Modeling perspective on edge computing

HPS

Sebastian B. Mohr[*]

∗ Corresponding Author: Sebastian Bernd Mohr (sebastian.mohr@stud.uni-goettingen.de)

Abstract

Edge computing has emerged as a response to the increasing growth of data producers and consumers, offering a promising solution to the challenges posed by the sheer volume of data that needs to be processed in real-time or most efficiently. However, the distributed nature of the infrastructure, heterogeneity of devices and variability of the network make it difficult to manage and optimize resources. In this context, modeling becomes a key component in edge computing to facilitate the allocation of tasks and resources. This work presents an overview of the different modeling aspects of edge computing, including task modeling, resource modeling, communication modeling, and resource orchestration. Overall, this work aims to provide insights into the modeling aspect of edge computing and its significance in optimizing the performance of edge systems.

Contents

1 Introduction

In today's digital age, data has become a crucial asset for many organizations and even business models. Collecting, processing, and analysis of data are essential drivers in business decisions. As a result, cloud computing has become increasingly popular[1] as a means of managing and processing large volumes of data. However, as the volume of data grows, processing and analyzing it in real-time becomes more challenging too. Real-time applications where a single machine or device might not be able to compute a task in a reasonable time or the device might not be powerful enough are especially challenging. For instance in cloud computing, the latency might be too high. In such cases, an alternative solution is needed to offload the computation or to schedule it on additional hardware with lower response times. This is where edge computing comes into play, as it enables data processing and analysis to be performed closer to the source of the data, rather than in a centralized cloud infrastructure.

Edge computing is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed and used. For instance, when running a web application there might be responsive elements that fetch data from a server. This might happen in real time based on user interactions, such as searching for products or updating a shopping cart. Instead of sending all responses to a centralized cloud the closest server is used. In this case, edge computing can be used to improve the user experience by reducing latency and improving the responsiveness of the web application. Further, this paradigm might be applied in the manufacturing industry. Thereof, edge computing can be used to improve operational efficiency and reduce downtime by enabling real-time monitoring and control of equipment and processes without the need for a central data processing facility. By leveraging edge computing, organizations can make more informed and timely decisions based on real-time data, leading to improved efficiency, reduced downtime, and ultimately, increased profitability.

As edge computing is a term coined mainly by the industry there are multiple definitions and interpretations of its implementation. One of the key aspects of edge computing is the orchestration of computational tasks across the available hardware i.e. multiple edge devices and the cloud or servers. However, there is no universal agreement on the terminology used in edge computing, and different sources use a range of technical jargon, often inspired by cloud computing. Terms such as mist, fog, and cloud are used to define different parts of the proposed infrastructure [2]. Despite the lack of uniformity in the terminology, the underlying principles of edge computing remain the similar.

While edge computing offers many benefits, there are also several challenges associated with its implementation [3]. One of the main challenges is ensuring data security and privacy. With edge computing, data is processed and stored across a distributed network of devices, which can increase the risk of security breaches and data leaks [4]. Another challenge associated with edge computing is device interoperability. As edge computing involves heterogeneous devices across a network, these devices must be able to communicate and share data effectively. This requires standardization of protocols and interfaces, which can be a complex and time-consuming process [5]. Finally, resource allocation is another critical consideration when deploying

[1]As of 2019 enterprise spending on cloud infrastructure has overtaken Data Center Hardware & Software spending [1]

an edge computing infrastructure. Edge devices typically have limited processing power and storage capacity, so it is important to allocate these resources effectively to ensure optimal performance [6].

In the following, we will focus on the orchestration aspect of edge computing, with a specific emphasis on lower power devices. In detail, we will look into the trade-off between communication and compute latency and power consumption [7]. In other words, how can we optimize the distribution of computing tasks across the available hardware with subject to the optimal latency and power consumption while still ensuring efficient use of computational resources? To address this, we will explore different modeling approaches that help us to understand the trade-offs involved in edge computing orchestration.

Generally, we identify four parts of edge computing. These include computational tasks, computing devices, communication, and orchestration. In the following, we want to establish each part more rigorously.

2 Compute tasks

The computational task refers to the operation that has to be performed by the edge infrastructure[2]. A task can come in a variety of forms, for instance machine learning or real time data processing. For some tasks it might be beneficial to perform them perform them on the edge infrastructure, while for others it might be more suitable to perform them on a centralized cloud server. The properties of the task determine the type of compute that is used for it in the network as not every task has the same requirements. For instance a task that displays a result to a user can't be offloaded, as it has to be done on the current device. On the other hand, a background task that performs data analysis could be offloaded to a another device.

Not only the type of the task, the size and complexity of it can impact the choice of the compute device. Tasks might require high computational power, such as those involving large data-sets or complex algorithms. Thus these tasks may require higher-power devices such as servers. In contrast, less demanding tasks can be performed on lower-power devices.

Let's consider a task A. In general, a task might be described by its properties and requirements. For simplicity consider that a task might depend on its input data size l, computational intensity x and an arbitrary deadline τ, thus forming the task $A(l, \tau, x)$. To put this into different words, we might collect some data of size l perform some computation on this data with intensity x and want the computation to finish at least in τ seconds. These properties are independent of the computing device i.e. they will stay the same if computing on a high performance server or low-power devices. The deadline τ is formulated as hard deadline but might also be formulated as a soft deadline, where the task can still be executed after the deadline, but with a penalty on the overall performance.

It is important to note that this task definition is not fixed. Tasks may have additional properties and requirements. For example, a task might require certain levels of accuracy or precision, or it might have constraints on the amount of energy or memory that can be used during its execution. Moreover, a task might have dependencies on other tasks or data sources, which can impact its execution and performance.

[2]Edge infrastructure refers to the full stack of connected devices including lower power edge devices and higher power servers and their respective connectivity.

Further, the computational intensity of a task may not always be known beforehand. This is particular true for machine learning tasks. Machine learning tasks often involve training or tuning models, which can require significant computational resources and time. Moreover, the computational intensity of these tasks can vary depending on the complexity of the model, size of the data set, and the optimization algorithm being used.

In such cases, it is important to monitor the progress and performance of the task during its execution, and adjust the computational resources allocated to it accordingly. This can involve dynamically scaling the compute resources, such as by adding or removing devices from the edge infrastructure, or optimizing the algorithms or parameters being used.

Especially for heavy tasks, it might be beneficial to split them into smaller sub-tasks. This can be done using a Directed Acyclic Graph (DAG), which allows to represents the dependencies between the sub-tasks. These sub-tasks can be used similarly to the previous defined tasks but might depend on results of different sub-tasks, thus the between task dependencies have to be kept in mind. For instance if we can split the task A into four sub-tasks a, b, c, d with dependencies $A = a \circ b \circ (c, d)$. Here we can compute c and d independently but need the results to compute b and a respectively. Although, tasks can be split into smaller sub-tasks, it is important to keep in mind the associated overhead cost. A high number of smaller tasks might impede the overall performance of the system due to the added communication and synchronization overhead.

In summary, tasks in edge computing networks can come in different forms, and their properties and requirements determine the type of compute device that is suitable for their execution. Factors such as data size, computational intensity, and deadline can impact the choice of compute device. Additionally, tasks may have dependencies on other tasks or data sources, and their computational intensity may not always be known beforehand. In such cases, it is important to monitor the progress and performance of the task during its execution and adjust the compute resources allocated to it accordingly. Furthermore, heavy tasks can be split into smaller sub-tasks using a DAG, but it is important to consider the associated overhead cost. Overall, understanding the properties and requirements of tasks is crucial for efficient execution in an edge infrastructure.

3 Compute devices

Compute devices or simply devices in the edge computing paradigm refer to the hardware components that are responsible for performing the computational tasks. These devices can range from low-power devices, e.g. micro-controller to high-performance servers located in data centers. Depending on the distance of these devices concerning a data producer or the highest power device in the infrastructure (typically a server), people tend to group them into categories. For instance, devices that are close to the data producers/source tend to be called edge devices, synonymous with devices close to the edge of the network.

Edge devices are often characterized by their limited processing power, memory, and energy resources. Examples of such devices include sensors, smartphones, IoT devices, and microcontrollers. Due to their limited resources, edge devices may not be able to perform computationally intensive tasks, and may rely on more powerful devices, such as servers or cloud infrastructure. However, edge devices can still perform some local processing, such as data filtering or preprocessing, before sending the data to a more powerful device for further processing. On the other hand, more powerful devices, such as servers are typically located in data centers and are characterized by their high computational power, memory, and storage capacity. These servers can handle complex and computationally intensive tasks, such as machine learning and big data analytics. However, the latency involved in transmitting data to and from cloud servers can be a major bottleneck in certain applications, such as real-time data processing and control systems. Thus some people tend to deploy local computing infrastructure, which is closer to the edge devices, to reduce latency and improve response time [3].

Intermediately, if thinking about executing a task A on a specific device d, the question of how long it takes for a given task to be finished on the device or how much power it takes to run this task on this device arises. Let's consider a device d, very simplified we can think about a device in terms of its CPU frequency f_d, normally given in Hz. This tells us how many CPU cycles are completed in a second. Using this simplification, we are now able to compute the time it takes to complete a task on a given device t_d.

$$t_d = \frac{lx}{f_d} = \frac{\omega}{f_d} \tag{1}$$

Here l is the input data size of the task and x is the computational intensity of the task. Additionally, the measure ω is a derived task metric and can be thought of as the required number of CPU cycles to complete the task.

This simplification might hold for some devices but generally, the CPU frequency of a device might not be static or might change depending on the load of the device or the general power consumption policy. Thus, more considerations are needed to get a more accurate measure of the execution time of a task but this is an initial first-order approximation. In a practical setting, one might want to create a more detailed model for each device depending on the desired accuracy.

As edge computing is mostly interested in low-power and also mobile and/or wireless devices, the power consumption of each task for each device is also of interest. E.g. a mobile device might not be able to compute a task because of the limited battery capacity or because of thermal limitations. Again we use a simplification to determine the power consumption, for low-power devices we disregard contributions to the energy consumption such as the memory and network interfaces. We only consider the CPU contribution as it is the dominant factor.

It can be shown using circuit theory and experimental validation, that the power consumption of a CPU is roughly proportional to the frequency squared [8]. Introducing a device-specific constant κ (Js^2) the energy required for computation is then given by

$$E_d = \kappa l x f_d^2 = \kappa \omega f_d^2. \tag{2}$$

The constant κ can be acquired using experiments on the specific device. In contrast, as servers are normally multi-core devices, this simple power consumption model has to be adjusted to incorporate multiple running

[3] This intermediate infrastructure part between edge devices and the data center/cloud is sometimes called fog.

tasks. For instance, we can just choose to sum all contributions.

$$E_s = \sum_{i \leq k} \kappa w_i f_{s,i}^2 \tag{3}$$

Whereby, $f_{s,i}$ is the CPU frequency allocated for the task i and w_i the required number of CPU cycles to complete the task i. This might not be very realistic for all kinds of devices. In reality, the power consumption of a device depends on various factors, including the type of processor, the workload, and the operating conditions such as temperature and voltage. Therefore, a more comprehensive model of power consumption would need to take into account these factors and their interactions. Especially for a server, which can still consume up to 70% [9] of its maximum energy consumption when idle, an alternative model is needed. Using a utilization-based approach to model the power consumption of a server might be more realistic. Energy consumption is roughly linear to CPU utilization ratio [10]. Given a server maximum energy consumption $E_{\max}$ and a fraction α for the idle energy consumption, we can define an alternate energy consumption model.

$$\bar{E}_s = \alpha E_{max} + (1 - \alpha) E_{max} u \tag{4}$$

This does not directly make use of our previously defined tasks, but we might be able to define the utilization of the server u as a sum of all currently running tasks. For instance, assuming each task A_i increases the utilization by Δu_i we can compute the increase of energy consumption of the server E_s.

$$E_s = (1 - \alpha) E_{max} \Delta u_i \tag{5}$$

$$1 \geq u + \Delta u_i \tag{6}$$

Again this is highly simplified. For example, the efficiency of the server's power supply and cooling system can also affect energy consumption at different utilization levels.

Concluding, to determine the execution time and power consumption of a task on a given device one can use models based on CPU frequency, computational intensity, and energy consumption per CPU cycle. In practice, more detailed models may be necessary if more accurate estimations of execution time and power consumption are required. Depending on the device these models may include other metrics such as memory access latency, cache size, disk access speed, the network controller, and more. Moreover, software-level optimizations can also greatly impact the performance and energy efficiency of a device.

4 Communication between devices

In addition to execution time and power consumption of devices the network used in communication or offloading tasks also plays a crucial role in overall performance and energy consumption. Communication between devices can be achieved using different types of protocols, such as cellular networks, Wi-Fi, Bluetooth, Zigbee, etc. Each of these protocols has its characteristics, such as data rate, latency, reliability, and energy consumption (see [11]).

Generally, communication is a crucial aspect to consider in edge computing, as it also impacts the performance and energy consumption of the system. Especially for evaluating if a task should be offloaded to a more powerful device, it is important to consider the impact of transmitting to the tasks compute time and energy consumption. For instance, a single task allocating most of the network's data rate can bottleneck other currently running tasks.

The energy impact and time delay of communication can be significant, especially when dealing with large data amounts or tasks requiring low latency. For instance, in scenarios where robots in a factory need to react in real-time to human interactions, delays in communication could potentially result in dangerous or even deadly situations.

Therefore, it is essential to carefully consider the communication infrastructure and protocol used for communication, to ensure optimal performance and energy efficiency of the system. This includes factors such as data compression, message prioritization, and error-handling mechanisms.

It's worth noting that the type of communication used can also affect the determinism of the system. Wired networks, such as Ethernet or fiber-optic cables, typically provide deterministic and reliable communication with low latency and negligible packet loss. On the other hand, wireless networks, such as Wi-Fi, cellular networks, or satellite, are inherently stochastic in nature due to interference, noise, and varying signal strengths. This stochastic behavior can lead to packet loss, delay, and higher energy consumption due to the re-transmission of lost packets. Additionally, wireless technologies normally have significantly higher power requirements than their wired counterparts. On the other hand, wireless networks offer more flexibility, are generally less expensive in the acquisition, and require less maintenance. Therefore, the choice of network for communication between devices should be carefully considered, taking into account the desired level of determinism, energy efficiency, and reliability of the system.

To simplify the communication between devices, let's interpret the communication between devices as a simple data transfer between two endpoints, abstracting away the details of the underlying network and protocols. Additionally, latency is disregarded as the transfer times are normally significantly bigger. A connection between two endpoints can be established with a mean bandwidth b. The time to communicate a task t_c with data size l is then given by

$$t_c = \frac{l}{b} \tag{7}$$

Overall, this abstraction might hold well for wired communication but is too simplified and inaccurate for wireless communication. This is due to the stochastic nature of wireless networks. Typical problems can include but are not limited to atmospheric ducting, reflection, and refraction from scattering objects in the environment. Additionally, other broadcast signals can lead to interference. This might introduce unpredictable delays and packet loss and cannot be captured by a simple bit pipe model. Therefore, more sophisticated models and protocols are required to predict communication speeds in wireless networks [12].

In addition to effective task execution or timings of the network, the choice of communication infrastructure might also impact power consumption. Wireless networks generally consume more energy than wired networks due to the need for radio frequency transmission and reception, which can be energy-intensive. In addition, wireless networks may require more frequent re-transmissions of lost packets, which further increases energy consumption. To reduce the consumption of the communication, the data size can be decreased through compression or low-power communication technologies with the trade-off of range or bandwidth.

Approximating, the energy impact of communication is among other dependent on the transferred amount of data, the bandwidth, the type of communication, the number of routers on the path, and their utilisation [13, 14]. One common approach to approximating communication energy consumption is to use mathematical models based on the physical characteristics of the communication channel, such as signal propagation, attenuation, and noise [15]. For instance, the Friis transmission equation might be used to approximate wireless transmissions [16]. Other approaches involve using empirical measurements or simulations to estimate the energy consumption of different communication protocols [17, 18, 19]. Taking into account factors such as packet size, transmission rate, modulation scheme, error correction, and network topology.

Security is another critical aspect to consider when it comes to communication between devices in edge computing. With the increasing number of devices and the complexity of the network, the potential attack surface also increases. In addition, the distributed nature of edge computing means that data is often transmitted over untrusted and heterogeneous networks, which makes it vulnerable to interception, tampering, and other types of attacks. To ensure the security of communication in edge computing, it is essential to implement robust security measures such as encryption, authentication, access control, and intrusion detection. Furthermore, it is important to stay up to date with the latest security threats and vulnerabilities and apply security patches and updates on time.

In summary, communication in edge computing should factor data rate, latency, reliability, and energy consumption of the selected communication protocol. Communication can significantly impact the performance and energy consumption of the system, especially for tasks requiring low latency or dealing with large amounts of data. The choice of a wired or wireless network also affects the determinism, energy efficiency, and reliability of the system. In addition, communication infrastructure can impact power consumption, and sophisticated models and protocols are required to predict communication speeds in wireless networks. Security is also a critical aspect that must be considered, and robust security measures must be implemented to ensure the security of communication in edge computing.

5 Task and resource orchestration

Using the previously defined tasks, compute resources, and communication we are now able to design specific algorithms to allocate tasks to specific devices. This is called resource orchestration. Here resource refers to the process of allocating and scheduling computational tasks across the hardware and infrastructure while considering the constraints and requirements of each task and device. This is a critical component of edge computing, as it enables efficient use of computational resources and ensures that tasks are executed in a timely and/or effective manner.

Resource orchestration is essential in edge computing environments due to the distributed nature of the infrastructure, the heterogeneity of devices, and the variability of the network. Resource orchestration algorithms aim to minimize a variety of metrics depending on the specific use case. For instance, in a real-time video analytics application, the objective could be to minimize the latency of the system, while in a smart home automation system, the goal could be to minimize the energy consumption of the devices.

Metrics might include but are not limited to latency, energy efficiency, throughput, bandwidth usage, resource utilization, reliability, and security. But most predominantly and also considered earlier are latency and energy consumption.

Latency is especially important for time-critical applications, such as real-time monitoring, control systems, and autonomous vehicles, where even a small delay can have serious consequences. Energy consumption, on the other hand, is crucial for battery-powered or energy-sensitive devices, such as IoT sensors, and wearable, and mobile devices, where minimizing energy usage is critical to extending battery life and reducing operational costs.

Apart from these two metrics, other factors like throughput, bandwidth usage, reliability, and security can also be important in certain applications. For instance, high throughput can be important for applications that handle a large number of concurrent users or high-volume data streams, while bandwidth usage can be a constraint in networks with limited bandwidth or high costs. Reliability and security, on the other hand, can be critical for applications that handle sensitive data or require high levels of privacy and protection against cyber-attacks.

Overall, the choice of metrics to optimize for resource orchestration depends on the specific use case and application requirements. Careful consideration of all the relevant factors can help ensure that the edge computing system is optimized for its intended purpose while balancing performance, efficiency, and cost-effectiveness.

Let's consider an energy-constrained example using a single device d, a server s. The device has a limited battery capacity, thus it might be beneficial to offload computations to increase the device's lifetime. For instance, we can introduce a decision boundary to decide if we should offload the task. If the Energy needed to compute a task on the device E_d is greater than the total energy needed to communicate E_c and the idle energy consumption of the device E_d^i we offload the task.

$$E_d > E_c + E_d^i \tag{8}$$

Further, one might introduce additional factors, depending on the application and metrics of interest. As a note, the decision boundary does not have to be linear and might follow more complex functions depending on the specific use case and requirements.

Let's also consider the latency of the task as a factor in the decision-making process. If the latency of offloading the task and waiting for the server to complete it exceeds a certain threshold, it might be more efficient to perform the computation locally, even if it consumes more energy. This can be written as follows

$$E_d > E_c + E_d^i \quad \text{if } t_d > t_s + t_c \tag{9}$$

$$E_d \leq E_c + E_d^i \quad \text{otherwise} \tag{10}$$

where t_d is the time to perform the computation locally, t_s is the time to perform the computation on the server, and t_c is the time to communicate the task data to the server and receive the results back[4].

Additionally, stochastic factors such as varying network conditions, device workload, and energy availability can also affect the decision to offload a task. To account for these stochastic factors, one can use

[4]These contributions might be split but for simplicity let's consider them as a single contribution

the probabilistic models or machine learning algorithms to predict the expected energy consumption or latency, and other performance metrics of different offloading strategies, based on historical data and real measurements [20].

For instance, let's consider a varying wireless connection and a device with variable CPU frequency. For simplicity let's say both of these are independent random variables and we observe a normal distribution for the frequency distribution and the additional energy consumption for the communication. We infer this by performing some test measurements on the proposed infrastructure.

$$f_d \sim \text{Normal}(\mu_d, \sigma_d) \tag{11}$$

$$E_c \sim \text{Normal}(\mu_c, \sigma_c) \tag{12}$$

Given a fixed task A, we can still calculate the time needed to compute the task on the device t_d as in eq. 1. Additionally, we can see the energy as in eq. 2 is now similar to a squared normal distribution. Obtaining closed-form solutions of the decision boundaries as in e.g. eq. 8, for this example, might still be possible but generally speaking, this is rarely the case. Here one might want to use a more general measure to compare different distributions, for instance, the KullbackLeibler divergence or if metric properties are desired the Wasserstein distance.

In conclusion, resource orchestration is a critical component of edge computing, as it enables efficient use of computational resources and ensures that tasks are executed in a timely and/or effective manner. The choice of metrics to optimize for depends on the specific use case and application requirements. Apart from latency and energy consumption, other factors like throughput, bandwidth usage, reliability, and security can also be important. Additionally, stochastic factors such as varying network conditions, device workload, and energy availability can also affect the decision to offload a task and probabilistic models or machine learning algorithms [21] can be used to predict the expected performance metrics of different offloading strategies.

6 Conclusion

The edge paradigm has emerged as a response to the increasing growth of data producers and consumers. Its ability to bring compute closer to the data is a major advantage, particularly for real-time applications. However, the terminology surrounding edge and related concepts can be inconsistent, making it difficult to distinguish between them. In general edge computing is concerned with the processing and analysis of data by offering a promising solution to the challenges posed by the sheer volume of data that needs to be processed in real-time or most efficiently.

In this context, modeling has a crucial role in enabling the effective implementation of edge computing systems or designing them from scratch for a specific use case. By modeling the tasks, resources, and communication infrastructure, we can create algorithms and frameworks that optimize resource orchestration, data processing, and analytics. This allows us to develop efficient, scalable, and adaptive edge computing systems that can respond to the changing demands of data processing.

Furthermore, modeling enables us to study the performance of different edge computing architectures, evaluate their limitations and identify areas of improvement. This helps us to continuously refine and enhance the design of edge computing systems to meet the evolving needs of users.

In conclusion, modeling is a critical component of edge computing, allowing us to design, develop and evaluate systems that can process and analyze data in real time or in the most efficient manner. The ability to model tasks, resources, and communication infrastructure enables us to optimize resource orchestration, develop effective algorithms and frameworks, and design efficient edge computing architectures. As such, modeling will continue to play a vital role in the advancement of edge computing, paving the way for innovative solutions in various fields, including IoT, smart cities, healthcare, and beyond.

References

[1] Statista. Enterprise spending on cloud and data centers by segment from 2009 to 2021 (in billion u.s dollars). [Graph], March 28 2022.

[2] Klervie Toczé and Simin Nadjm-Tehrani. A taxonomy for management and optimization of multiple resources in edge computing. *Wireless Communications and Mobile Computing*, 2018, 2018.

[3] Mohammad S Aslanpour, Adel N Toosi, Claudio Cicconetti, Bahman Javadi, Peter Sbarski, Davide Taibi, Marcos Assuncao, Sukhpal Singh Gill, Raj Gaire, and Schahram Dustdar. Serverless edge computing: vision and challenges. In *2021 Australasian Computer Science Week Multiconference*, pages 1–10, 2021.

[4] Yinhao Xiao, Yizhen Jia, Chunchi Liu, Xiuzhen Cheng, Jiguo Yu, and Weifeng Lv. Edge computing security: State of the art and challenges. *Proceedings of the IEEE*, 107(8):1608–1631, 2019.

[5] Yuan Ai, Mugen Peng, and Kecheng Zhang. Edge computing technologies for internet of things: a primer. *Digital Communications and Networks*, 4(2):77–86, 2018.

[6] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[7] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358, 2017.

[8] Karel De Vogeleer, Gerard Memmi, Pierre Jouvelot, and Fabien Coelho. The energy/frequency convexity rule: Modeling and experimental validation on mobile devices. In *Parallel Processing and Applied Mathematics: 10th International Conference, PPAM 2013, Warsaw, Poland, September 8-11, 2013, Revised Selected Papers, Part I 10*, pages 793–803. Springer, 2014.

[9] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH computer architecture news*, 35(2):13–23, 2007.

[10] Rasoul Rahmani, Irene Moser, and Mohammadmehdi Seyedmahmoudian. A complete model for modular simulation of data centre power load. *arXiv preprint arXiv:1804.00703*, 2018.

[11] Mohamed Cherif and Khaled Ben Letaief. A multiprotocol wireless sensor network for high performance sport applications, 2016. [accessed 24 Mar, 2023].

[12] Yongsen Ma, Gang Zhou, and Shuangquan Wang. Wifi sensing with channel state information: A survey. *ACM Computing Surveys (CSUR)*, 52(3):1–36, 2019.

[13] Yueying Zhang, Hang Long, Yuexing Peng, Kan Zheng, and Wenbo Wang. User-oriented energy- and spectral-efficiency tradeoff for wireless networks. *KSII Transactions on Internet and Information Systems (TIIS)*, 7(2):216–233, 2013.

[14] Loic Guegan and Anne-Cécile Orgerie. Estimating the end-to-end energy consumption of low-bandwidth iot applications for wifi devices. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 287–294. IEEE, 2019.

[15] Yu Xiao, Yong Cui, Petri Savolainen, Matti Siekkinen, An Wang, Liu Yang, Antti Ylä-Jääski, and Sasu Tarkoma. Modeling energy consumption of data transmission over wi-fi. *IEEE Transactions on Mobile Computing*, 13(8):1760–1773, 2014.

[16] Joseph A Shaw. Radiometry and the friis transmission equation. *American journal of physics*, 81(1):33–37, 2013.

[17] Seyed Mahdi Darroudi, Raül Caldera-Sànchez, and Carles Gomez. Bluetooth mesh energy consumption: A model. *Sensors*, 19(5):1238, 2019.

[18] Ashima Gupta and Prasant Mohapatra. Energy consumption and conservation in wifi based phones: A measurement-based study. In *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 122–131. IEEE, 2007.

[19] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, and Johanna Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *2012 IEEE wireless communications and networking conference workshops (WCNCW)*, pages 232–237. IEEE, 2012.

[20] Yuyi Mao, Jun Zhang, SH Song, and Khaled B Letaief. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, 16(9):5994–6009, 2017.

[21] Baris Yamansavascilar, Ahmet Cihat Baktir, Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Deepedge: A deep reinforcement learning based task orchestrator for edge computing. *IEEE Transactions on Network Science and Engineering*, 2022.

Supplementary information

Acknowledgements

SBM received funding from the "Infrastructure for exchange of research data and software" (crc1456-inf) project.