How to see through the Fog? Using Peer to Peer (P2P) for the Internet of Things

David Tracey[1]
Dept. Of Computer Science,
University College Cork, Ireland

Cormac Sreenan
Dept. Of Computer Science,
University College Cork, Ireland

***Abstract*—The Internet of Things (IoT) faces the challenge of scaling to handle tens of billions of connected devices. This challenge is made more difficult by the range of constituent IoT parts from Cloud-based applications to constrained nodes in Wireless Sensor Networks (WSNs). Achieving the desired scale and interoperability requires an architecture for IoT that is scalable and allows seamless operation across networks and devices. This paper considers the requirements for IoT and considers a number of existing architectural approaches and the emergence of Fog computing. It proposes that Fog computing architectures must cater for the flow of data from constrained sensor nodes to powerful applications. It considers the suitability of a Peer to Peer (P2P) approach for Fog computing. Using a prototype implementation, it demonstrates how a Holistic Peer to Peer (HPP) architecture and application layer protocol meet the requirements set for IoT.**

Keywords–Internet of Things(IoT), RESTFul Style, Fog Computing, Wireless Sensor Network (WSN), Peer to Peer (P2P).

I. INTRODUCTION

Wireless Sensor Networks (WSNs) interact with the physical world allowing nodes to be deployed close to sensed phenomena. The "Internet of Things" (IoT) is a distributed system of devices and applications for sensing, actuation and computation. The estimated tens of billions of connected IoT devices [1] will need solutions to program and manage them, with services to gather, store and analyze vast amounts of data. Cloud services and Big Data approaches allow the scalable storage and analysis of this data, while Fog and Edge computing offer rich functionality at the edge of the Internet. Protocols, such as the Constrained Application Protocol (CoAP) [2] and data models, e.g. IPSO (Smart Objects) [3], have emerged to support better application interoperability.

The potential of IoT is, however, limited by the difficulties imposed by the constrained processing, memory and energy consumption of WSN nodes and their heterogeneous nature, limited development environments and diverse software and protocols. A key challenge with such diversity is to enable the growth of IoT in terms of scalability and also of developing services and node software. The scalability challenge requires being able to scale up to billions of devices, but also to scale down to resource-constrained devices in relatively small WSNs [4]. This requires seamless interoperability and a consistent set of abstractions and APIs/protocols, particularly at the application layer to realise Mark Weiser's vision of tiny networked computers woven into everyday life [5]. [6] highlights the importance of interoperability: "Of the total potential economic value the IoT enables, interoperability is required for 40 percent on average and for nearly 60 percent in some settings". Approaches demonstrating scalability and interoperability at Internet-scale and the value of an architectural approach include the RESTFul architectural style [7] and BitTorrent [8]. The RESTFul style is based on specified constraints and components. Its success can be seen in the development of HTTP1.1 and the wide adoption of REST APIs. BitTorrent has also shown that Peer-to-Peer (P2P) can provide a low barrier to entry, greater autonomy, scale and robustness.

This paper considers which architectural approaches may be suitable for IoT and how they relate to Fog computing. As such, we present a set of requirements for IoT and consider lessons from the RESTFul style and BitTorrent to meet those requirements. This paper proposes that meeting the challenges presented by IoT requires an architecture and a set of consistent abstractions for all components in the entire flow as data is sent (and aggregated/stored/acted on) from constrained devices to edge gateways to Cloud services. As such, this paper proposes that Fog computing [9] architectures must consider constrained devices as part of that flow and that a P2P overlay network can be used to achieve scalability and high availability, especially at the edges of the Internet as in Fog computing. In this context, we present a Holistic Peer-to-Peer (HPP) application layer protocol that has been extended to use a Distributed Hash Table (DHT) based on Kademlia [10] as part of our previously presented architecture [11]. Implementations of the architecture have demonstrated interoperability by allowing a constrained node to provide IPSO data using HPP and CoAP and shown a straightforward service to integrate with HBase.

The remainder of this paper is organized as follows. Section II presents requirements for IoT, section III reviews a number of architectural approaches and section IV gives an overview of our Holistic Peer to Peer (HPP) architecture and the addition of a DHT. The paper concludes in section V.

II. IOT ARCHITECTURAL REQUIREMENTS

The requirements for an IoT architecture in [11] have been refined and it should:

- define the roles of nodes running services. Nodes must meet a minimum level of functionality, e.g. respond to a request for its capabilities (**Req-1**).

[1] David Tracey is employed by Rapid7, Dublin, Ireland.

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

- provide abstractions to support the basic operations required of a sensor node and the services using it. These must map easily to a range of heterogeneous devices and higher level services (**Req-2**).
- be independent of particular node hardware and handle a range of node capabilities (**Req-3**).
- provide simple, consistent APIs for developers of device and application software (**Req-4**).
- provide a consistent means to exchange information independent of the underlying technology and support the modelling of (sensor) data to allow its use by higher level services (**Req-5**).
- support a (sensor) node informing other nodes and services of its capabilities (**Req-6**).
- be dynamic to handle small, static networks and adapt as the network changes and support applications discovering and collaborating without a centralized coordination facility (**Req-7**). It must also be robust to support challenging wireless environments [12].
- use protocols that are sufficiently simple for low capability devices to participate (**Req-8**).

These requirements for nodes and services would enable an "Opportunistic IoT Service", which is defined to provide an "*interface that allows an IoT entity to be engaged, under specific constraints and pre/post-conditions, in a temporary, contextualized and localized usage relationship*" [13].

III. ARCHITECTURAL APPROACHES

A. RESTFul Architectural Style

The RESTFul architectural style uses a *resource* as a key abstraction of information that can be represented in a number of representations using the Internet media types. It is based on the following five interface constraints in [7]:

- All important resources are identified by one resource identifier. This is generally a Universal Resource Identifier (URI). This constraint leads to the interface being simple, visible, and reusable.
- Access methods have the same semantics for all resources. For HTTP, this results in a limited set of verbs, such as HEAD, GET, POST, PUT, DELETE with easily understood semantics. This leads to the interface being visible, scalable, and available.
- Resources are manipulated through the exchange of representations. This constraint leads to the interface being simple, visible, reusable, cacheable and evolvable using information hiding.
- Representations are exchanged via self-descriptive messages. This leads to the interface being visible, scalable, available and evolvable.
- Hypertext as the engine of application state. This leads to the interface being simple, visible, reusable, and cacheable through data-oriented integration, evolvable via loose coupling, and adaptable though late binding of application transitions.

The RESTFul architectural style also includes processing elements that are determined by their roles, i.e. origin server, gateway proxy, user agent. A recent paper reflecting on the RESTFul architectural style [14], including the original authors, considers that there have been different interpretations of the term REST, but reiterates that *"REST is not an architecture, but rather an architectural style. It is a set of constraints that, when adhered to, will induce a set of properties; most of those properties are believed to be beneficial for decentralized, network-based applications, while others are the negative trade-offs that can result from any design choice*". Importantly it also states that "*REST does not directly constrain the Web's architecture. Rather, an application developer may choose to constrain an architecture in accordance with the REST style*". The RESTFul style has been shown to facilitate application development and scalability as a result of its decoupled nature.

CoAP [2] is a specialized protocol for constrained nodes and constrained (e.g. low-power, lossy) networks. It was originally a binary format on top of UDP, but has been extended to also support TCP and TLS in RFC8323[2]. It uses RESTFul concepts such as URIs, with its own schema coap://, and media formats. It is designed to be easy to proxy to/from HTTP. CoAP provides resource discovery via the Resource Directory (RD) and specific message types to provide reliability. The use of an "observe" flag in the GET Request provides observe/notify on a given resource. RESTful approaches with CoAP are increasing [15], e.g. an end-to-end IP based architecture for greenhouse monitoring integrating CoAP over a 6LowPAN WSN using Contiki [16].

B. Middleware Approaches

Middleware is software that acts as an intermediary between IoT devices and applications. [17] considers three types of IoT middleware: service-based, cloud-based and actor-based. Service-based is a service-oriented architecture (SOA) where IoT devices may be represented as services. Cloud-based services allow users to upload their sensor data to the Cloud for storage, querying and analysis using Cloud databases, NoSQL stores and Machine Learning toolsets. These offerings use a proxy/gateway and APIs to provide the integration with a Cloud service, usually limited to a given Cloud provider and perhaps to a given device environment. The actor-based architecture exposes IoT devices as reusable, distributed actors. It is designed to be lightweight and flexible enough to run in all components according to their capability, e.g. a constrained node might not include a storage service.

The service-based and cloud-based middleware generally provide separate components and abstractions as the systems become more capable. For example, Figure 1 shows the three Eclipse software stacks[3]. The first stack is for constrained devices, showing OS, Hardware Abstraction and Communication layers, with remote management across layers. The second stack is for Gateways, which aggregate data and coordinate the connectivity of these devices to each other and to an external network with layers to support IoT

[2] "RFC8323, Constrained Application Protocol) over TCP, TLS, and WebSockets," 2018. https://datatracker.ietf.org/doc/rfc8323

[3] "Eclipse IoT". https://iot.eclipse.org

protocols, network management and data management/messaging. It runs on an OS with more functionality and may provide container or specific application environments, e.g. for Java. The third stack is for IoT Cloud platforms which is expected to scale horizontally to support a large number of devices and vertically to support a variety of IoT scenarios and devices. It has layers for device management, data management and storage, event management and analytics.

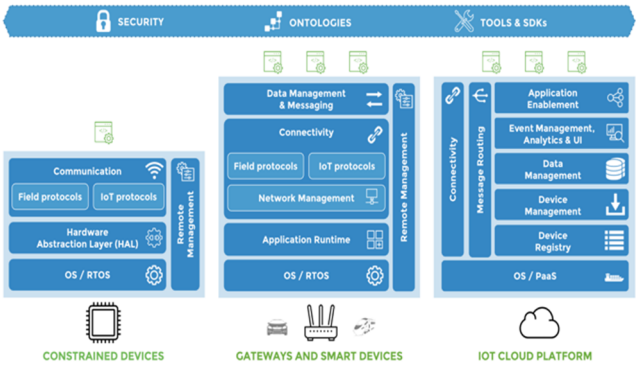

Figure 1 Eclipse IoT Stacks

Other middleware approaches such as Sensation [18] treat the sensor network as an information source similar to a database. It acts as an integration layer between applications and networks, with a high-level set of APIs for applications supported by a proxy for particular WSNs to hide device and network heterogeneity. Agent based middleware composes sensing tasks from sets of services, where the service code moves across nodes autonomously, but requires particular node computational capability and may reduce node lifetime due to the additional associated network traffic [19].

C. *Fog Computing*

Cloud computing is an important part of IoT as it can store and process large amounts of data, providing benefits in scalability, flexibility and cost. There are, however, several issues with simply handling data directly from WSNs:

- Response Time: certain applications may require more rapid response time than the latency introduced by sending data to the Cloud, e.g. connected vehicles.
- Intermittent Connectivity: the Cloud may not meet application requirements for timely data processing or data may be lost if device storage is exceeded.
- Bandwidth: the amount of data from large numbers of devices on a link may exceed the bandwidth available.
- Device Connection: devices may have to be connected directly to each other, e.g. wearable health monitoring devices, connected vehicles.
- Data Security and Privacy: regulation may limit where data can reside, e.g. health data may require specific physical security guarantees.

One approach to address these issues is to move some applications or some of the processing/storage to the edge of the network. Fog computing is "a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of network" [9]. Such devices may consume from and send to the Cloud, as well as load balancing that traffic. In such a federated system, a service may execute using components running in different networks/providers. This requires that Fog/Edge components be interoperable at the level of providers and architecture models and interfaces. The OpenFog Consortium [20] have published an OpenFog Reference Architecture as a basis to develop and test an open fog-enabled architecture.

Distinctions between Fog and Edge computing vary, but tend to use the closeness of the processing to the source of data. Edge computing performs computing on an edge device like a programmable controller and Fog computing performs it at the local network level, e.g. by a gateway or specialized node. There are a number of challenges:

- Scalability: each edge system will manage the data of a set of nodes which will have to scale as nodes are added. The overall system must also scale to manage, deploy and run large numbers of applications as more edge networks are added.
- Heterogeneity: an edge system should handle the storage, computational and operational requirements of heterogenous nodes and services, e.g. the different data formats used by devices.
- Management: discovery and monitoring will be required for fog nodes by the Cloud service and vice versa. A key question is whether this will be orchestrated in the Cloud and to what degree the fog nodes/systems will be autonomic and decentralised.
- Data Security: edge nodes will have different capabilities, which should be considered in deciding where data is stored or processed.

The platforms, applications or services for Fog computing must contribute to the seamless interoperability desired in IoT and not create islands of data and services. An expansive view of Fog and Edge computing is proposed as Osmotic computing [21]. This is based "on the need for a holistic distributed system abstraction enabling the deployment of lightweight microservices on resource-constrained IoT platforms at the network edge, coupled with more complex microservices running on large-scale datacenters". It proposes Edge Micro Data Centres in a federated environment of public/private cloud, edge cloud and devices. It uses microservices in containers on IoT and Edge devices and includes an interoperability layer for remote orchestration of heterogeneous Edge devices.

A recent example of an Edge architecture for healthcare incorporates smart devices for healthcare applications and an edge gateway [22]. The gateway provides layers for access network control, device management, application control, edge management (data processing), middleware to manage data and an API layer for remote interfaces and other edge devices. This architecture provides considerable flexibility in the edge gateway to handle multiple radio interfaces, application layers and device management, including discovery. It considers the flow of data from the device (sensor or smart device) through an edge gateway to Cloud

services with defined roles, but it does not provide a high level set of consistent abstractions.

The Mobile Edge Computing (MEC) approach uses increased edge processing power to handle data streams at the mobile edge [23]. It connects each Base Station to a fog node. This fog node provides local computing resources and a proxy Virtual Machine (VM), which collects, classifies and analyses raw data streams from devices, converts them into metadata and transmits the metadata to the corresponding application VMs (owned by IoT service providers). A Software Defined Networking (SDN) based cellular core is used to forward packets among fog nodes.

D. OpenFog Reference Architecture

The OpenFog Reference Architecture describes a set of high-level attributes of Fog computing termed "pillars". These are security, scalability, openness, autonomy, agility, reliability, hierarchical organization and programmability, and it describes desirable characteristics for each. It considers several scenarios including traffic control, security surveillance and air transportation, which illustrate the range of actors, interactions and the types of device and services involved. Figure 2 from the OpenFog consortium shows their view of an N-tier environment, where the volume of data is reduced as the intelligence from data is increased at each level.

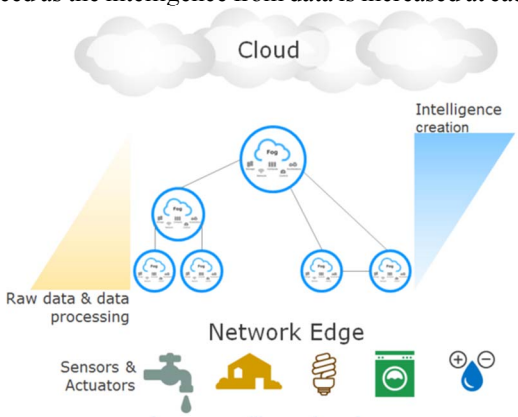

Figure 2 Intelligence from Data

The Architecture considers the four layers of *Devices* (sensors, actuators, cameras), *Monitoring and Control* (control logic using the sensor telemetry, e.g. to generate alerts and events), *Operational Support* (operational analytics) and *Business Support* (such as large-scale historic analysis). The three upper layers may be deployed only on fog nodes or only on cloud nodes, e.g. where the physical infrastructure may not support fog nodes. The OpenFog Consortium also considers the cross-layer perspectives by using three views of the architecture - the "Software" view in the top three layers, the "System" view in the middle layers and the "Node" view in the lower layers.

In terms of WSN constrained nodes, the OpenFog architecture considers "Sensors, Actuators, and Control" as hardware or software-based devices, where several hundred of these could be associated with a single fog node. Some of these nodes may have significant processing capability and be able to implement some basic fog functions. The protocol abstraction layer exists to bring these devices under the supervision of a fog node so that their data can be provided to higher layers. The OpenFog Reference Architecture states that future versions will describe "Minimum Viable Interfaces", with more detail about the protocols and abstraction layers. It currently identifies protocols such as CoAP and MQTT for node-cloud and node-node communications.

E. Autonomic and Cognitive Architectures

Autonomic architectures are another way to realize complex, loosely-coupled, decentralized, dynamic systems. They are characterized by self-configuration, self-healing, self-optimization and self-protection. Cognitive approaches use information based on experience to improve overall performance [24]. These frameworks are generally concerned with higher layer functions such as translating vendor specific data into a vendor-neutral form or semantics around state transitions, reasoning engines and automatic management functions, perhaps using virtualisation.

IV. HPP Architecture

A. Architectural Lessons

A key lesson that can be seen in the RESTFul style is that an interoperable architecture for IoT must provide consistent abstractions to simplify the development and deployment of nodes and applications. While valuable in presenting the range of actors and scenarios, the layered Eclipse and OpenFog architectures distinguish at the level of abstraction between the different entities without providing a consistent set of constraints (as in REST) or abstractions or considering entities as having a Peer relationship as in BitTorrent. This leads to the need for edge proxies with different architectures and abstractions, reducing the ability to achieve large scale [25].

The RESTFul Architectural Style is based on constraints and defined components as above and has proven its scalability and flexibility. Given scale, flexibility and interoperability are desired in IoT, we consider that a similar architecturally driven approach is necessary. Furthermore, the scale and autonomy possible with P2P, as demonstrated by BitTorrent, suggest the use of P2P in IoT is appropriate.

B. The Holistic Peer to Peer (HPP) Architecture

Using the above analysis, a key design constraint for our holistic architecture is the ability to run the same code and consistent abstractions on a constrained node and Cloud services. It also uses a simple P2P protocol with defined roles, including Source, Sink, Forwarder, Bootstrap and Aggregator. It uses concepts from tuple-spaces to share and cache data easily. The architectural analysis also resulted in the addition of a Distributed Hash Table (DHT) since the protocol was presented in [11]. This DHT uses Kademlia k-buckets to provide a P2P Overlay network to identify nodes and groups. This provides the basis for nodes and services to operate in a self-organising manner and allows nodes to act as peers at an application layer throughout the flow of data, regardless of whether they are in a WSN or a Cloud service.

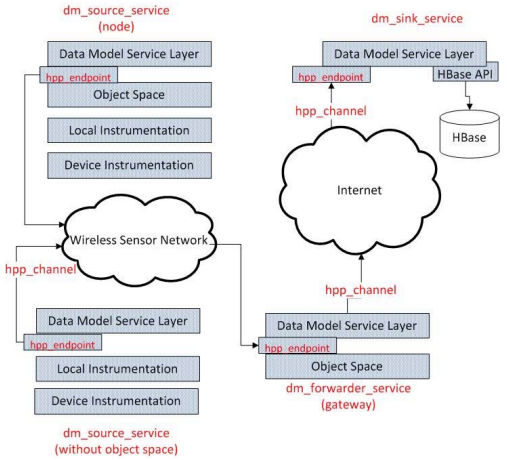

Figure 3 HPP Layers and the Interaction of Node Services

The HPP architecture consists of four layers. The Data Model Service layer provides a high-level abstraction for node data and services (on a node or in a Cloud service). It uses defined roles and abstractions to decouple the application developer from the network and node specifics. It is independent of a particular data model with a simple data store API. The Object Space layer is inspired by tuple-spaces and is a data store with leases and a simple API with a limited set of operations. It holds the node's data or data it has cached from remote nodes with a cache algorithm that uses leases in its cache replacement policy. The Local Instrumentation layer maps the data from hardware on a node to the Object Space and the Device Instrumentation layer is a device specific interface to a node's OS or hardware.

Figure 3 shows the HPP architecture layers on nodes, based on their capabilities, i.e. a node running only a forwarder service does not have a Local Instrumentation layer and the Cloud Service only has a Data Model Service layer above a datastore. The HPP protocol is sufficiently simple for low capability devices. It provides a consistent means to exchange information between nodes and services, independent of the underlying network, using a small set of simple commands (Hello, Bye, Get, Add, Take, Notify). Every Peer must support Hello and respond with its identifier (if known) and its capabilities. All peers should handle at least a Get for its Peer Instance. Peers may support any of the other HPP messages, as per the capabilities in its Hello reply. A new peer joins a HPP network by sending a Hello message to a known peer address in the desired network.

C. *Evaluation of HPP Architecture*

HPP meets our requirements for IoT as follows: **Req-1** and **Req2** are provided by the Data Model Service layer**. Req-3** is provided by the Local Instrumentation layer. **Req-4** is provided by the Object Space layer. **Req-5** and **Req-6** are provided by the HPP protocol **. Req-7** is provided by the P2P Overlay network using HPP. **Req-8** is met by all layers.

The RESTFul use of a well-known URI for discovery is replaced by the use of a Hello message with one node's address to join a peer network. HPP's limited set of message types in the protocol and object space API aligns with the RESTFul use of methods with the same semantics for all resources. The use of HPP to manipulate objects aligns with the RESTFul use of self-descriptive messages to exchange resource representations.

In terms of the OpenFog Reference Architecture, the Local Instrumentation layer and the Object Space layer align with the Node view of the Sensor and Actuator layer and the Protocol Abstraction layer, but they provide a richer set of abstractions and detail. HPP provides a means to exchange information between fog nodes and Cloud services. The defined Data Model Service layer and its service roles fit the storage functionality and its abstractions could be useful for the layers to manage nodes and for the Application Layers. As such, HPP allows Fog components to be interoperable at the level of providers and architecture models and interfaces. The addition of a DHT to identify and find nodes and data in an application overlay using HPP further extends its capabilities for OpenFog scenarios across a range of nodes and services.

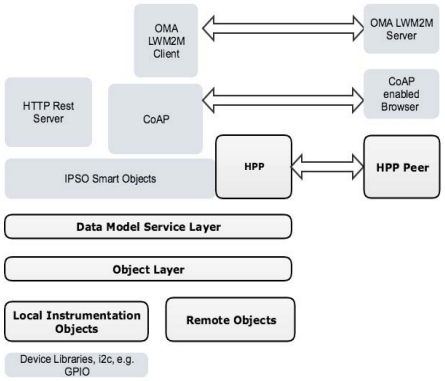

Figure 4 HPP with IPSO and LWM2M

The flexibility of HPP to address the scenarios in Fog computing was shown previously. [11] demonstrated how simple it was to integrate HPP messages and a Data Model (DM) Sink service with HBase as an example of a Cloud NoSQL database. [26] showed that the HPP Data Model layer integrates easily with other models by storing objects from IPSO Smart Objects and Open Mobile Alliance Lightweight specification (OMA LWM2M) and providing them over CoAP. Figure 4 shows recent work where the IPSO objects are stored once in the DM layer and available over the HPP protocol as well as CoAP. The prototype implementation on a constrained Wismote node running the Contiki OS also includes the HPP DHT implementation for the P2P Overlay.

V. CONCLUSION

Based on their successful use at scale, the constraints of the RESTFul architectural style and the scalability and autonomy of P2P BitTorrent peers provide lessons on how to achieve the required scalability and seamless interoperability for IoT. This paper has shown that approaches such as OpenFog focus on the higher layers of the stack(s) for IoT. As also seen in a number of middleware solutions, this results in an architecture which assumes that constrained nodes and

WSNs are so limited that they require proxies or virtualisation to allow them to be handled by higher layers and to make integration and development easier. Furthermore, the goal of seamless interoperability is made even more difficult by the wide diversity of protocols and middleware approaches, each using different architectures and programming abstractions (sometimes even for components in the same architecture).

In contrast, our approach was architecturally driven to meet a set of requirements and constraints for an interoperable and scalable IoT, which includes constrained devices. It was demonstrated that our architecture could be scaled down to run on constrained devices and scaled up to Cloud services in an overlay P2P network, using a DHT that allowed peers in WSNs and external networks to exchange information using the same application layer protocol, without requiring proxies or additional middleware.

This shows that P2P approaches should be considered further in Fog computing. Future work will test the scalability of this P2P architecture and further investigate its alignment with the OpenFog Reference architecture.

REFERENCES

[1] D. Reed, J. R. Larus and D. Gannon, "Imagining the Future: Thoughts on Computing," *Computer,* vol. 45, no. 1, 2012.

[2] Z. Shelby, "RFC 7252, The Constrained Application Protocol (CoAP)," 2014. [Online]. Available: https://datatracker.ietf.org/doc/rfc7252/.

[3] IPSO, "IP for Smart Objects (IPSO) Alliance," 2014. [Online]. Available: http://www.ipso-alliance.org.

[4] M. Kovatsch, "Scalable Web Technology for the Internet of Things (PhD Thesis)," ETH Zurich, 2015.

[5] M. Weiser, "The computer for the twenty-first century," Scientific American, September 1991 (reprinted in IEEE Pervasive Computing, Jan-Mar 2002), 1991.

[6] McKinsey, "The Internet of Things: Mapping the Value Beyond the Hype," 2015. [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world.

[7] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, 2000.

[8] B. Cohen, "The BitTorrent Protocol Specification," 2008. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html.

[9] F. Bonomi, R. Milito, J. Zhu and S. Addepalli , "Fog Computing and Its Role in the Internet of Things," in *MCC Workshop on Mobile Cloud Computing*, 2013.

[10] P. Maymounkov and D. MaziIIeres, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," in *First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[11] D. Tracey and C. Sreenan, "A Holistic Architecture for the Internet of Things, Sensing Services and Big Data," in *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2013.

[12] S. Nawaz, X. Xu, D. Rodenas-Herraiz, P. Fidler, K. Soga and C. Mascolo, "Monitoring a Large Construction Site Using Wireless Sensor Networks," in *RealWSN*, 2015.

[13] G. Fortino, C. Savaglio and M. Zhou, "Toward Opportunistic Services for the Industrial Internet of Things," in *13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017.

[14] R. Fielding, R. N. Taylor, J. R. Erenkrantz, M. M. Gorlick, J. Whitehead, R. Khare and P. Oreiz, "Reflections on the REST Architectural Style and "Principled Design of the Modern Web Architecture"," in *11th Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'17)*, 2017.

[15] M. Kovatsch, "A Low Power CoAP for Contiki," in *IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2011.

[16] W. Colitti, "Integrating Wireless Sensor Networks with the Web," in *Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN)*, 2011.

[17] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal and . Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal,* no. 1, 2017.

[18] T. Hasiotis, "Sensation: A Middleware Integration Platform for Pervasive Applications in Wireless Sensor Networks," in *2nd European Workshop on Wireless Sensor Networks*, 2005.

[19] A. Boulis, "A Framework for Efficient and Programmable Sensor Networks," in *OPENARCH*, 2002.

[20] "Openfog Consortium," [Online]. Available: https://www.openfogconsortium.org.

[21] M. Villari, S. Dustdar, O. Rana and R. Ranjan, "Osmotic Computing: A New Paradigm for Edge/Cloud Integration," *IEEE Cloud Computing,* vol. 3, no. 6, pp. 76-83, 2016.

[22] P. Pace, G. Aloi, . G. Raffaele, G. Caliciuri, G. Fortino and A. Liotta, "An Edge-Based Architecture to Support Efficient Applications for Healthcare Industry 4.0," *IEEE Trans. Industrial Informatics,* vol. 15(1), pp. 481-489, 2019.

[23] X. Sun, "EdgeIoT: Mobile Edge Computing for the Internet of Things," *IEEE Communications Magazine,* no. 12, 2016.

[24] C. Savaglio and G. Fortino, "Autonomic and cognitive architectures for the Internet of Things," in *International Conference on Internet and Distributed Computing Systems*, 2015.

[25] D. Clark, "Making the world (of communications) a different place.ACM SIGCOMM CCR, 35(3):91–96, 2005.," *ACM SIGCOMM,* vol. 35, no. 3, pp. 91-96, 2005.

[26] D. Tracey and C. Sreenan, "OMA LWM2M in a Holistic Architecture for the Internet of Things," in *", IEEE 14th International Conference on Networking, Sensing and Control (ICNSC-2017)*, 2017.