Gateways to Heaven: Observations and Predictions on the Software Architecture of IoT Gateways

Antero Taivalsaari
Nokia Bell Labs
Tampere, Finland
antero.taivalsaari@nokia-bell-labs.com

Tommi Mikkonen
University of Helsinki
Helsinki, Finland
tommi.mikkonen@helsinki.fi

ABSTRACT

The Internet of Things (IoT) enables connected devices that are an integral part of the physical world. The possibility to connect, manage, configure and dynamically reprogram remote devices through local and global cloud environments will open up a broad variety of new use cases, services, applications and device categories, and will enable entirely new product and application ecosystems as well. In this paper we discuss the software architecture options of IoT gateways as a follow-up to our earlier paper that defined a taxonomy of software architectures for IoT devices. We summarize several different software architecture options for IoT gateways. These options have a significant impact on the overall end-to-end architecture and topology of IoT systems, e.g., in determining how much computation can be performed on the edge of the network. Based on our observations and industry experiences we then make predictions on the future of gateway solutions and IoT systems more broadly.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; **Distributed architectures**; • **Software and its engineering → Distributed systems organizing principles**; *Software architectures.*

KEYWORDS

Internet of Things, IoT, IoT Gateways, Programmable World, Software Architecture, Software Platforms, Edge Computing, Isomorphic Software Systems

ACM Reference Format:
Antero Taivalsaari and Tommi Mikkonen. 2019. Gateways to Heaven: Observations and Predictions on the Software Architecture of IoT Gateways. In *17th International Conference on Advances in Mobile Computing & Multimedia (MoMM '19), December 2–4, 2019, Munich, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3365921.3365930

1 INTRODUCTION

The Internet of Things (IoT) represents the next significant step in the evolution of the Internet, bringing us connected devices that are an integral part of the physical world. We believe that this evolution will ultimately result in the creation of a *Programmable World* in which even the simplest and most ordinary everyday things and artifacts in our surroundings are connected to the Internet and can be controlled and programmed remotely [17]. The possibility to connect, manage, configure and dynamically reprogram remote devices through local and global cloud environments will open up a broad variety of new use cases, services, applications and device categories, and will enable entirely new product and application ecosystems as well [8].

The emergence of the Internet of Things is enabled primarily by two factors: (1) advances in hardware development, and (2) advances in connectivity. The low-cost computing chips and development boards that became available in the mid-2010s already match or exceed the memory and processing power capabilities that mobile phones had in the late 1990s, making it possible to embed connectivity and programmability virtually anywhere. Correspondingly, new network technologies such as 5G and Narrowband IoT (NB-IoT) broaden the availability of cellular connectivity to a much larger spectrum of user cases requiring either very low cost, very low battery consumption, low latencies, very high bandwidths, or better coverage.

In our 2017 IEEE Software paper [12], we pointed out that a common, generic end-to-end (E2E) architecture for IoT systems has already emerged. We also summarized the architecture options of IoT client devices in another IEEE Software paper in 2018 [13]. The observations presented in those earlier papers were based on our practical experiences in designing and constructing a number of commercial and R&D IoT systems since 2014.

In this paper we follow up on those previously presented topics, and analyze the software architecture options for IoT *gateways*, i.e., devices that are used for connecting IoT devices to the cloud back-end(s) [3]. While it may not be obvious at the first blush, there exists a wide spectrum of software architecture options for IoT gateways, reflecting the divergent needs of consumer-oriented, commercial and industrial IoT solutions. There are also some interesting trends in network technology evolution – including the aforementioned 5G and Narrowband IoT technologies – that may ultimately eliminate the need for gateways in IoT systems altogether. These options and trends can have a dramatic impact on the overall architecture of an IoT system, e.g., in defining the overall communication topology as well as in determining how much computation can be performed on the edge of the network.

The structure of this paper is as follows. We start the paper by providing some background on IoT system architecture in Section 2, followed by a summary of the basic architecture options for IoT gateways in Section 3. Section 3 additionally discusses the emergence of Cellular IoT radio technologies, setting the stage

MoMM '19, December 2–4, 2019, Munich, Germany

ACM ISBN 978-1-4503-7178-0/19/12…$15.00
https://doi.org/10.1145/3365921.3365930

for some predictions later in the paper. Section 4 discusses the software architecture options for gateways in light of the IoT device architecture options presented in our earlier papers. Section 5 makes some predictions on the future of gateway solutions and IoT systems more broadly based on our industry experiences and observations. Finally, Section 6 concludes the paper.

2 IOT SYSTEMS – THE BIG PICTURE

Given the connected nature of smart things and the presence of a backend service, IoT systems are *end-to-end* (E2E) systems that consist of a number of architectural elements that are typically present in one form or another in all IoT solutions. In our IEEE Software article, we pointed out that a common, generic end-to-end (E2E) architecture for IoT systems has already emerged (see Fig. 1) [12].

As depicted in Fig. 1, IoT systems generally consists of Devices, Gateways, Cloud and Applications. *Devices* are the physical hardware elements that collect sensor data and may perform actuation as well (e.g., trigger an action to turn lights on or control water flow). *Gateways* (also sometimes known as *Hubs*) collect, preprocess and transfer sensor data from devices, and may deliver actuation requests from the cloud to devices. *Cloud* has a number of important roles, including data acquisition, data storage and query support, real-time and/or offline data analytics, device management and processing of actuation requests. *Applications* range from simple web-based data visualization dashboards to highly domain-specific web and mobile apps. Furthermore, some kind of an administrative web or mobile user interface is typically needed.

Historically, IoT systems were very *cloud-centric* in the sense that nearly all the computation was performed in the cloud in a centralized fashion [2]. However, as more computing power and storage becomes available on the edge of the end-to-end system (devices and gateways), the more realistic it will be to perform significant computation also in IoT devices and gateways. We predict that this evolution will ultimately lead the industry towards *isomorphic IoT system architectures* in which the same software packages can run in any computational element of the end-to-end system, regardless of their perceived simplicity or complexity. In such isomorphic systems, software packages may even be dynamically migrated from one computational element to another in order to optimize the tradeoffs between computing capabilities, network latencies, storage capacity and other factors, using various technologies [4]. We will discuss this topic in more detail in Section 5 after examining the architecture options.

3 IOT GATEWAYS – BASIC OPTIONS

Gateway devices have a central yet often overlooked role in IoT systems today. The primary role of gateways is to serve as the *connectivity bridge* between IoT devices and the cloud, allowing the data collected by IoT devices to be uploaded to backend servers, and passing the actuation requests from the applications and the cloud to devices. Basically, since most IoT devices today only support local (near-range) connectivity technologies such as Bluetooth (https://www.bluetooth.com/specifications) or Zigbee (http://www.zigbee.org/download/standards-zigbee-specification/), IoT devices themselves are unable to communicate with the cloud directly. Thus, an intermediary communication solution is required for cloud connectivity.

In addition to handling cloud connectivity and data uploading, gateways may perform *preprocessing of data* and *run analytics algorithms to filter out and preselect most relevant data* before data is uploaded. They may also *generate alerts* when data values exceed certain predefined ranges. In general, since gateways typically have a lot more computing power and other resources than IoT devices they serve, a significant part of the functionality that needs to be carried out in the edge of the end-to-end solution is typically handled by gateways.

Today's gateway solutions can be divided broadly into two categories based on the use of the IoT system: (1) *consumer-oriented* and (2) *professional* solutions. These two categories exhibit different characteristics especially in the connectivity area. In addition, there is a third category, consisting of IoT solutions that operate without a specific gateway, relying on available telecommunications infrastructure.

3.1 Consumer-Oriented IoT

IoT solutions intended for consumers commonly utilize the consumers' smartphones as the gateway solution. For instance, smartwatches or sports watches are usually paired with the user's smartphone, leveraging the smartphone for data uploading, device data updates and firmware upgrades. In addition to (or instead of) smartphone based connectivity, many consumer-oriented IoT systems such as smart home lighting or security systems may also utilize the Wi-Fi network that the consumers already have available in their homes.

Even in those consumer-oriented solutions that utilize Wi-Fi connectivity, or in which there is a dedicated gateway device such as a set-top home box for controlling the user's smart home appliances, the smartphone is still used for complementing the overall end-to-end solution, e.g., to host the mobile apps that are used for controlling and configuring the IoT devices as well as for monitoring the overall state of the system.

3.2 Professional IoT

Commercial and professional IoT solutions tend to have special requirements that necessitate specialized gateway solutions. For instance, in industrial applications (e.g., in malls, warehouses, factories, or mines) there may be a need for tamper-proof, waterproof, dustproof, and/or vibration-resistant devices, or solutions that are embedded in moving equipment such as assembly lines or forklifts, for example.

IoT devices meant for professional use are usually still based on stock hardware, but they are packaged in custom form factors as necessitated by the specific use case, domain and/or physical location. Furthermore, professional IoT gateways utilize a broad spectrum of cloud connectivity technologies ranging from Wi-Fi and cellular to fixed network connectivity.

3.3 IoT Solutions with No Gateways

In addition to the two categories presented above, there is a third, emerging category of IoT systems. As summarized above, the vast

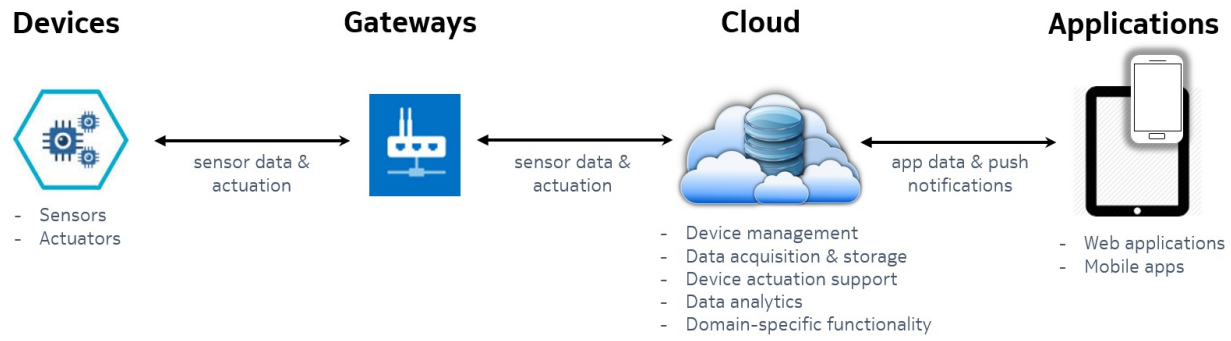

Figure 1: Common generic end-to-end (E2E) IoT system architecture

majority of IoT devices are connected to the Internet indirectly via local connectivity (near-range radio) technologies such as Bluetooth or Wi-Fi. This means that the devices are bound either *to a specific location* (such as a fixed home gateway providing Wi-Fi access) or *to a specific mobile gateway solution* (such as a smartphone serving as a Bluetooth gateway). See Fig. 2 for an illustration.

With the emergence of Cellular IoT radio technologies such as Narrowband-IoT and LTE-M, it will be possible to connect IoT devices directly to the Internet from virtually anywhere where network coverage is available (Fig. 3). Furthermore, such direct connectivity can be accomplished in *friction-free* fashion – in other words, without the usual setup hassles such as Bluetooth pairing or Wi-Fi security passwords that burden the usage of local connectivity technologies.

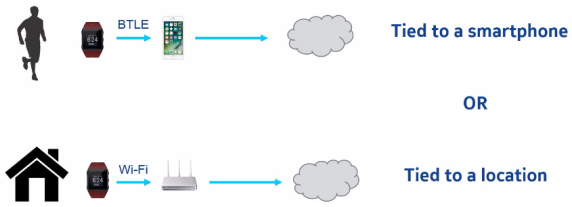

Figure 2: Wearable IoT devices today: cloud connectivity provided indirectly via short-range radio technologies

As a concrete scenario, let us take a look at one of the most popular categories of connected IoT devices today: personal wearable smartwatches or sports watches. Today, the dominant solution for providing cloud (Internet) connectivity is to pair these devices either with a smartphone (and an application therein) or to a Wi-Fi gateway in a certain physical location (Fig. 2). In contrast, Fig. 3 presents another scenario in which such a device is connected directly to the cloud via Narrowband IoT.

Until recently, the widespread emergence of Internet-connected, low-power IoT devices was hindered by the lack of infrastructure that would make it possible to support such devices on a global scale. While there have been standardization efforts around specific low-power wide area network (LPWAN) technologies such as *LoRa* (https://lora-alliance.org/) and *Sigfox* (https://www.sigfox.com/),

Figure 3: Wearable IoT devices in the future: direct cloud connectivity

none of those standards has reached worldwide acceptance in the same scale as cellular connectivity standards established by 3GPP (https://www.3gpp.org/). Cellular IoT technologies standardized by 3GPP are defined as two new categories of LTE, originally included in 3GPP Release 13 in 2016 (https://www.3gpp.org/release-13).

- **Category M1** (also known as CAT-M1, LTE-M or eMTC) is a downscaled version of the original LTE standard (i.e., Release 8 with Categories 1-5) with the data rate reduced to 1 Mbps, while keeping intact the majority of the other features such as full mobility, relatively low latency, and even the ability to support voice calls. The main benefits of CAT-M1 are lower cost, lower power consumption and improved coverage.
- **Category NB1** (also known as NB-IoT, Narrowband IoT or CAT-NB1) deviates more from the baseline LTE standard, while still maintaining deployment compatibility. NB1 provides the lowest-cost, low-power wide area connectivity solution for simple delay-tolerant applications. However, NB1 only provides data rates in the order of 20-60 kbps; furthermore, NB1 has longer latencies and does not support mobility (automatic handovers).

For more details, refer to [7, 9, 16]. What is important is that these new technologies can be taken into use in existing LTE/4G infrastructure *incrementally* – often just an over-the-air software upgrade suffices to enable support for these new technologies. Given that no significant new network infrastructure investments are required by the operators, the adoption curve for Cellular IoT technologies is significantly smoother than for technologies that require significant investments in competing infrastructure.

4 IOT GATEWAYS – SOFTWARE ARCHITECTURE OPTIONS

As can be determined from the discussion above, there are a broad range of software architecture options and stacks for IoT gateways, depending on the expected usage, power budget, and the need to support dynamic programming and/or third-party application development. Computational requirements of gateways are highly dependent on whether gateways are used only for collecting and passing data onto the cloud, or whether gateways are expected to perform significant computation as well, e.g., by running complex analytics libraries and algorithms. Available computing power is also dependent on whether the gateway devices are battery-powered or mains-operated. In certain application domains, e.g., large-scale lighting systems, the *price* of the gateway solution is also a determining factor.

In our earlier paper on IoT device software architecture choices [13], we summarized the basic architecture choices for IoT client devices as follows, ranging from simple to more complex architectures:

(1) *No OS architecture*: for simplest sensing devices that do not need any operating system at all.
(2) *RTOS architecture*: for slightly more capable IoT devices that benefit from a real-time operating system.
(3) *Language runtime architecture*: for relatively simple devices that require dynamic programming capabilities, e.g., in the form of a Python, Java or JavaScript runtime.
(4) *Full OS architecture*: for devices that are capable enough to host a full operating system, but which do not need any on-device user interface. The actual OS is typically some variant of Linux.
(5) *App OS architecture*: for devices that are designed specifically to support third party application development including an on-device user interface. A typical App OS example is the Wear OS (http://wearos.google.com).
(6) *Server OS architecture*: for devices that are capable enough to run a server-side operating system stack (typically Linux + Node.js).
(7) *Container OS architecture*: for high-end devices that are powerful enough to host a virtualized, container-based operating system stack such as *Docker* (http://www.docker.com/) or *CoreOS rkt* (http://coreos.com/rkt/).

At the high level, the same classification applies also to IoT gateways. The main difference is that in gateway solutions there is often a need for a lot more processing power in order to perform sensor data pre-processing and analytics ahead of uploading the data to the cloud. The increased need for computing power and frequent cloud communication means that low-end No OS architectures are seen much less commonly. In contrast, Server OS solutions are more common, since gateways often have to support a large number of IoT devices, e.g., in order to collect measurement data and/or video streams from all devices in a certain room or factory floor. There may also be a need for an on-device web server that can be used for remotely accessing and tuning the behavior of the gateway. Often, communication needs alone rule out the use of the lowest device categories. Furthermore, especially in consumer-oriented use cases, mobile phones are an important gateway category, which is also reflected in the popularity of different configurations.

Table 1 provides a condensed discussion of the software architecture options for IoT gateways in light of our previous articles focusing on the software architecture of IoT devices. However, it should be noted that the architecture options summarized Table 1 are by no means exclusive. For instance, devices based on the language runtime architecture commonly have an RTOS underneath. Likewise, on Full OS platforms, it is obviously possible to run various types of language runtimes and virtual machines as long as an adequate amount of resources are available to host those runtime(s). In general, the more capable the underlying execution environment is, the more feasible it is to run various types of software architectures, platforms and applications on it.

At this point, we do not have any empirical measurements on the popularity or exact percentage split between different types of architecture choices. However, in our own commercial and academic IoT development projects, we have encountered several examples of gateway solutions that are based on real-time operating systems (choice 2 in the list above). We have also built systems that are based on a Full OS (choice 4) or Server OS (choice 5) gateway architecture. In addition, as already mentioned earlier in the paper, consumer-oriented IoT systems typically utilize the user's smartphone as the gateway solution, reflecting the popularity of the App OS approach.

5 DISCUSSION AND FUTURE PREDICTIONS

As already noted at the beginning of the paper, IoT systems were historically very cloud-centric, with the majority of computation taking place in a centralized fashion. However, given the rapidly increasing computing and storage capabilities of IoT devices, it is likely that in the future computation and intelligence will be much more equally distributed and balanced between the cloud, IoT devices and gateways. This is beneficial, since the ability to preprocess data "at the source" can reduce latencies considerably and can also significantly reduce data traffic between the devices and cloud [5, 10].

5.1 Edge Computing

In recent years there has been noticeable interest in *edge computing*, i.e., cloud computing systems that perform a significant part of their data processing at the edge of the network, near the source of the data [6, 11]. In IoT systems supporting edge computing – sometimes referred to as *fog computing* [1] – devices and gateways play a key role in filtering and preprocessing the data, thus reducing the need to upload all the collected data to the cloud for further processing. Together with the emergence of mesh networking and low-power wide area networking (LPWAN) technologies (including the NB-IoT technology mentioned earlier), edge computing can be expected to significantly alter the topologies and the overall software architecture of IoT systems [11]. In the near term, edge computing will result in a significantly more important role for gateways in the overall end-to-end IoT system architecture. In the longer term – assuming widespread adoption of Cellular IoT technologies – edge computing will refer to systems in which a significant portion of computation is performed in the IoT devices themselves.

Architecture option	Device	Gateway
No OS architecture	Simplest sensing devices that do not need any operating system at all	Very optimized use cases, in particular when energy consumption is of utmost concern
RTOS architecture	Slightly more capable IoT devices that benefit from a real-time operating system	Rarely used; system and domain specific use cases exist
Language runtime architecture	Relatively simple devices that require dynamic programming capabilities	No concrete use cases
Full OS architecture	Devices that are capable enough to host a full operating system, but which do not need any on-device user interface	Common option for gateways in numerous contexts, including both commercial and industry systems
App OS architecture	Devices that are designed specifically to support third party application development including an on-device user interface (e.g., Wear OS)	Smartphones that act as a gateway for numerous wearables, in practice usually Android or iOS; some industry applications
Server OS architecture	Devices that are capable enough to run a server-side operating system stack	Common option for both consumer and industry applications
Container OS architecture	Very high-end IoT devices; rarely used in practice (emerging area; foundation for isomorphic IoT systems)	Critical industry applications where modifications are common; rarely used in practice (emerging area)

Table 1: High-level summary of device and gateway architecture options

In practical applications, so far edge computing has not been embraced very broadly in spite of its inclusion on Gartner's Top 10 List of Strategic Technology Trends in 2018[1]. The global edge computing market size was valued at USD 1.47 billion in 2018[2], representing only a fraction of the total USD 80 billion cloud computing market.

While the current low adoption rate may still reflect the lack of application domains in which low latencies are of utmost concern, we believe that this can be attributed also to technical factors. So far, edge computing platforms still have not been as effortless to use as the commercially most successful and dominant cloud platforms such as Amazon Web Services (AWS) and Microsoft Azure. One of the key technical factors is also the apparent diversity of programming models and technologies that permeates the IoT systems today, as discussed in the following.

5.2 Diversity of Programming Models

Today, the vast majority of application developers have been trained to do either mobile development or web development [15]. Many of these developers tend to assume that their skills would be directly applicable to IoT development. However, this is not really true, as IoT systems have many characteristics that do not apply to mobile or web applications at all. IoT developers must consider several factors that are unfamiliar to most mobile and client-side web application developers today. Such factors include:

- multidevice programming;
- heterogeneity and diversity of devices;
- intermittent, potentially unreliable connectivity;
- the distributed, highly dynamic, and potentially migratory nature of software;
- the reactive, always-on nature of the overall system; and
- the general need to write software in a fault-tolerant and defensive manner.

In general, a typical IoT application is *continuous and reactive*. On the basis of observed sensor readings, computations get triggered (and retriggered) and eventually result in various actionable events. The systems are essentially *asynchronous, parallel, and distributed*.

More broadly, IoT devices are bringing back the need for embedded software development skills and education. An interesting additional observation is related to software engineering education. Software development for IoT devices is very similar to "classic" embedded systems development, and is thus bringing back the need for embedded, small memory software development skills [18]. This is a relevant note especially from academic viewpoint, since in the past 10-15 years a lot of universities – at least in Northern Europe – have scaled back their courses on embedded systems development, focusing on presumably more modern and desirable areas such as web and mobile software development instead. A recent Developer Economics survey reports strongly confirms the focus on higher-level programming skills [15]. We have recently studied the educational aspects of IoT development in another paper [14].

One can summarize the diversity of programming models and development platforms as illustrated in Fig. 4 [14]. In brief, IoT devices today use various kinds of software stacks, but IoT device development takes place primarily using embedded software development methods. In contrast, techniques used for gateway development are either those used in the embedded or in the mobile domain (in the latter case utilizing the higher-level tools and APIs of mobile platforms such as Android and iOS). Cloud platforms vary in their functions and configuration, utilizing a diverse range of development technologies and open source components. Finally, applications that are used for interacting with the system are typically mobile or web apps. Each of these areas uses different

[1] https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018/
[2] https://www.grandviewresearch.com/industry-analysis/edge-computing-market

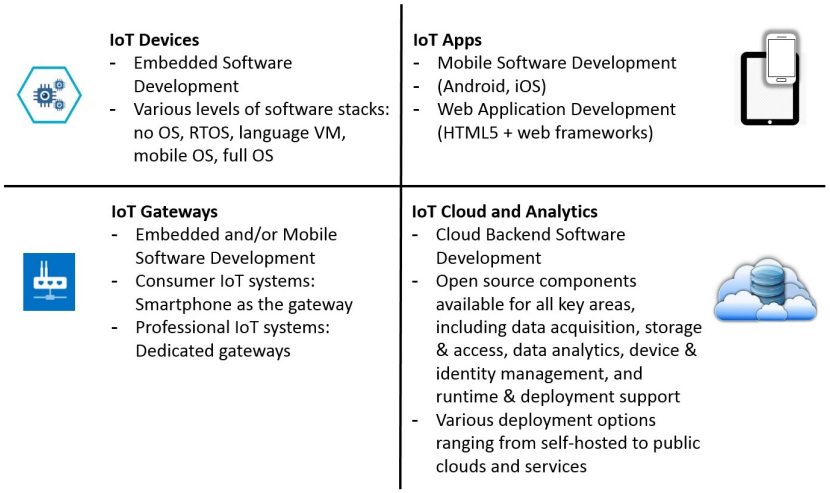

Figure 4: Diversity of Development Approaches in End-to-End IoT Systems

development technologies, and therefore it is not easy to transfer functions from one element in the end-to-end system to another, even if this would be desirable from performance, energy efficiency or latency viewpoint. This will also make the development of the end-to-end system more complex, since the developers need to master a wide variety of different development technologies and methods, including different programming languages and tools.

5.3 Isomorphic IoT System Architectures

It can be argued that the development diversity discussed above has resulted in an exaggerated role of the cloud in the end-to-end system. Given the rapidly increasing computing and storage capacities of IoT devices, we foresee that within the next 5-10 years IoT devices will be able to host considerably more capable software stacks. Ultimately, this may lead us to *isomorphic IoT system architectures* in which the devices, gateways and the cloud will be able to run the same applications and services, allowing flexible migration of code between any element in the overall system. In an isomorphic system architecture, there does not have to be any technical differences between software that runs in the backend or in the edge of the network. Rather, when necessary, software can freely "roam" between the cloud and the edge in a seamless, liquid fashion.

Isomorphic architectures can be seen as the "holy grail" in IoT development. Instead of having to learn many different incompatible ways of software development, in an isomorphic architecture one base technology will suffice and will be able to cover all aspects of E2E development. At this point it is still difficult to predict which technologies will "rule them all", so to speak. Container-based architectures such as *Docker* (https://www.docker.com/) or *CoreOS rkt* (https://coreos.com/rkt/) seem like good guesses at this point, even though their memory and computing power requirements may seem exorbitant from the viewpoint of today's IoT devices. Amazon's *Greengrass* system (https://aws.amazon.com/greengrass/) also points out to a model in which the same development technology can be used both in the cloud and in IoT devices; in Greengrass, the programming platform is Amazon's Lambda.

In the long run, we expect that isomorphic IoT systems will dilute the roles of the cloud and the edge, leading us to "soup computing". Isomorphic systems will allow computations to be transferred dynamically and performed in those elements that provide the optimal performance, storage, network speed, latency and energy-efficiency characteristics, thus enabling the overall behavior of the IoT system to be optimized based on a "soup" of available, diverse computational elements in the overall end-to-end system. Therefore, although fully isomorphic IoT systems are still years away, their arrival may ultimately dilute or even dissolve the boundaries between the cloud and its edge.

6 CONCLUSIONS

According to a popular saying – often attributed to the French writer and journalist Jean-Baptiste Alphonse Karr – the more things change the more they stay the same. This is definitely the case with some recent occurrences in the computing industry. For instance, the technical parallels between today's IoT devices and the early personal computers in the late 1970s and early 1980s are remarkable. There are also interesting technical resemblances between today's IoT devices and mobile phones in the late 1990s and early 2000s.

In this paper we have presented observations on the software architecture options for IoT gateways. We pointed out that there exists a wide spectrum of software architecture options for IoT gateways, reflecting the divergent needs of consumer-oriented, commercial and industrial IoT solutions. We also noted that there are some interesting trends in network technology evolution – including 5G and Narrowband IoT technologies – that may ultimately eliminate the need for gateways in IoT systems altogether. These options and trends can have a dramatic impact on the overall software architecture of an IoT system.

In later parts of the paper, we made some predictions about the role of gateways and broader IoT system architecture in light of

the ongoing transition towards edge computing and less cloud-centric IoT systems. We remarked that the current diversity of programming models, languages, tools and methods used in the cloud and the edge of an IoT system makes it difficult to transfer functions from one element to another, even if this would be desirable from performance, energy efficiency or latency viewpoint. This will also make the development of the end-to-end system more complex, since developers need to master a wide variety of different development technologies and methods. This approach seems unsustainable in the long run. We foresee the industry gradually moving towards isomorphic IoT system architectures, i.e., systems that are based on consistent software development technologies and methods throughout the end-to-end system.

REFERENCES

[1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. ACM, 13–16.

[2] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. 2016. Integration of Cloud Computing and Internet of Things: a Survey. *Future Generation Computer Systems* 56 (2016), 684–700.

[3] Hao Chen, Xueqin Jia, and Heng Li. 2011. A Brief Introduction to IoT Gateway. In *IET International Conference on Communication Technology and Application (ICCTA 2011)*. IET, 610–613.

[4] Andrea Gallidabino, Cesare Pautasso, Tommi Mikkonen, Kari Systä, Jari-Pekka Voutilainen, and Antero Taivalsaari. 2017. Architecting Liquid Software. *J. Web Eng.* 16, 5&6 (2017), 433–470.

[5] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. 2015. Edge-Centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review* 45, 5 (2015), 37–42.

[6] Najmul Hassan, Saira Gillani, Ejaz Ahmed, Ibrar Yaqoob, and Muhammad Imran. 2018. The Role of Edge Computing in Internet of Things. *IEEE Communications Magazine* 56, 11 (2018), 110–115.

[7] Nitin Mangalvedhe, Rapeepat Ratasuk, and Amitava Ghosh. 2016. NB-IoT Deployment Study for Low Power Wide Area Cellular IoT. In *27th Annual IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC'2016)*. IEEE.

[8] Dejan Munjin and Jean-Henry Morin. 2012. Toward Internet of Things Application Markets. In *2012 IEEE International Conference on Green Computing and Communications (GreenCom)*. IEEE, 156–162.

[9] Rapeepat Ratasuk, Benny Vejlgaard, Nitin Mangalvedhe, and Amitava Ghosh. 2016. NB-IoT System for M2M Communication. In *Workshop on Device to Device communications for 5G NETWORKS (WD5G'2016)*. IEEE.

[10] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *IEEE Computer* 50, 1 (2017), 30–39.

[11] Weisong Shi and Schahram Dustdar. 2016. The Promise of Edge Computing. *Computer* 49, 5 (2016), 78–81.

[12] Antero Taivalsaari and Tommi Mikkonen. 2017. Roadmap to the Programmable World: Software Challenges in the IoT Era. *IEEE Software, Jan/Feb 2017* 34, 1 (2017), 72–80.

[13] Antero Taivalsaari and Tommi Mikkonen. 2018. A Taxonomy of IoT Client Architectures. *IEEE Software, May/Jun 2018* 35, 3 (2018), 83–88.

[14] Antero Taivalsaari and Tommi Mikkonen. 2018. On the Development of IoT Systems. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 13–19.

[15] VisionMobile. 2016. Developer Economics State of the Developer Nation, Q1 2016. http://www.visionmobile.com/reports/developer-economics-state-developer-nation-q3-2016. [Online; accessed 18 November 2018].

[16] Y.-P. Eric Wang, Xingqin Lin, Ansuman Adhikary, Asbjörn Grövlen, Yutao Sui, Yufei Blankenship, Johan Bergman, and Hazhir S. Razaghi. 2017. A Primer on 3GPP Narrowband Internet of Things. *IEEE Communications Magazine* 55, 3 (2017), 117–123.

[17] Bill Wasik. 2013. In the Programmable World, All Our Objects Will Act As One. *Wired. Available online: http://www.wired.com/2013/05/internet-of-things-2/ (accessed 22 June 2019)* (2013).

[18] Charles Weir and James Noble. 2000. *Small Memory Software: Patterns for Systems with Limited Memory.* Addison-Wesley (Software Pattern Series).