Feature Article

A Deviceless Edge Computing Approach for Streaming IoT Applications

Marjan Gusev, Bojana Koteska, Magdalena Kostoska, and Boro Jakimovski
Ss. Cyril and Methodius University

Schahram Dustdar, Ognjen Scekic, Thomas Rausch, and Stefan Nastic
TU Wien

Sasko Ristov, and Thomas Fahringer
University of Innsbruck

***Abstract*—In this paper, a deviceless edge computing solution is analysed in contrast to the "traditional" edge server solution. We compare centralized with the distributed deviceless approaches for horizontal offloading of data and computations, and analyze the requirements of protocols to realize such solutions. The proposed deviceless solutions are more energy-efficient (IoT and edge devices may work for a longer period without recharging), provide a scalable and elastic environment and extended fault tolerance.**

■ **CLOUD COMPUTING IS** a paradigm that has transformed the vision of *computing as a utility* into reality.[1] However, even the exascale computational potential of cloud resources cannot be efficiently utilized for applications and data sources that are distributed across geographically distant locations.[2] Transferring large amounts of data to a centralized cloud over WAN networks generates latencies that are higher than the expected response for real-time distributed applications. Apart from the large quantities of data, a further challenge is that the Internet of Things (IoT) devices, such as sensors, actuators, and controllers,[3] generate *fast data*. For example, an ECG sensor streams data with sampling rates higher than 100 samples per second, and each sample is at least 2-byte integer number.[4] The streams of these massively distributed devices can be processed into valuable information. However, a huge amount of this data usually

Digital Object Identifier 10.1109/MIC.2019.2892219
Date of current version 6 March 2019.

remains underused for data analytics applications as they cannot be transferred to the cloud.

The emergence of edge computing allowed moving the processing closer to data sources instead of another way around. Intermediate edge devices take responsibility for processing and data storage instead of cloud servers, which diminishes intensive data transfers, thereby extending the life of IoT devices. Cloud servers can be exploited only for services and additional tasks not provided by the edge devices.[5] However, although the edge is an appropriate paradigm for processing data near its source, a surge in a number of available data sources poses challenges to hosting sufficient processing services on a single edge device, due to its energy, computing, storage, and communication limitations. Edge devices, therefore, need to offload data or specific processing functions (computations) to other edge devices or edge servers.[6]

In this paper, we continue our line of research towards realizing deviceless edge computing paradigm.[6,7] We analyze a streaming IoT device that needs to process data incoming at high rate and volume. In addition, we assume that a mobile edge device is wirelessly connected to the IoT device on one side and to a cloud server on the other side. As such a mobile edge device is equipped with a limited energy source, the challenge is to relieve it from complex processing and data storage duties as much as possible. This paper discusses the architectures of edge computing solutions for streaming IoT, where the edge device needs to horizontally offload data and processing. In particular, we will address the deviceless architectures where the offload is realized among various edge devices.

STATE OF THE ART

Various models of computation for scalable IoT data processing have been proposed in research. In particular, cloud-based data stream processing (DSP) systems have been extended to deal with IoT and the edge. Pu *et al.*[8] developed a system that scales DSP across multiple cloud regions to enable low-latency geo-distributed data analytics. Cardellini *et al.*[9] have tackled operational challenges of DSP over heterogeneous infrastructures, such as the QoS-aware deployment of DSP operators onto decentralized resources and have also discussed this in the context of Fog computing architecture.[10] To deal with the challenges of IoT, Nardelli *et al.*[11] later fused these ideas with the computational paradigm of Osmotic Computing[12] into the Osmotic Flow model. While these approaches provide solid models of computation in the context of IoT and edge computing, they do not address the architectural challenges of scaling such systems to the edge.

Another related field of research, which underpins our approach presented in this paper, is deviceless edge computing paradigm.[7] Deviceless edge computing extends the serverless paradigm to the edge of the network, enabling IoT and edge devices to be seamlessly integrated as application execution infrastructure. Recently, similar approaches attempting to enable function execution (FaaS) at the edge have emerged.[6,13,14] Compared to such approaches, which deal with fundamental aspects of the deviceless paradigm, our approach deals with a particular problem of supporting the IoT streaming applications form the architecture perspective.

USE-CASE SCENARIOS AND RESEARCH CHALLENGES

A wearable ECG sensor acting as a streaming IoT device and smartphone as an edge device is an example of a typical use-case scenario. The sensor, equipped with two ECG electrodes, is worn on the patient's chest. In order to save the battery on the sensor, all data are transmitted via a Bluetooth low energy protocol to the patient's smartphone. Data is received by the smartphone application, analyzed and results presented to the patient. In addition, data are transmitted to the server for permanent storage and extended analysis.

Data analytics functions on the smartphone application include temporary storage, interpretation of data samples, alerting in a case of interrupted monitoring, and identified arrhythmia. The most challenging problems in this use case occur if the smartphone detects insufficient resources, such as the inability to calculate all functions and respond in real-time, low battery level, or small storage capacity. This prompts an *offloading* of certain functions to other computing devices. Either storage or processing capabilities may be offloaded in this scenario.

The identified offloading problem may be resolved by a 1:N mapping, where a certain number of edge devices (smartphones) or edge servers located nearby may take the responsibility to calculate a different function specified in the application.

The next scenario shows an N:1 mapping, where a number of IoT devices (ECG sensors) contact a single edge device (tablet). In case of a disaster, such as an earthquake or a fire, an emergency medical team has to examine numerous patients on the spot, assess in real time the seriousness of injuries and determine priorities for administering medical assistance and transporting patients. A set of sensors are attached to the patients at the disaster location to monitor vital functions. The members of the medical team use a tablet to inspect the processed data and make critical decisions on the spot.

Again, a problem might arise if the tablet is not able to process the streaming data of all connected sensors and needs to offload certain functions to another tablet, or to the edge server at the ambulance vehicle using the same WiFi or LAN connection.

The problems identified in these two use-case scenarios can be summarized to offloading a system function and/or data to another edge device or edge server. The associated research challenges include at least the following:

- *architecture* How to design the data flow in the edge computing solution?
- *availability* How to detect insufficient resources?
- *collaboration* How to contact neighboring edge devices?
- *compatibility* How to find if an edge device is capable of taking over a system function?
- *delegating* How to transfer a certain function to another edge device?

Most of these challenges may be solved by a *scheduling* algorithm in the edge computing architecture, i.e., by finding an edge device and rescheduling a certain system function to it.

Although one may find that a simple round robin principle will provide an uncomplicated and yet efficient solution to this problem, it still remains far from trivial. For example, during the runtime procedure of one edge device taking over a processing function from another edge device, what if the pairing (connection) to the IoT source is not successful, or the function cannot be realized due to the inability of the new edge device to successfully execute the corresponding algorithm? In this paper, we elaborate concept ideas on architecture and protocols needed to establish such a solution.

OVERVIEW OF EDGE COMPUTING ARCHITECTURES

In this section, we first discuss the conventional cloud-based IoT architectures and then present several approaches for realizing deviceless computing solutions. The corresponding protocols and scenarios are designed for use with streaming IoT devices.

Traditional Cloud-Based IoT Solution

A typical cloud-based architecture is based on two layers, the lower layer with end-user devices (IoT devices) and the upper layer with the cloud servers.

Note that the IoT devices need to communicate to the cloud server via local area network (LAN), radio area network (RAN) including 3G/4G/5G mobile network, and wide area network (WAN). The energy consumption needs to be relatively high to enable uninterrupted performance. Thus, these IoT devices are mostly stationary and equipped with a continuous power supply.

In the case of a static IoT sensor that provides and/or consumes data occasionally, such as measuring temperature, or controlling an access to a door, a very small quantity of data is exchanged between the cloud and the IoT device. However, a problem arises in the case of a *streaming IoT* device, such as a sensor that generates large quantities of data at a high rate that need to be transferred to the cloud.

The conventional cloud-based architecture cannot cope with the increased demand for data

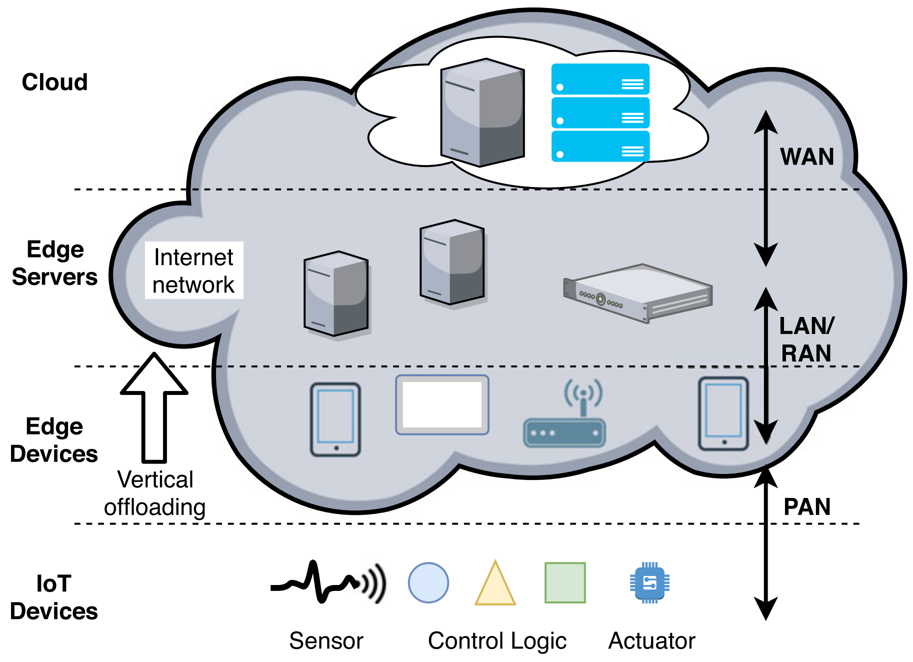

Figure 1. Vertical edge computing architecture for a streaming IoT solution.

transfer. In addition, if the IoT device is a mobile device wirelessly connected, then it does not have a sufficient power supply and this architecture will not be able to support an energy efficient, sustainable solution.

Edge Computing Architecture

A potential solution to the previously analyzed problem is an edge-centric solution. An edge computing architecture is a cloud-based computing architecture, where the computing and data storage are located closer to the user/data source.

Practically, another intermediate layer (edge layer) is introduced between the cloud and IoT layers. The IoT device offloads data to an edge device (in the upper layer) in order to cope with the storage and processing requirements. Therefore, the IoT device does not perform any computations and saves energy for extended performance. The edge device performs the computations and returns results to the IoT device. Although most of the services are performed by the edge device and results are transferred back to the IoT device, the edge device still needs to transfer data to the cloud server for extended processing and exchange in a collaborative environment.

Extended Edge Computing Architecture for Streaming IoT

The previously presented architectural solution cannot cope with the situation when the edge device cannot perform the required services, e.g., in the case of mobile edge devices with limited resources and wireless connectivity. In order to preserve energy, the edge device will want to distribute the processing task to other devices. The first architectural solution that supports such offloading is the *vertical edge computing architecture* presented in Figure 1.

A more powerful edge server is located above the edge device and takes responsibility for performing the bulk data storage and complex operations so that the edge device is relieved and performs only (basic) essential processing services and acting mostly as a data transfer hub between the IoT devices and the edge server. The edge device communicates with the IoT devices using Bluetooth, ZigBee, ultrasound, or similar PAN communication technology. On the other side, it communicates with the edge server via WiFi, 3G/4G, or other LAN/RAN technology. This enables the IoT device to spend only a small portion of its stored energy to transfer data to a nearby edge device, which is taking care to transfer data to a more distant edge server.

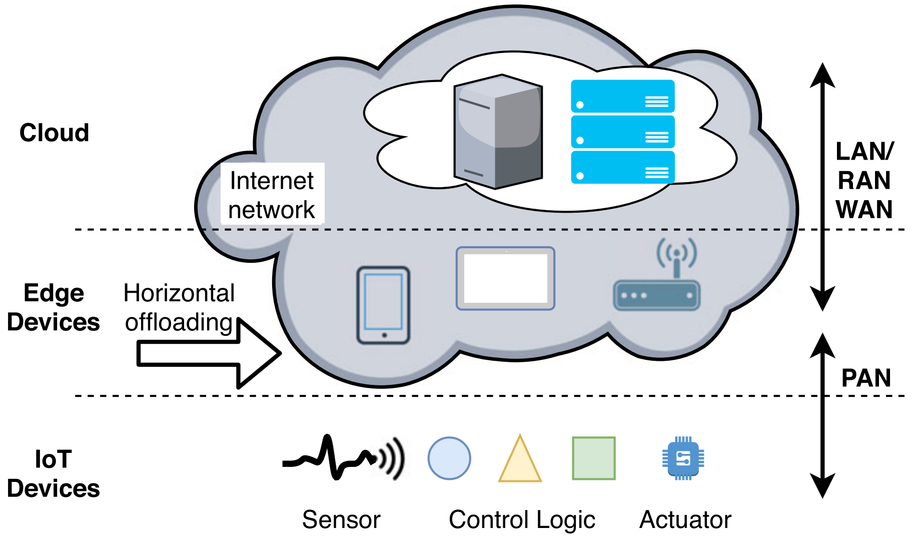

Figure 2. Horizontal edge computing (serverless) architecture of a streaming IoT solution.

A more promising solution is based on our *deviceless edge architecture*.[6,7] This architecture does not require explicit management of edge devices and/or servers, and offloading is realized horizontally to nearby edge devices (Figure 2).

MODELING A DEVICELESS SOLUTIONS FOR STREAMING IOT APPLICATIONS

In this section, we describe and discuss the communication and horizontal offloading models for IoT and edge devices in a deviceless edge computing architecture.

Functional Model

An abstract model of the IoT device includes just the streaming data output at a specific sampling frequency (that determines the data velocity) where each data sample has a predefined bit resolution (that defines the streaming data volume). One can assume that this fits in the Big Data definition, assuming that the other V properties (Value, Variety, and Veracity) are not relevant for this abstract model. Actually, we consider only fast-moving data, where all data items are of the same type (no variety), and trustworthiness to data (veracity) is high, assuming that the noise is eliminated. Value property refers to a high ability to extract information out of data.

An edge device abstract model includes connectivity to an IoT device and transferability to another edge device, both with certain data velocities and volumes. The ability of an edge device to receive and send data as input and output depends on the network delay and throughput of the corresponding PAN or LAN networks.

Coordination Protocols for Offloading and Scheduling

In the deviceless paradigm, one of the requirements is that the edge device needs to communicate with other edge devices and decide whether to offload data and computations to them. This was not required in the vertical edge computing solution since the transfer is directly realized from the edge device to the edge server.

Every edge device can play different roles in this model: *gateway*, *processor*, or both. A *gateway* terminates one or more IoT devices and plays the role of data acquisition and retransmission. The *processors* represent nodes that are capable of data processing. Both roles are subject to horizontal offloading but can be managed by integrated or separate scheduling processes.

A number of protocols/algorithms are critical to the establishment and functioning of the deviceless edge solution, including the following:

- *Discovery* – used to establish the distributed system of edge devices (eg., mDNS,[15] uPNP[16]);

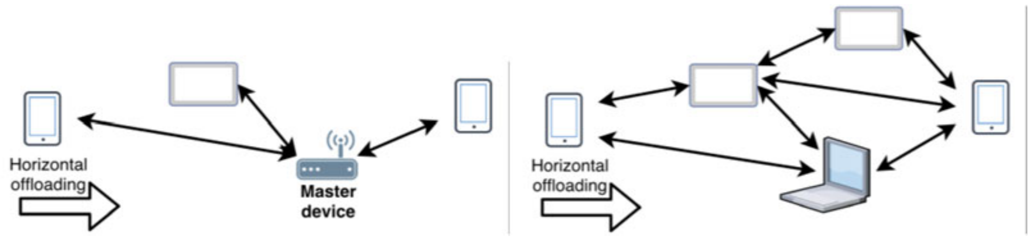

Figure 3. Centralized (left part) and distributed (right part) horizontal edge computing (serverless) architecture for a streaming IoT scenario.

- *Election* – used to select the master scheduler device among all edge devices in a local network (e.g.,[17]);
- *Coverage* – used to gather visibility and compatibility information across edge devices and sensors (e.g., based on LEACH,[18] CCP[19]);
- *Scheduling* – used by devices for scheduling and rescheduling operations, functions, and responsibilities among edge devices, in order to optimally distribute the workload (similar to MLBS, STG, VSG, ILR[20]).

The scheduling task involves many subtasks, where specific protocols can be used in the horizontal, deviceless edge computing solution, including:

- *Compatibility check*, used by an edge device to communicate with the neighbors to exchange property information and check if they are compatible to take over a specific or complete functionality from the asking edge device;
- *Handover*, when an edge device (*gateway*) starts negotiation with another edge device to take over operations;
- *Migration*, when an edge device (*processor*) delegates another edge device to perform a specific operation.

There are three approaches for the realization of horizontal offloading used in such a deviceless scenario, which are based on P2P approaches: *decentralized*, *distributed*, and *hybrid*.

The decentralized solution is based on the election of a *master scheduler device* that monitors the performance of edge devices, as presented in the left part of Figure 3. One edge device is assumed to be the master (coordinator) and takes the role and responsibility of coordinating coverage and scheduling tasks in the local network.

The distributed solution is also a kind of an unstructured P2P overlay network, as presented in the right part of Figure 3, where there is no master scheduler device, and nodes distribute the coverage information between each other in order to consensually coordinate and implement the scheduling task.

Both the centralized and decentralized solutions have pros and cons. For example, the master node in a centralized solution has the full view of the edge network and can thus employ the most efficient scheduling algorithm. The decentralized solution, on the other hand, provides high scalability. The pros of both approaches can be achieved by a hybrid solution.

Instead of having a single master node, the nodes will choose an arbitrary number of master nodes depending on the total number of edge devices. Master nodes are connected to each other through a high-speed LAN and each of them will be responsible for a group of edge devices. In this architecture, in case a considerable number of devices under one master node require offloading, the supervising master node can contact other master nodes for a list of edge devices capable of accepting part of the load.

Advantages

Our deviceless solution for streaming IoT applications offers a number of advantages over the "traditional" architectures, including:

- *Energy efficiency* – Energy efficiency is achieved on multiple levels. On the physical architecture level, IoT devices consume smaller amounts of energy and can perform longer without recharging, since PAN networking requires less power. Edge devices are also more energy efficient and can perform longer without recharging, since they principally act as a hub between the IoT devices and cloud servers.

- *Scalability and elasticity* – The edge computing solutions can scale easily. Realizing the negotiation protocol and assigning tasks to other edge devices is the key prerequisite to achieving elastically scalable deviceless systems. For example, this can be realized dynamically by using the proposed solutions.
- *Increased fault tolerance* – Once the master scheduler device finds an edge device that does not perform the required functions, it can delegate these functions to another edge device. However, this is also a trait of the distributed approach.

CONCLUSION

In this paper, we described several typical use cases of employing IoT devices in streaming scenarios and discussed different architectural approaches to tackling the associated challenges. We have analyzed two edge computing architectural approaches: 1) our previously introduced deviceless approach, where all edge devices collaborate on the same architectural level in order to provide extended functionalities; and 2) a four-layer edge server approach, where the edge servers are located one architectural level above the edge devices as intermediate computing devices to reach the cloud server.

The main differences are found in data and computation offloading. In our deviceless solution, the offloading is horizontal—to other edge devices in the same architectural level. In the edge server solution, the offloading is always vertical—to a more powerful server (or eventually the cloud) at a higher architectural level.

Device abstraction modeling was introduced and properties were specified both for the IoT and edge devices. We discussed how the deviceless solution can offer several advantages, such as energy efficiency, scalability, and elasticity and increased fault tolerance.

The introduced model will be used as a motivation for future activities, such as defining a more detailed specification of the presented protocols, in order to build a simulation model to verify the expected performance advantages, as well as to further advance our concept of deviceless edge paradigm.

ACKNOWLEDGMENT

This work was supported by the bilateral MKD-AUT project 18779 (Scalability and Elasticity Performance of Cloud Services), which was financed equally by the Austrian Federal Ministry of Science, Research, and Economy and by the Ministry of Education and Science of the Republic of Macedonia.

REFERENCES

1. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility,” *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009. [Online]. Available at: http://www.sciencedirect.com/science/article/pii/S0167739X08001957
2. B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” *Future Gener. Comput. Syst.*, vol. 79, Part 3, pp. 849–861, 2018.
3. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (IoT): A vision, architectural elements, and future directions,” *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
4. M. Gusev and S. Dustdar, “Going back to the roots: The evolution of edge computing, an IoT perspective,” *IEEE Internet Comput.*, vol. 22, no. 2, pp. 5–15, Mar./Apr. 2018.
5. S. Nastic, H.-L. Truong, and S. Dustdar, “SDG-Pro: A programming framework for software-defined IoT cloud gateways,” *J. Internet Serv. Appl.*, vol. 6, no. 1, pp. 1–17, 2015.
6. S. Nastic *et al.*, “A serverless real-time data analytics platform for edge computing,” *IEEE Internet Comput.*, vol. 21, no. 4, pp. 64–71, 2017.
7. A. Glikson, S. Nastic, and S. Dustdar, “Deviceless edge computing: Extending serverless computing to the edge of the network,” in *Proc. 10th ACM Int. Syst. Storage Conf.*, 2017, p. 28.
8. Q. Pu *et al.*, “Low latency geo-distributed data analytics,” *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 421–434, Aug. 2015.

9. V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, “Optimal operator placement for distributed stream processing applications,” in *Proc. 10th ACM Int. Conf. Distrib. Event-Based Syst.*, 2016, pp. 69–80.
10. V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, “On QoS-aware scheduling of data stream applications over fog computing infrastructures,” in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2015, pp. 271–276.
11. M. Nardelli, S. Nastic, S. Dustdar, M. Villari, and R. Ranjan, “Osmotic flow: Osmotic computing + IoT workflow,” *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 68–75, Mar./Apr. 2017.
12. M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, “Osmotic computing: A new paradigm for edge/cloud integration,” *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 76–83, Nov./Dec. 2016.
13. E. D. Lara, C. S. Gomes, S. Langridge, S. H. Mortazavi, and M. Roodi, “Poster abstract: Hierarchical serverless computing for the mobile edge,” in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2016, pp. 109–110.
14. J. Gascon-Samson, M. Rafiuzzaman, and K. Pattabiraman, “ThingsJS: Towards a flexible and self-adaptable middleware for dynamic and heterogeneous IoT environments,” in *Proc. 4th Workshop Middleware Appl. Internet Things*, 2017, pp. 11–16.
15. S. Cheshire and M. Krochmal, “Multicast DNS,” RFC 6762, Feb. 2013. [Online]. Available at: https://rfc-editor.org/rfc/rfc6762.txt
16. A. Presser, L. Farrell, D. Kemp, and W. Lupton, “UPnP device architecture 1.1,” in UPnP Forum, vol. 22, 2008.
17. N. Malpani, J. L. Welch, and N. Vaidya, “Leader election algorithms for mobile ad hoc networks,” in *Proc. 4th Int. Workshop Discrete Algorithms Methods Mobile Comput. Commun.*, 2000, pp. 96–103.
18. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy- efficient communication protocol for wireless microsensor networks,” in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, p. 10.
19. X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, “Integrated coverage and connectivity configuration in wireless sensor networks,” in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 28–39.
20. Y. Zhao, J. Wu, F. Li, and S. Lu, “On maximizing the life time of wireless sensor networks using virtual backbone scheduling,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1528–1535, Aug. 2012.

Marjan Gusev is a Professor with the University Sts Cyril and Methodius, Skopje, Macedonia. He received the Ph.D. degree from University of Ljubljana, Ljubljana, Slovenia, in 1992. His research interests include Internet of Things, cloud computing, and eHealth solutions. Contact him at marjan.gushev@finki.ukim.mk.

Bojana Koteska is an Assistant Professor with the University Sts Cyril and Methodius (UKIM), Skopje, Macedonia. She received the Ph.D. degree from UKIM in 2018. Her research interests include scientific and cloud computing, and software quality. Contact her at bojana.koteska@finki.ukim.mk.

Magdalena Kostoska is an Assistant Professor with the University Sts Cyril and Methodius (UKIM), Skopje, Macedonia. She received the Ph.D. degree from UKIM in 2014. Her research interests include cloud computing and Internet of Things. Contact her at magdalena.kostoska@finki.ukim.mk.

Boro Jakimovski is an Associate Professor with the University Sts Cyril and Methodius (UKIM), Skopje, Macedonia. He received the Ph.D. degree from UKIM in 2010. His research interests include grid computing, high-performance computing, parallel and distributed processing, and genetic algorithms. Contact him at boro.jakimovski@finki.ukim.mk.

Schahram Dustdar is a Full Professor of Computer Science heading the Distributed Systems Group, TU Wien, Vienna, Austria. His work focuses on Internet technologies. He is an IEEE Fellow, a member of the Academia Europaea, and an ACM Distinguished Scientist. Contact him at dustdar@dsg.tuwien.ac.at.

Ognjen Scekic is a Postdoctoral University Assistant at the Distributed Systems Group, TU Wien, Vienna, Austria. His research interests include social computing, cloud computing, and smart cities. He received the Ph.D. degree from the TU Wien, in 2016. Contact him at oscekic@dsg.tuwien.ac.at.

Thomas Rausch is currently working toward the Ph.D. degree at the Distributed Systems Group, TU Wien, Vienna, Austria. His research interests include Internet of Things, edge computing, and event-based systems. He received the master’s degree from TU Wien in 2016. Contact him at trausch@dsg.tuwien.ac.at.

Stefan Nastic is a Postdoctoral Research Assistant with the Distributed Systems Group (DSG), TU Wien, Vienna, Austria. His research interests include IoT, edge computing, cloud computing, big data analytics, and smart cities. He received the Ph.D. degree from TU Wien, in 2016. Contact him at snastic@infosys.tuwien.ac.at.

Sasko Ristov is a Postdoctoral University Assistant with the University of Innsbruck, Innsbruck, Austria, and an Assistant Professor with the University Sts Cyril and Methodius (UKIM), Skopje, Macedonia (leave of absence). His research interests include performance modeling, optimization, scheduling, and resource management in distributed and parallel systems. He received the Ph.D. degree from UKIM in 2012. Contact him at sashko@dps.uibk.ac.at.

Thomas Fahringer is a Full Professor of Computer Science at the University of Innsbruck, Innsbruck, Austria. His research interests include software architectures, programming paradigms, compiler technology, performance analysis, and prediction for parallel and distributed systems. Contact him at tf@dps.uibk.ac.at.