different protocols to support programming interactions between the things. However, considering how intuitive it is to use the approach by developer it is also important to offer high-level architecture definitions. From the above approaches, Mozilla's Web Thing API, Evrythng's Web Thing Model, and W3C approach are the only ones that offer such definitions. Mozilla's Web Thing API is one of the most recent approaches and has a lot in common with the Evrythng's Web Thing Model approach. The fact that this approach comes from a company may, however, be a downside since other companies may not be willing to follow this approach. On the other hand, it is also a positive thing that the approach comes from a big company with a long tradition of open-source publishing their works is good thing since it supports continuity. For example, some of the approaches coming from small groups or individual authors (JSON-HAL [46] and HSML [47]) seem to have expired for now. Nevertheless, all these attempts for standardisation are important since these highlight the importance of the standardisation work for WoT. A more in-depth comparison of these APIs has been conducted by Martins, Mazayev, and Correia in their paper 'Hypermedia APIs for the Web of Things' [44].

At present, the IoT has a strong focus on establishing connectivity between a variety of constrained devices and services. Therefore, the next logical goal is to build on top of this connectivity and begin focusing on the application layer. Thus, in contrast to the IoT approaches, it would be great if the standardisation work by W3C serves its purpose as this would enable considering the devices as first-class citizens of the Web. If the developers would have clear abstractions and they could consider that the connection between the things established, this would allow the developers to focus on building the applications.

We believe that at large, the WoT has all the potential to materialise into an open ecosystem of digitally augmented physical objects and new experiences which genuinely can help people in their lives. At this point, we are not there yet, and plenty of research must be conducted on improving the interoperability between the things, as well as between cloud services.

5.4 RQ4: what are the current security and privacy issues, and can these threats be covered?

WoT involves numerous heterogeneous entities interacting with each other. Given the enormous number of connected devices that are potentially vulnerable, security and privacy protection became extremely necessary [50]. In fact, poorly secured interconnected (malicious) IoT devices could serve as entry points for cyber attacks towards more critical targets. In this section, we give some considerations on current IoT technology and related security breach and solutions (if any). Source of the following discussion comes from some recent state of the art survey in the area [51–55] as well as security issues analysis of the WoT [50], where a more through discussion can be found.

The attacker model for IoT architecture is described in [56] where the attacker can be a malicious user, a bad manufacturer, or an external adversary:

- The malicious user is the owner of the IoT device with the potential to perform attacks to learn the secrets of the manufacturer and gain access to restricted functionality. By uncovering the flaws in the system, the malicious user can obtain information, sell secrets to third parties, or even attack similar systems.
- The bad manufacturer is the producer of the device with the ability to exploit the technology to gain information about the users, or other IoT devices. Such a manufacturer can deliberately introduce security holes in its design to be exploited in the future for accessing the user's data and exposing it to third parties. Equally, the production of poorly secured goods results in compromising the users' privacy. Besides, in IoT context, where different objects are connected to each other, a manufacturer can attack other competitors' devices to harm their reputation.
- The external adversary is an outside entity that is not part of the system and has no authorised access to it. An adversary would try to gain information about the user of the system for malicious purposes such as causing financial damage and undermining the user's credibility.

Notice that this is indeed different from classic Dolev-Yaho model [57], where the adversary can overhear, intercept, and synthesise any message and is only limited by the constraints of the cryptographic methods used. In other words: 'the attacker carries the message' and has a sort of 'omnipotence' not easy to implement and verify, and usually considered diminished.

In [58], the authors report an in-depth study of possible weakness and their exploitation on IoT devices. In particular, a study of HP conducted some years ago [59], analysed ten of the most popular IoT devices on the market and revealed a generalised poor security level of the majority of them: most of the devices showed privacy and confidential information leakage; two-thirds of them used too low authentication requirements, and not strong enough password used; only some of them used encrypted network services, and suffer from XSS weakness.

The most common and easily addressable security issues of IoT devices reported by HP in 2015 include [50, 59]:

- *Privacy concerns*: The study reports that 80% of the devices were leaking private information.
- *Insufficient authorisation*: According to the study 80% of the tested devices were not protected by a proper password.
- *Lack of transport encryption*: According to the study 70% of the tested devices did not encrypt communication.
- *Insecure Web interface*: According to the study 60% of the tested devices had security concerns in their Web-based user interfaces, and 70% of the systems behind the devices allowed resolving the users' accounts.
- *Inadequate software protection*: The study reports that 60% of the devices were not securing their software updates with encryption.

In [60], the IoT architecture is composed of three layers (perception, network and application). We follow such view and we will analyse security issue in each of the levels.

5.4.1 Perception layer: The perception layer is strictly connected to the technology used for the communication. Wireless sensor network (WSN) is the general term to classify all such (mesh) connected devices, from centimetres to several meters of distances. When dealing with low distance technology and protocol (such as near field communication (NFC) or wireless network of wearable devices (BAN)) built following 802.15.6 standards, security is inferior. Indeed, NFC suffers from many threats (denial of service, and information leakage) [61]. Moreover, since for backward compatibility reason with RFID it is not encrypted, it suffers from many man-in-the-middle attacks, using antennas or skimmers to intercept the signals.

WSN also refers to protocols connected to the 802.15.4 standard, whose major implementations are ZigBee and 6LowPan (and maybe someway still Bluetooth). Bluetooth is indeed the oldest of the three technologies. Despite it is currently adopted for indoor application with iBeacons, and the particular care of energy consumption in the low-energy (BLE) version [62, 63], the technology will be probably substituted for security reasons soon by the more advanced implementation of Zigbee and 6LowPan. In fact, despite BLE uses AES-128 CCM for encryption and authentication purposes, it still suffers from many vulnerabilities, and at today many of the countermeasures rely upon user security problem awareness.

ZigBee represents a new protocol for WSN and the main used implementation of the IEEE standard 802.15.4. Differently, from Bluetooth, the ZigBee protocol came natively with security features and management of both long-term and session keys. The long-term (Master Key) is part of the factory setting, while all the devices share the session (network) keys on the network. ZigBee uses AES-128 encryption as the default, however, since often some trade-offs between security and power consumption [64] and some threats such as traffic sniffing (eavesdropping), packet decoding, and data manipulation/injection could be possible.

6LowPan (that stand for IPv6 low-power personal area networks) is the newest WSN standard. Its main innovation is the use of an IPv6 address for each sensor in the mesh. The use of IPv6 addressing gives to 6LowPan universality, extensibility and stability [65], and wrt IPv6 small packet size and low bandwidth. Unfortunately, devices that support 6LowPan are still resource consuming and this is the primary challenge to solve in the future for global adoption of 6lowPan. However, from the security point of view, 6LowPan implements elliptic curve cryptography encryption algorithm that has smaller-packet sizes w.r.t. RSA.

5.4.2 Middleware and application layers: The middleware layer in IoT contains a vast number of proposals, each of them with their pros and cons. Indeed, no real standard is used in this layer because all the vendors propose their solutions. Describing numerous solutions and their security concerns would result in a not complete description, and thus we suggest to the reader to start from the top survey by Razzaque *et al.* [66].

The application layer, on the contrary, is today enough standardised and the majority of the implementations use the message queue telemetry transport (MQTT) protocol. The MQTT protocol was proposed by Stanford-Clark and Nipper [67] as a light-weight protocol designed for constrained devices and low-bandwidth. At today MQTT is an OASIS and ISO/IEC JTC1 standard [68]. The OASIS MQTT TC is producing a standard for the MQTT protocol compatible with MQTT V3.1, together with requirements for enhancements, documented usage examples, best practices, and guidance for the use of MQTT topics with a commonly available registry and discovery mechanisms. The standard supports bi-directional messaging to uniformly handle both signals and commands, deterministic message delivery, necessary QoS levels, always/sometimes-connected scenarios, loose coupling, and scalability to support large numbers of devices. Candidates for enhancements include message priority and expiry, message payload typing, request/reply, and subscription expiry. As an IoT connectivity protocol, MQTT is designed to support messaging transport from remote locations/devices involving small code footprints (e.g. 8-bit, 256 KB ram controllers), low power, low bandwidth, high-cost connections, high latency, variable availability, and negotiated delivery guarantees.

However, the present implementation of MQTT provides support for only identity, authentication and authorisation policies. Identity specifies the client that is being authorised. Authentication provides the identity of the client and authorisation is the management of rights given to the client. The primary approaches used to support these policies are by using a username/password pair, which is set by the client, for identification or by authentication performed by the MQTT server via client certificate validation through the SSL protocol. The MQTT server identifies itself with its IP address and digital certificate. Its communication uses TCP as transport layer protocol. By itself, the MQTT protocol does not provide encrypted communication. Authorisation is also not part of MQTT protocol but can be provided by the servers. MQTT authorisation rules control which client can connect to the server and what topics a client can subscribe to or publish.

In addition to investigating and discussing other security and privacy challenges of introducing fog computing in IoT environments, Alrawais *et al.* [69] necessitate that the fog computing research should be focused on how to overcome the challenges related to authentication in the context of fog computing in IoT applications since this is far from trivial in such decentralised environment. Also Zhang *et al.* discuss about similar security and privacy threats towards IoT applications and discuss the security and privacy requirements in fog computing in [70]. Further, they demonstrate potential challenges to secure fog computing and review the state-of-the-art solutions used to address security and privacy issues in fog computing for IoT applications and define several open research issues. In our recent paper, we have presented a possible solution in [37] by forming trusted coalitions of devices [71]. The approach is connectivity agnostic, but requires a library support for the leveraged device which may limit its usage in WoT environment.

5.5 Threats to validity

We discuss threats to the validity of this work in the different steps of our study.

5.5.1 Threats to the identification of primary studies: We found it challenging to find the scope of our study. In this article for WoT special issue, we focus on the avenues of future research. In the paper, we assume that the emerging fog computing will eventually enclose cloud computing. Hence, we reviewed some of the most recent fog computing articles and mapped the KET that were mentioned to foster the paradigm shift. The microservice architectural style and communication-related aspects were then present in most articles, and thus formed the core of our study and were used as search terms for finding papers. The importance of data and its analysis was also a significant theme among fog computing and WoT papers, and for this reason, CEP was used as a search term. The IoT and WoT are expected to increase privacy and security threats dramatically, and thus we decided to focus primarily on this aspect while selecting the studies. While this

protocol helped us to focus on selecting the major themes for our study, we naturally were not able to include all the minor themes.

5.5.2 Threats to selection and data extraction consistency: We formulated the research questions RQ1–RQ4 which helped us to select the relevant papers for our study. Naturally, new studies are being published all the time, especially related to security and privacy issues, fog computing, and microservices since these topics are very relevant at the moment. We also intentionally excluded thesis as well as some older papers from our study.

5.5.3 Threats to data synthesis and results: We tried to mitigate the threat with a standard protocol of several steps, by piloting and by externally evaluating our process by professors that were not participating in making our study.

6 Conclusions

In this paper, we focused on describing the current technical design space for WoT, focusing on the emerging fog computing paradigm. We went through the critical enabling technologies that have gained particular popularity among the developers, and which form the basis of building WoT systems and software. Afterwards, we defined research questions of WoT research and outlined motivation for future research in this area. We described some research areas trying to solve our outlined research challenges. Then, we continued the discussion on the retrospective.

The outcome of this work is that WoT research should take a new path, moving from centralised RESTful and cloud service-based approaches towards more decentralised approaches, and aim at leveraging the full potential of the dynamic and modern computing environment, reaching from the edge devices and network nodes to the clouds. As the modern computing environment is heterogeneous, the Web-based technologies give a good base and support the fluidity required by the fog computing. Like all technologies, also Web technologies have their flaws which mainly are related to their limited support in some hardware platforms and some limitations in accessing the hardware resources. Nevertheless, due to the openness and support for heterogeneity, Web technologies in the context of the IoT have earned their place, and thus will continue growing their popularity among the IoT development.

7 Acknowledgments

The work of N. Mäkitalo was supported by the Academy of Finland (project 295913).

8 References

[1] Stirbu, V.: 'Towards a restful plug and play experience in the Web of Things'. 2008 IEEE Int. Conf. Semantic Computing, August 2008, pp. 512–517
[2] Guinard, D.: 'Towards the Web of Things: Web mashups for embedded devices'. In MEM 2009 in Proc. WWW 2009, 2009
[3] Guinard, D., Trifa, V., Mattern, F., *et al.*: 'From the Internet of Things to the Web of Things: resource-oriented architecture and best practices', in Uckelmann, D., Harrison, M., Michahelles, F. (Eds.): '*Architecting the Internet of Things*' (Springer, Berlin, Heidelberg, 2011), pp. 97–129
[4] Tran, N.K., Sheng, Q.Z., Babar, M.A., *et al.*: 'Searching the Web OF Things: state of the art, challenges, and solutions', *ACM Comput. Surv.*, 2017, **50**, pp. 55:1–55:34
[5] Bonomi, F., Milito, R., Zhu, J., *et al.*: 'Fog computing and its role in the Internet of Things'. Proc. First Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13–16
[6] Dastjerdi, A.V., Buyya, R.: 'Fog computing: helping the Internet of Things realize its potential', *Computer*, 2016, **49**, pp. 112–116
[7] Taivalsaari, A., Mikkonen, T., Anttonen, M., *et al.*: 'The death of binary software: end user software moves to the Web'. Creating, Connecting and Collaborating Through Computing (C5), 2011 Ninth Int. Conf., January 2011, pp. 17–23
[8] Shi, W., Dustdar, S.: 'The promise of edge computing', *Computer*, 2016, **49**, pp. 78–81
[9] Taivalsaari, A., Mikkonen, T.: 'A roadmap to the programmable world: software challenges in the IoT era', *IEEE Softw.*, 2017, **34**, pp. 72–80
[10] Mäkitalo, N., Aaltonen, T., Mikkonen, T.: 'Coordinating proactive social devices in a mobile cloud: lessons learned and a way forward'. Proc. Int. Conf. Mobile Software Engineering and Systems, MOBILESoft '16, New York, NY, USA, 2016, pp. 179–188
[11] Fielding, R.T.: 'REST: architectural styles and the design of network-based software architectures'. Doctoral dissertation, University of California, Irvine, 2000
[12] Internet engineering task force (IETF), RFC 7540 – hypertext transfer protocol version 2 (HTTP/2), May 2015
[13] Bermbach, D., Pallas, F., Pérez, D.G., *et al.*: 'A research perspective on Fog computing'. Proc. 2nd Workshop on IoT Systems Provisioning & Management for Context-Aware Smart Cities, Springer, 2017
[14] Gallidabino, A., Pautasso, C., Ilvonen, V., *et al.*: 'Architecting liquid software', *J. Web Eng.*, 2017, **16**, (5&6), pp. 433–470
[15] Luckham, D.: '*The power of events*' (Addison-Wesley, Reading, 2002), **vol. 204**
[16] Cugola, G., Margara, A.: 'Processing flows of information: from data stream to complex event processing', *ACM Comput. Surv. (CSUR)*, 2012, **44**, (3), p. 15
[17] Dayarathna, M., Perera, S.: 'Recent advancements in event processing', *ACM Comput. Surv.*, 2018, **51**, pp. 33:1–33:36
[18] Randika, H., Martin, H., Sampath, D., *et al.*: 'Scalable fault tolerant architecture for complex event processing systems'. Advances in ICT for Emerging Regions (ICTer), 2010 Int. Conf., 2010, pp. 86–96
[19] Zhang, H., Diao, Y., Immerman, N.: 'On complexity and optimization of expensive queries in complex event processing'. Proc. 2014 ACM SIGMOD Int. Conf. Management of data, 2014, pp. 217–228
[20] Cugola, G., Margara, A.: 'Deployment strategies for distributed complex event processing', *Computing*, 2013, **95**, (2), pp. 129–156
[21] Soto, J.A.C., Jentsch, M., Preuveneers, D., *et al.*: 'CEML: mixing and moving complex event processing and machine learning to the edge of the network for IoT applications'. Proc. 6th Int. Conf. Internet of Things, IoT'16, New York, NY, USA, 2016, pp. 103–110
[22] Mayer, R., Tariq, M.A., Rothermel, K.: 'Minimizing communication overhead in window-based parallel complex event processing'. Proc. 11th ACM Int. Conf. Distributed and Event-based Systems, 2017, pp. 54–65
[23] Starks, F., Plagemann, T.P., Kristiansen, S.: 'DCEP-SIM: an open simulation framework for distributed CEP'. Proc. 11th ACM Int. Conf. Distributed and Event-based Systems, 2017, pp. 180–190
[24] Alshuqayran, N., Ali, N., Evans, R.: 'A systematic mapping study in microservice architecture'. 2016 IEEE 9th Int. Conf. Service-Oriented Computing and Applications (SOCA), November 2016, pp. 44–51
[25] Francesco, P.D., Malavolta, I., Lago, P.: 'Research on architecting microservices: trends, focus, and potential for industrial adoption'. 2017 IEEE Int. Conf. Software Architecture (ICSA), April 2017, pp. 21–30
[26] Lewis, J., Fowler, M.: 'Microservices'. Available at https://martinfowler.com/articles/microservices.html, accessed March 2015
[27] Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., *et al.*: 'Microservices: yesterday, today, and tomorrow', CoRR, vol. abs/1606.04036, 2016
[28] Thönes, J.: 'Microservices', *IEEE Softw.*, 2015, **32**, pp. 116–116
[29] Pahl, C., Jamshidi, P.: 'Microservices: a systematic mapping study'. CLOSER, 2016, vol. 1, pp. 137–146
[30] Mayer, S., Karam, D.S.: 'A computational space for the Web of Things'. Proc. Third Int. Workshop on the Web of Things, 2012, p. 8
[31] Aggarwal, C.C., Ashish, N., Sheth, A.: 'The Internet of Things: a survey from the data-centric perspective', in '*Managing and mining sensor data*' (Springer, Boston, MA, USA, 2013), pp. 383–428
[32] Guinard, D., Trifa, V.: '*Building the Web of Things: with examples in node. js and raspberry pi*' (Manning Publications Co., Greenwich, CT, USA, 2016)
[33] Mahmud, R., Kotagiri, R., Buyya, R.: 'Fog computing: a taxonomy, survey and future directions', in '*Internet of everything*' (Springer, Singapore, 2018), pp. 103–130
[34] Wen, Z., Yang, R., Garraghan, P., *et al.*: 'Fog orchestration for IoT services: issues, challenges and directions', *IEEE Internet Comput.*, 2017, **21**, (2), pp. 16–24
[35] Byers, C.C.: 'Architectural imperatives for fog computing: use cases, requirements, and architectural techniques for fog-enabled IoT networks', *IEEE Commun. Mag.*, 2017, **55**, (8), pp. 14–20
[36] Rahmani, A.M., Gia, T.N., Negash, B., *et al.*: 'Exploiting smart e-health gateways at the edge of healthcare Internet-of-Things: a fog computing approach', *Future Gener. Comput. Syst.*, 2018, **78**, pp. 641–658
[37] Mäkitalo, N., Ometov, A., Kannisto, J., *et al.*: 'Safe and secure execution at the network edge: a framework for coordinating cloud, fog, and edge', *IEEE Softw.*, 2018
[38] Guinard, D., Trifa, V., Wilde, E.: 'A resource oriented architecture for the Web of Things', Internet of Things (IOT), 2010, pp. 1–8
[39] Fielding, R.T., Taylor, R.N.: 'Principled design of the modern Web architecture', *ACM Trans. Internet Technol. (TOIT)*, 2002, **2**, (2), pp. 115–150
[40] Kajimoto, K., Kovatsch, M., Davuluru, U.: 'Web of Things (WoT) architecture', Technical Report, World Wide Web Consortium (W3C), 09 2017
[41] Kaebisch, S., Kamiya, T.: 'Web of Things (WoT) thing description', Technical Report, World Wide Web Consortium (W3C), 09 2017
[42] Koster, M.: 'Web of Things (WoT) protocol binding templates', Technical Report, World Wide Web Consortium (W3C), 10 2017
[43] Kis, Z., Nimura, K., Peintner, D., *et al.*: 'Web of Things (WoT) scripting API', Technical Report, World Wide Web Consortium (W3C), 10 2017
[44] Martins, J. A., Mazayev, A., Correia, N.: 'Hypermedia APIS for the Web of Things', *IEEE Access*, 2017, **5**, pp. 20058–20067
[45] Trifa, V., Guinard, D., Carrera, D.: 'Web thing model W3C Member Submission', Technical Report, World Wide Web Consortium (W3C), 04 2017
[46] Kelly, M.: 'JSON hypertext application language (JSON-HAL)', Technical Report, IETF, 11 2016
[47] Koster, M.: 'Media types for hypertext sensor markup (HSML)', Technical Report, IETF, 09 2017

[48] Hartke, K.: 'The constrained RESTful application language (CoRAL)', Technical Report, IETF, 10 2017
[49] Francis, B.: 'Web thing API (unofficial draft)', Technical Report, World Wide Web Consortium (W3C), 03 2018
[50] El Jaouhari, S., Bouabdallah, A., Bonnin, J.-M.: 'Security issues of the web of things', in '*Managing the web of things*' (Elsevier, Boston, MA, USA, 2017), pp. 389–424
[51] Mendez, D.M., Papapanagiotou, I., Yang, B.: 'Internet of Things: survey on security and privacy', CoRR, vol. abs/1707.01879, 2017
[52] Yang, Y., Wu, L., Yin, G., *et al.*: 'A survey on security and privacy issues in internet-of-things', *IEEE Internet Things J.*, 2017, **4**, (5), pp. 1250–1258
[53] Alaba, F.A., Othman, M., Hashem, I.A.T., *et al.*: 'Internet of Things security: a survey', *J. Netw. Comput. Appl.*, 2017, **88**, pp. 10–28
[54] Granjal, J., Monteiro, E., Silva, J.S.: 'Security for the Internet of Things: a survey of existing protocols and open research issues', *IEEE Commun. Surv. Tutorials*, 2015, **17**, (3), pp. 1294–1312
[55] Fremantle, P., Scott, P.: 'A survey of secure middleware for the Internet of Things', *Peer J. Comput. Sci.*, 2017, **3, p**. e114
[56] Atamli, A., Martin, A.P.: 'Threat-based security analysis for the Internet of Things', 2014 Int. Workshop on Secure Internet of Things, SIoT 2014, Wroclaw, Poland, September 10, 2014, pp. 35–43
[57] Dolev, D., Yao, A.C.: 'On the security of public key protocols', *IEEE Trans. Inf. Theory*, 1983, **29**, (2), pp. 198–207
[58] Dragoni, N., Giaretta, A., Mazzara, M.: 'The internet of hackable things', CoRR, vol. abs/1707.08380, 2017
[59] 'Internet of Things research study'. Available at http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676, 2015, accessed 30 November 2017
[60] Zhao, K., Ge, L.: 'A survey on the Internet of Things security'. Ninth Int. Conf. Computational Intelligence and Security, CIS 2013, EMEI Mountain, Sichan Province, China, December 14–15, 2013, pp. 663–667
[61] Madlmayr, G., Langer, J., Kantner, C., *et al.*: 'NFC devices: security and privacy'. Proc. Third Int. Conf. Availability, Reliability and Security, ARES 2008, Technical University of Catalonia, Barcelona, Spain, March 4–7 2008, pp. 642–647
[62] Zafari, F., Papapanagiotou, I.: 'Enhancing ibeacon based micro-location with particle filtering'. 2015 IEEE Global Communications Conf., GLOBECOM 2015, San Diego, CA, USA, December 6–10 2015, pp. 1–7
[63] Zafari, F., Papapanagiotou, I., Devetsikiotis, M., *et al.*: 'Enhancing the accuracy of ibeacons for indoor proximity-based services'. IEEE Int. Conf. Communications, ICC 2017, Paris, France, May 21–25 2017, pp. 1–7
[64] Boyle, D., Newe, T.: 'Securing wireless sensor networks: security architectures', *JNW*, 2008, **3**, (1), pp. 65–77
[65] Sheng, Z., Yang, S., Yu, Y., *et al.*: 'A survey on the IETF protocol suite for the Internet of Things: standards, challenges, and opportunities', *IEEE Wirel. Commun.*, 2013, **20**, pp. 91–98
[66] Razzaque, M. A., Milojevic-Jevric, M., Palade, A., *et al.*: 'Middleware for Internet of Things: a survey', *IEEE Internet Things J.*, 2016, **3**, (1), pp. 70–95
[67] OASIS Message Queuing Telemetry Transport (MQTT) TC: 'MQTT Version 3.1.1', Standard, Oasis, October 2014. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard. Available at http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
[68] I.J.I. Technology.: 'ISO/IEC 20922:2016', Standard, International Organization for Standardization (ISO), June 2016
[69] Alrawais, A., Alhothaily, A., Hu, C., *et al.*: 'Fog computing for the Internet of Things: security and privacy issues', *IEEE Internet Comput.*, 2017, **21**, (2), pp. 34–42
[70] Ni, J., Zhang, K., Lin, X., *et al.*: 'Securing fog computing for Internet of Things applications: challenges and solutions', *IEEE Commun. Surv. Tutorials*, 2017
[71] Ometov, A., Masek, P., Urama, J., *et al.*: 'Implementing secure network-assisted d2d framework in live 3GPP LTE deployment'. 2016 IEEE Int. Conf. Communications Workshops (ICC), Kuala Lumpur, Malaysia, May 2016, pp. 749–754