Simulation of Energy Consumption in Fog Computing Environments

by

Philipp Wiesner
Matriculation Number 389759

Master's Thesis

Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Complex and Distributed IT Systems

Reviewers:
Prof. Dr. habil. Odej Kao
Prof. Dr. rer. nat. Volker Markl

Supervised by:
Dr. Lauritz Thamsen

30. Mai 2020

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used. The independent and unaided completion of the thesis is affirmed by affidavit.

Berlin, 30. Mai 2020 Philipp Wiesner

Abstract

Fog Computing is an emerging paradigm that extends traditional cloud computing by bringing processing capabilities closer to the edge of the network. A distributed layer of small-size data centers is supposed to filter, aggregate, and preprocess the vast amount of data generated by IoT, smart cities, and other interconnected environments. Despite of constant improvements in efficiency, today's data centers and networks require tremendous amounts of energy, and their consumption is expected to rise further due to ever-increasing demand. A principal research question is whether and how fog computing can curb this trend.

As of today, real-life deployments of fog infrastructure are rare, which is why a significant part of the research relies on simulations. However, none of the many existing simulators offers the possibility to model power usage beyond compute nodes and battery-constrained end devices. This thesis introduces LEAF, a simulator for modeling Large Energy-Aware Fog computing environments. It provides a holistic but granular energy consumption model, covering different parts of the infrastructure, including the network, and enables research on energy-conserving fog architectures. The model can trace back the relative power requirements of applications and allows for the implementation of energy-aware task placement strategies and dynamic energy-saving mechanisms. Furthermore, LEAF is designed to enable simulations with hundreds or thousands of devices and applications.

Based on a smart city scenario, a series of experiments was carried out to demonstrate the usefulness of the proposed model. The results indicate that fog computing can conserve energy by significantly reducing wide area network usage, but at the same time increases static power consumption due to additionally required infrastructure.

Zusammenfassung

Fog Computing ist ein neuartiges Konzept für Netzwerkarchitekturen, das traditionelles Cloud Computing um eine dezentrale Datenverarbeitung am Rand des Netzwerks erweitert. Ziel ist es, die riesigen Datenmengen, welche vor allem durch das Internet der Dinge, intelligente Städte und andere vernetzte Umgebungen erzeugt werden, so nah wie möglich an den Erzeugern zu verarbeiten. Bereits heute verbrauchen Rechenzentren und Netzwerkinfrastruktur enorme Mengen an Energie und es ist zu erwarten, dass dieser Verbrauch aufgrund der ständig wachsenden Nachfrage in Zukunft weiter steigt. Eine grundlegende Forschungsfrage ist deshalb, ob und wie Fog Computing diesen Trend eindämmen kann.

Da Fog Computing bis zum heutigen Tag in der Praxis noch wenig Anwendung findet, stützt sich ein bedeutender Teil der Forschung auf Simulationen. Jedoch beschränken sich existierende Simulatoren auf die Energieverbrauchsmodellierung von Rechenknoten und Endgeräten und lassen die Kosten für Netzwerkübertragung außen vor. Diese Arbeit präsentiert LEAF, einen Simulator, welcher eine ganzheitliche, aber granulare Energieverbrauchsmodellierung von groß angelegten Fog-Computing-Umgebungen ermöglicht und damit Forschung an energiesparenden Fog-Computing-Architekturen erlaubt. Darüber hinaus kann das Modell den relativen Energiebedarf von Anwendungen zurückverfolgen und ermöglicht die Implementierung von adaptiven Algorithmen für die energieeffiziente Verteilung von Aufgaben im Netzwerk sowie die Einbindung dynamischer Energiesparmechanismen. LEAF kann Szenarien mit Hunderten oder Tausenden von Geräten und Anwendungen simulieren.

Ausgehend von einem Smart-City-Szenario wurde eine Reihe von Experimenten durchgeführt, um die Funktionalität des vorgestellten Modells zu demonstrieren. Die Ergebnisse deuten darauf hin, dass Fog Computing durch den reduzierten Netzwerkverkehr in der Lage ist, Energie zu sparen. Gleichzeitig erhöht sich jedoch auch der statische Stromverbrauch aufgrund der zusätzlich erforderlichen Infrastruktur.

Contents

1 Introduction

With the rise of the internet of things (IoT) and the beginning of the fourth industrial revolution, many sectors and environments are expected to become interspersed with connected sensors. Examples are so-called smart cities: From traffic management and transportation systems over water networks, energy supply, and waste management to schools and hospitals, a variety of sensors is supposed to help managing resources more efficiently [1, 2]. As these sensors produce an ever-increasing amount of data, the classical cloud computing approach of sending all information directly to a few powerful data centers, often far away from the data source, is practically not feasible anymore. On top of that, energy consumption of information and communications technology is getting out of hand, already accounting for more than 10 % of global power consumption and expected to exceed the 20 % mark by 2030 [3]. In view of the ongoing climate crisis, it is crucial to reduce this figure in the future.

One attempt to solve these emerging problems is to deploy a distributed layer of compute nodes close to the edge of the network that filters, aggregates, and preprocesses the raw data generated by end devices and sensors. This new paradigm, called fog computing, is expected to save large amounts of bandwidth, reduce latencies for time-sensitive applications and increase end-user privacy as the fog nodes can take care of aggregating and anonymizing data [4]. However, due to their distributed and heterogeneous nature, fog computing environments are much more challenging to manage than centralized data centers [5].

A major open research question is whether fog computing will be able to reduce the overall energy consumption of future computing infrastructure or if it may even be responsible for a further increase [4, 6, 7]. Although the general trend is continuing the centralization of data centers to so-called hyperscale data centers for efficiency gains [8], research has shown that distributed architectures could save between 14 % to more than 80 % of the infrastructure's total energy consumption compared to fully centralized architectures [9, 10].

Some aspects of this question have already been studied in depth. For example, energy-efficient data centers have been a research area for years, since electricity costs account for a significant part of a data center's operating expenditure [11]. Another field of research with many contributions is the in-

vestigation of energy consumption of battery-constrained edge devices like mobile phones or sensors [12–14].

Less attention has been devoted to the question to what extent fog computing may affect the overall power usage of network infrastructure. On the one hand, it has been shown that fog computing can highly reduce network traffic due to it's geo-proximity to data producers [9, 10, 13, 15], which, as a consequence, also lowers the network's power usage. Furthermore, nano data centers usually require less cooling than bigger data centers [16]. On the other hand, hyperscale data centers are highly optimized for energy-efficiency and power-proportionality [8, 17], and some recent studies suggest that their overall consumption may be significantly lower than previously assumed [18]. Large-scale deployments of distributed fog infrastructure will inevitably impose additional power requirements and it is questionable whether a performance per watt comparable to that of centralised data centers can be achieved.

Since large-scale, real-life deployments of fog computing infrastructure are still to come [5, 19], research on new communication protocols and scheduling algorithms is mostly limited to theoretical models and emulation tools. Over the last four years numerous emulators [20–22], simulators [13, 23–26] and analytical models [6, 9, 15] have been proposed. These tools are essential for finding meaningful network architectures and thus planning future investments in fog data centers and network infrastructure.

Although there exist some elaborate cloud computing simulators with the ability to assess energy consumption [27–29], only few fog computing simulators comprise energy models at all. The ones that do are either overly simplistic [23] or focus only on modeling the consumption of specific entities within the fog environment like battery-constrained mobile phones or wireless sensors [13, 14, 30]. However, in order to obtain a holistic view of the fog's energy consumption, also the power usage of the network - in particular the wide area network (WAN) - has to be examined. In addition to insufficient energy consumption modeling, many simulators fail to realistically model fog environments in general by leaving out substantial elements like the mobility of edge devices [23, 24]. Furthermore, tools that model the network and computing environment in extensive detail are usually not able to perform simulations with a large number of devices due to performance limitations.

Existing analytical approaches on modeling the energy consumption in fog computing environments mostly suffer from the fact that they incorporate a very high level ob abstraction and, for example, make the invalid assumption that fog nodes are homogeneous [9, 15]. Moreover, they do not allow for online decision making during the simulation, which is indispensable when researching on scheduling algorithms or task placement strategies.

1.1 Model Requirements

As stated above, existing simulators cannot sufficiently model energy consumption of fog computing environments in large-scale experiments. There is a need for a simulator that allows identifying the causes of power usage and enables research on energy-efficient architectures and algorithms to fully exploit the fog's potential for conserving energy. Derived from these use cases and considering the numerous challenges in modeling fog computing environments [7, 31, 32], the following requirements are placed on the proposed model:

(1) Realistic Fog Computing Environment (1.1) The model enables the simulation of distributed, heterogeneous, and resource-constrained fog computing environments, where compute nodes are linked with different types of network connections. (1.2) The model ensures location-awareness of nodes, for example, to calculate closeby mobile access points. Nodes can be mobile; namely, their location can change according to predetermined or randomized patterns, and they can join or leave the topology ant any time. (1.3) The model enables the simulation of different streaming applications, from linear data pipelines to applications with complex topologies, including a multitude of data sources, processing steps, and data sinks.

(2) Holistic Energy Consumption Model The model is able to asses the power usage of all (2.1) data centers, (2.2) edge devices, and (2.3) network links in the fog computing environment at any point in time during the simulation. Hence, it allows for the creation of detailed power usage profiles for individual parts of the infrastructure. (2.4) Additionally, the model makes the energy consumption of infrastructure traceable to the responsible applications to let researchers identify inefficiencies and opportunities to improve in the network's architecture.

(3) Energy-Aware Online Decision Making The model enables simulated schedulers, resource management algorithms, or routing policies to asses the power usage of individual compute nodes, network links, and applications during the run time of the simulation and, hence, allows for energy-aware decisions. This enables research on algorithms such as energy-efficient task placement strategies or routing policies. Furthermore, the model enables said algorithms to dynamically apply energy-saving mechanisms such as powering off idle fog nodes.

(4) Performance and Scalability The model allows the simulation of complex scenarios with hundreds of parallel existing devices that execute thousands of tasks. The runtime of simulations of such scale is at least two orders of magnitude faster than real-time.

1.2 Contributions

This thesis presents LEAF, a simulator for modeling Large Energy-Aware Fog computing environments. LEAF features a comprehensive energy consumption model that enables measuring, analyzing, and improving the overall energy consumption of fog computing scenarios. Researchers can compare the power usage characteristics of heterogeneous network infrastructures with different kinds of wired and wireless connections like IEEE 802.11 and mobile telecommunications technology. End devices can be mobile and join or leave the network topology during the simulation. The realistic infrastructure and application model enables research on energy-conserving fog computing architectures, which can lead to more well-founded decisions when planning future investments in fog infrastructure.

LEAF combines approaches from numerical and analytical modeling: Network traffic and energy consumption are not modeled on a low level but via parameterizable mathematical equations. This abstraction allows the simulation of hundreds or thousands of devices and applications and ensures that the state of simulations is easy to analyze at any point in time, similar to analytical models. Nevertheless, LEAF is based on a discrete-event simulator (DES): The network topology, task placements, and all other characteristics of infrastructure and applications can change over time, enabling the implementation of adaptive, energy-aware task placement strategies or routing policies. To facilitate the implementation of these algorithms, LEAF can calculate the relative power requirements of applications that run on the fog computing environment, exposing potential inefficiencies. Furthermore, different kinds of energy-saving mechanisms can be incorporated into the simulation and evaluated regarding their impact.

A prototype of LEAF was implemented in Java, based on the CloudSim Plus [33] simulator. The prototype has been evaluated by conducting a series of experiments in a smart city scenario and it has been demonstrated that LEAF fulfills all requirements. By analysing the power usage of individual infrastructure components and applications over time, it has been shown that fog computing has a great potential for reducing overall energy consumption by avoiding power-intensive traffic over WAN, especially mobile access networks such as 4G LTE, which is in line with previous research on this topic [6, 9, 10]. Besides demonstrating how LEAF can be used to find energy-conserving architectures, an adaptive energy-saving mechanism has been implemented which further reduced the overall energy consumption of infrastructure by putting idle fog nodes to sleep during off-peak periods.

1.3 Outline of the Thesis

The remainder of this thesis is structured in the following way.

Chapter 2 provides background information on the discussed topics. First, fog computing and the many related edge computing paradigms are presented. Second, demand for emulation and simulation, and differences between these two approaches are outlined. Regarding simulation, a special focus is put on the difference between analytical and numerical modeling. Third, important concepts for describing energy consumption of computing and network infrastructure are presented.

Chapter 3 reviews the related work and outlines its limitations with regards to solving the described problem. First, three established energy-aware cloud computing simulators are presented. Second, the most important simulators in the field of fog computing are introduced and analyzed. Third, some existing analytical approaches to model fog computing environments are explained.

Chapter 4 presents the approach and concepts behind LEAF. First, the infrastructure, application, and power models are introduced. Second, a large variety of examples are given on how to parameterize power models for different kinds of infrastructure. Third, different energy-saving mechanisms are presented and how they can be incorporated into the model. Fourth, it is explained how LEAF manages to trace back the infrastructure's power usage to the applications in order to enable energy-aware task placement strategies.

Chapter 5 evaluates LEAF. First, the prototypical implementation is presented. Second, the experimental setup and its parameterization are explained. Third, the results of the experiments are explained, analyzed, and discussed to demonstrate the usefulness of LEAF. Fourth, a special focus was put on the performance requirement of the presented simulator. Fifth, it is shown that LEAF sufficiently fulfills all requirements.

Chapter 6 concludes the thesis by summarizing the findings and contributions. Furthermore, possible directions for future research are suggested.

2 Background

This chapter provides an overview of relevant concepts and terms in the fields of fog computing, network simulation, and energy consumption. Section 2.1 explains the rationale behind fog computing and the many related edge computing paradigms. Section 2.1 covers basic concepts behind modeling and simulating networks in general and fog environments in particular. Section 2.3 concludes with several definitions and concepts about describing, measuring, and optimizing energy consumption in networks and computing infrastructure.

2.1 Fog and Edge Computing Paradigms

As more and more devices become connected to the internet, the amount of produced data is soon expected to overwhelm today's network and compute infrastructure in terms of capacity [4]. In their latest annual internet report, Cisco states that there are already 8.9 billion machine-to-machine connections in 2020, and they expect this number to grow to 14.7 billion by 2023 [34]. Furthermore, there are serious privacy and security concerns associated with overcentralizing the processing and storage of IoT data [35].

Fog and edge computing describe computing paradigms targeted at solving the problems posed by the rise of IoT. A distributed layer of smaller data centers, termed nano data centers, cloudlets, or fog nodes, is supposed process data close to its producers - the so-called edge of the network [36]. Edge devices can be anything from mobile phones or air quality sensors to connected cars and traffic lights.

The geo-proximity to end-users can lead to highly reduced latencies for real-time applications, as well as less overall network traffic. Fog nodes can either directly respond to actuators or at least preprocess, filter, and aggregate data to reduce the volume sent over the WAN [37]. Moreover, many applications do not require centralized control and could instead operate - and therefore, scale - in a fully distributed manner. An example is smart traffic management, where connected cars and traffic lights can act by communicating purely among themselves [38]. Nevertheless, fog computing is mostly meant to complement cloud computing, not replacing it [15].

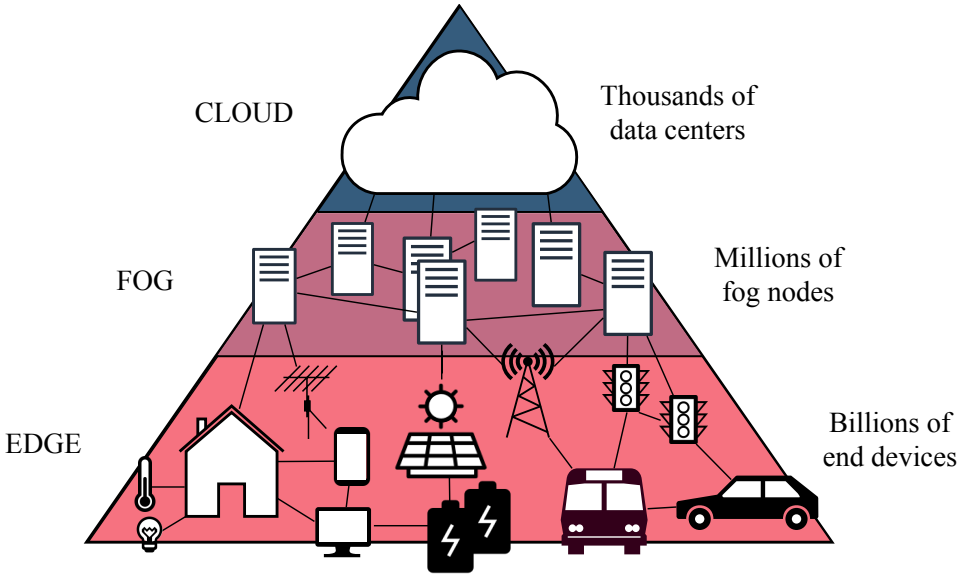

Figure 2.1: Pyramid of fog-related computing paradigms. Adapted from Qayyum *et al.* [26], Figure 1.

From a privacy point of view, fog nodes can anonymize data before forwarding it to prevent misuse by the cloud provider or another third party [35]. For example, more and more smart meters, namely electricity meters with internet access, are getting deployed. They regularly report the current power consumption to network operators to enable more efficient control of the electricity network. However, the raw sensor data might contain confidential information, such as when users switch the television on or off. Fog nodes could, for instance, aggregate and anonymize the measured data from different households to ensure privacy.

Besides fog computing, there exist numerous related compute paradigms: Edge computing, mist computing, mobile cloud computing, or cloudlet computing, to name a few. Despite attempts to highlight differences in these approaches [5, 39], like the proximity of computation to the edge of the network, or the adoption of pure peer-to-peer concepts, they mostly describe similar ideas. Further, terminology is sometimes used inconsistently across research papers, which can be attributed to the fact that the research field is still very young. In recent years, a consensus has emerged that edge computing refers to computing on - or extremely close to - the end devices, while the fog is located between edge and cloud. This hierarchy is illustrated in Figure 2.1.

For the course of this thesis, fog computing refers to all fog and edge related computing paradigms. LEAF is flexible enough to model all kinds of distributed topologies, including pure peer-to-peer architectures.

2.2 Simulation of Fog Computing Environments

Although fog computing is a rapidly growing field, as of today, there exist almost no commercial deployments on which researchers can conduct experiments [5, 19]. Moreover, setting up, managing, and maintaining a large number of heterogenous devices incurs high operational costs. To facilitate research on fog computing, a variety of emulators [20–22] and models [6, 9, 13–15, 23–26, 30, 40] have been presented in recent years. This section aims to provide some background on simulation and modeling. It is divided into two parts. Section 2.2.1 explains the advantages and disadvantages of emulation and simulation. Section 2.2.2 presents the two main approaches when designing a model: Analytical and numerical modeling.

2.2.1 Emulation and Simulation

Emulators enable the execution of real applications in virtual fog computing environments. By introducing latencies or failure rates among virtualized compute nodes, the emulator mimics specific characteristics of the fog environment during execution. This approach approximates reality closely and allows for empirical results but has one major drawback: The execution of applications in emulated environments is computationally even more expensive than running them natively. This constraint can be mitigated up to a certain point by the scalability of cloud computing. Nevertheless, large-scale experiments with thousands of devices and applications soon become very costly.

A different approach towards research on fog computing environments is the development and usage of models. A model abstracts some of the fog environment's underlying details and thus can make experiments more efficient, scalable, and - most importantly - easier to analyze. Creating and applying models can result in a more comprehensive understanding of the problem space by identifying "driving" variables [41]. Yet, modeling is invariably a trade-off between realism and simplicity, and what makes a good model always depends on the use case.

The execution of models is called simulation. A considerable advantage of simulation to other approaches is that its runtime is not necessarily bound to real-time. A sufficiently abstract simulation could, for instance, reproduce several hours of a fog computing environment's behavior in a few minutes. However, modeling also has its pitfalls: Wrong or biased assumptions can lead to invalid simulation results. This is especially relevant in fog computing environments where almost no real-life deployments of infrastructure could be used for validation by comparison [31, 32].

2.2.2 Analytical and Numerical Modeling

There are two main categories of modeling approaches: Analytical and numerical modeling. As the proposed model facilitates aspects from both approaches, they are briefly explained in this section.

Analytical models provide a purely mathematical description of the underlying environment. All properties of infrastructure and applications are expressed through a set of equations. For example, the throughput q of a network link can be described as

$$q = t \cdot b \cdot (1 - p),$$

where t is the time in seconds, b is the bandwidth in bits per second, and p is the probability of a packet loss. Analytical models are very abstract and thus enable a high-level understanding of the problem. Parameters and their effect on the result can be easily analyzed in isolation or consumption. The disadvantage of analytical models is that they have no sense of time or state within the system. For example, it is challenging to model an application scheduler in a meaningful way because it reacts to certain state and state changes in the infrastructure. Also, the simulation of heterogeneous infrastructures is not easily possible. Section 3.3 points out that existing analytical models on fog computing environments make various simplifications like assuming an equal distribution of edge devices and virtual machines (VMs) and disregarding mobility.

Numerical models overcome this limitation by using a time-stepping procedure to simulate the model over time. This mechanism enables the modeling of more complex scenarios that can incorporate scheduling, mobility models, and heterogeneous infrastructure and applications. Numerical models are commonly called "simulators" - although technically also the execution of an analytical model is simulating the modeled environment. This thesis follows this convention: The term "simulator" refers to numerical models only.

To take up the previous example, we define a numerical approach for modeling the throughput q of a network link. A network simulator could, for example, send packages of size s according to a random distribution D with a probability of p. Running this simulation for time t and then measuring how many bits have been sent in total results in the throughput.

Note that in the analytical model, p was a real number between 0 and 1. In the numerical model, it is an actual probability that a package gets sent. Hence, the numerical model entails randomness. Similarly, also D is a random distribution, in this context, usually a Poisson distribution. In contrast to the analytical model, which always returns precise - albeit more abstract - results, one should run sensitive simulations several times and aggregate

their results to reduce the effect of randomness. This circumstance reveals a disadvantage of numerical models: Their results cannot be analyzed as conveniently. Additionally, at least within the scope of network simulations, the execution of numerical models can take a lot of time. Analytical models are usually very fast.

Numerical network models are almost invariably so-called *discrete-event simulations* (DES) - in fact all simulators presented in Sections 3.1 and 3.2 are based on a DES. A DES is constituted of a queue of events that each take place at a certain time. Events get processed in chronological order, and each processing step may result in new events that are placed in the queue. It is assumed that the system state between consecutive events does not change; consequently, the simulator can jump directly from event to event. This allows simulations faster than real-time. The simulation is over if there are no more events in the queue, or a predefined time limit has been reached. A DES scales roughly linearly with the number of events, but due to its sequential nature, it is particularly hard to parallelize [32].

2.3 Power Usage and Energy Consumption

This section introduces basic definitions and explains relevant terms and techniques used for describing and modeling energy consumption in computing and network infrastructure.

The SI unit of power is the watt (W). In the context of electricity, it can be described as the electrical work when a current of one ampere (A) flows through an electrical potential difference of one volt (V); in mathematical terms $W = A \cdot V$. When a device draws a certain amount of power over time, it consumes energy for which there exist two common units of measurement: In everyday life, for example, in electricity consumption bills or reports, energy is usually stated in watt-hours (Wh). The SI derived unit for energy is the joule (J). Both represent the energy needed to produce or consume one watt of power over the duration of one hour (Wh) or second (J = Ws). Consequently, the conversion between these units is $1\,\text{Wh} = 3600\,\text{Ws} = 3600\,\text{J}$.

The total power usage of electronic hardware can is often described as the sum of its static and dynamic usage. The static, or idle, power usage P_{static} is consumed when the hardware is turned on but not performing any work. The dynamic, or load-dependent, power usage P_{dyn} depends on the utilization of the resource.

2.3.1 Performance per Watt

A general measure for describing the energy effectiveness of hardware is performance per watt, namely the rate of computation that a piece of hardware can provide for every watt of consumed power. Depending on the field of application, the metric used to specify performance varies: Scientific computing and computer graphics require large amounts of floating-point calculations, which is why the performance of supercomputers and GPUs is described in FLOPS (floating-point operations per second). The performance of non-scientific and general-purpose computing units is mostly expressed in MIPS (Millions of instructions per second) or other comparable benchmarks. As IoT applications are very diverse, in the following MIPS are used to determine a compute node's performance.

When designing detailed models of computing infrastructure, it is possible to determine performance per watt metrics for additional components like memory, hard discs, mainboards, fans, or the power supply itself [42]. Nevertheless, there is usually a clear correlation between MIPS and memory and storage access [43, 44]. Thus, in higher-level models, it is reasonable to describe a compute node's performance per watt based only on its computational load.

For network links, the term "performance" refers to the bandwidth, expressed in in bit/s [45, 46]. Hence, the performance per watt of a network link is specified in $(\mathrm{bit/s})/\mathrm{W}$, which equals to bit/J. In practice, most papers state the energy-efficiency as the inverted performance per watt. For example, efficiency of wired and wireless links is mostly stated in nJ/bit.

Expressing the dynamic power consumption of hardware as a single number presupposes that a linear dependence between performance and energy consumption exists. Surprisingly, this holds for the vast majority of computing and networking hardware. Studies have shown, that the power usage profile of nearly all hardware components is in fact linear [45–47]. However, the precise power consumption of some devices does depend on more variables than just the performance. A wireless transmitter, for example, requires more energy to send messages to receivers further away than to receivers very close, due to energy dissipation in free space [12].

2.3.2 Energy-Proportional Computing

Figure 2.2 depicts a measurement conducted by Barroso & Hölzle [43] on the energy efficiency of a server. It becomes evident, that the performance per watt drops dramatically when the target is under low load. This inefficiency is a huge problem: Albeit idle severs can require up to 60 % of their peak power draw [43, 44], on average, cloud data centers are only utilized at 20 %

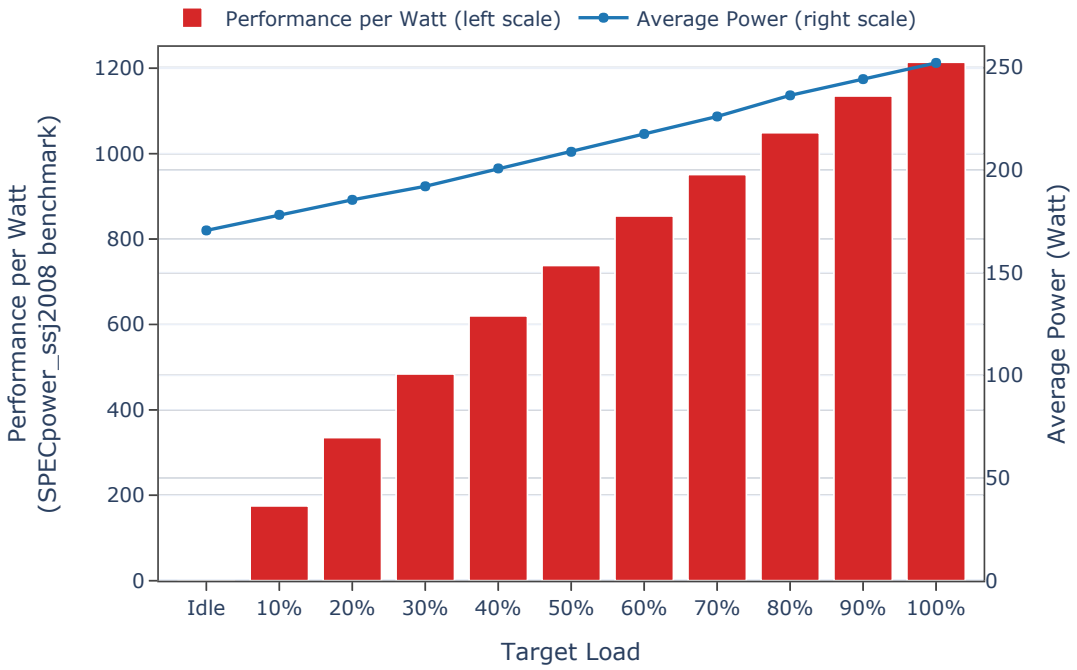

Figure 2.2: Example benchmark result for the SPECpower_ssj2008 power benchmark. Red bars describe energy efficiency at different target loads. The red line illustrates power consumption at different target loads. Adapted from Barroso & Hölzle [43], Figure 5.3.

to 30 % of their full capacity [43]. This means a huge potential for energy savings lies in the reduction of static energy consumption.

Energy-proportional computing is a concept proposed by Barroso & Hölzle [47], which states that energy consumption of a resource should ideally be directly proportional to its workload. Consequently, the goal of energy-proportional computing is to reduce the idle power consumption P_{static} of resources to zero.

Energy-proportional data centers and networks are desirable because they can run in an efficient manner even when not fully utilized. In practice, there are various open technical challenges in reducing static energy consumption of hardware. While CPUs became more and more energy-proportional over the years thanks to energy-saving mechanisms like dynamic voltage and frequency scaling (DVFS) [48], other components like memory, network and networks are traditionally energy disproportional [43]. Static energy consumption of resources like network adapters or routers can make up to more than 90 % of their maximum energy consumption [45, 49]. Nevertheless, due to many technological advances over the last decade, modern hyperscale data centers can operate in extremely energy efficient manners [17].

2.3.3 Power Usage Effectiveness

A global standard and frequently used metric to determine efficiency of data centers is called Power Usage Effectiveness (PUE) [50, 51]. PUE describes the relation of energy spent on actual computing compared to the data center's total consumption, including overhead for cooling, lighting, controls, or uninterrupted power supply. *The Green Grid* consortium defines it as:

$$\text{PUE} = \frac{\text{Total Facitily Energy}}{\text{IT Equipment Energy}}$$

Accordingly, a PUE of 1.0 describes a "perfect" data center where all consumed energy is used solely for computing. A PUE of 1.3 indicates that there is a 30 % overhead in operating this data center.

While the industry average PUE in 2019 was 1.67 [52], big hyperscale data center operators like Google manage to run data centers at PUEs of close to 1.1 [17]. This is achieved by utilizing optimized, energy-efficient hardware as well as advanced operation software, which, for example, applies machine learning to optimize cooling [17].

3 Related Work

This chapter surveys related work to point out the contributions of previous research and place the contributions of this thesis in the proper context. In particular, it is intended to show that none of the existing simulators nor analytical models is able to meet the requirements defined in Section 1.1. Table 3.1 provides an overview of the strengths and deficiencies of the different models presented in this chapter. The results are clustered in four groups that correspond to the requirements: Whether the model is suitable to model realistic fog environments, whether it features energy modeling for all parts of the infrastructure and applications, whether it supports online-decision making and whether it is scalable. ✓ implies that the model sufficiently meets the requirement; ❑ indicates that the model only partially complies with the requirement.

Table 3.1: Strengths and deficiencies of related work.

		LEAF	iCanCloud [27]	GreenCloud [28]	CloudSim [29]	iFogSim [23]	IOTSim [24]	EdgeCloudSim [25]	PureEdgeSim [13]	FogNetSim++ [26]	Sarkar & Misra [15]	Jalali *et al.* [6]	Ahvar *et al.* [9]
Fog Model	Infrastructure	✓				❑	❑	✓	✓	✓	❑	❑	❑
	Applications	✓				❑	✓	❑	❑	✓		❑	
	Mobility	✓						✓	✓	✓			
Energy Model	Data Centers	✓	✓	✓	✓	✓				✓	❑	✓	✓
	Edge Devices	✓							✓	✓	❑	✓	✓
	Network	✓	✓	✓					❑	❑	❑	✓	✓
	Applications	✓								✓			
Online-Decision Making		✓	✓	✓	✓	✓	✓	✓	✓	✓			
Scalability		✓	❑		❑	❑	❑	❑	❑		✓	✓	✓

Section 3.1 provides an overview of the most popular cloud computing simulators and their approach to modeling energy consumption. Section 3.2 highlights differences in various existing fog computing simulators and examines if and how they model energy consumption. Section 3.3 presents three analytical approaches for modeling the overall energy consumption of fog computing architectures.

3.1 Energy-Aware Cloud Computing Simulators

Since cloud computing has been a field of research for many decades, numerous simulators exist for different purposes. The energy consumption models of three well-known cloud computing simulators, namely iCanCloud [27], GreenCloud [28], CloudSim [29], are presented here. Although cloud computing simulators are not primarily designed to simulate heterogeneous fog environments, they often form the basis for fog computing simulators. None of the introduced simulators features a way to trace back infrastructure energy consumption back to the applications.

iCanCloud [27] is a cloud computing simulator based on OMNeT++, a DES library for network simulators. It has a heavy focus on performance and usability; namely, it allows the execution of large-scale experiments and features a graphical user interface for configuring experiments. Castañé *et al.* [53] extended iCanCloud with a fine-grained energy consumption model for hardware components and enables the simulation of resource provisioning policies which balance the trade-offs between performance and energy consumption. Although its focus on performance, it models cloud computing environments in great detail, which results in comparably poor scalability. Besides, the project is hard to extend and has not received any updates since 2013.

GreenCloud [28] is a comprehensive cloud computing simulator when it comes to modeling energy consumption. It is based on ns-2, another discrete-event network simulator, and implements a full TCP/IP protocol reference model. As a result, it enables highly realistic modeling of network traffic and energy consumption of different components. However, this comes at a cost: Experiments executed in GreenCloud are a lot slower than comparable experiments in CloudSim or iCanCloud, making it impractical for large-scale experiments. Moreover, similarly to iCanCloud, GreenCloud is very hard to extend and no longer actively maintained.

CloudSim [29] is arguably the most popular cloud computing simulator and provides the basis for the vast majority of fog computing simulators [13, 23–25, 30]. Unlike iCanCloud and GreenCloud, it is not based on an existing network simulator but comprises a custom discrete-event loop implemented in Java. This more high-level modeling sacrifices precision for performance.

CloudSim allows the modeling of cloud components such as data centers, hosts, and VMs as well as resource provisioning policies for CPU, storage, memory, and bandwidth utilization models. It is designed to be easily extendable, which led to a multitude of extensions, many of which have been integrated into the core system. Since version 2.0, CloudSim features an energy model proposed by Beloglazov & Buyya [54] that allows modeling the power usage of CPUs. Apart from being relatively simplistic, its main limitation is that it cannot be used in conjunction with CloudSim's network model [55]. The extension DartCSim+ [56] addresses this by introducing energy-aware network components that even allow modeling the energy consumed by VM migrations. However, it introduces a lot of conceptual and computational complexity, is coupled towards modeling inter-data center traffic, and does not support different link types like wireless connections. Because of that, this energy model is not appropriate for modeling energy consumption in fog computing environments.

Since the original CloudSim project has many long-standing bugs and has been mostly unmaintained for several years, Filho *et al.* [33] presented a fork called CloudSim Plus. Its improved code quality, documentation, and performance paired with many new features have established it as one of the most extensive tools for simulating cloud computing. The prototypical implementation of this thesis, explained in Section 5.1, is based on CloudSim Plus.

3.2 Fog Computing Simulators

The high interest in fog computing, combined with the lack of real-life deployments for testing, has led to a multitude of fog computing simulators within the last four years. The most established ones are surveyed in this chapter and analyzed regarding their approach to energy consumption modeling.

iFogSim [23] was one of the first fog computing simulators and is based on CloudSim. It enables the modeling of sensors, actuators, gateways, fog, and cloud data centers. A principal deficiency is that every simulated device has to be defined manually, and the network topology has to remain static during the simulation. This limits scalability and entirely prevents the simulation of mobile entities. iFogSim inherits CloudSim's energy model for data center hosts, but there is no way to model the power usage of network traffic, edge devices, or applications.

IoTSim [24] extends CloudSim with distributed storage and big data processing layers. It can be used to simulate IoT applications, namely MapReduce processes, by mocking network and storage delays. IoTSim has no concept of mobility, nor does it support the simulation of energy consumption.

EdgeCloudSim [25] addresses many of the limitations of iFogSim and IoTSim by incorporating a more realistic network model and supporting the dynamic generation of mobile edge devices. Additionally, it features a realistic and tunable network traffic load generator. Nevertheless, EdgeCloudSim neither supports the modeling of mobile telecommunications technology, nor does it incorporate any energy consumption model.

PureEdgeSim [13], one of the most recent CloudSim-based fog computing simulators, is characterized by its ability to model task offloading at the very edge of the network. Therefore it can simulate cloud, fog, and pure edge scenarios as well as combinations of these. To simulate battery constraints, it features a comprehensive energy consumption model for edge devices, including WiFi. However, the energy consumption of data centers or other types of network links cannot be modeled. Furthermore, during the implementation of LEAF, several bugs were discovered in PureEdgeSim, putting into question the validity of this simulator.

FogNetSim++ [26] is based on the OMNET++ network simulator, similar to iCanCloud. It inherits many of its properties and enables the simulation of unreliable network links and faulty packages. Users can model highly complex scenarios and implement custom mobility models and schedulers. Besides TCP and UDP, FogNetSim++ implements various IoT-specific communication protocols like Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and the Advanced Message Queuing Protocol (AMQP). FogNetSim++ is the only fog computing simulator presented here, which features an energy model for applications. Albeit looking promising at first sight, there is barely any documentation available for this simulator, and there have been no updates since the initial release, which may be reasons why it has not received much attention from the community. Furthermore, FogNetSim++ does not feature any abstraction or energy modeling for WAN, limiting the applicability for comparing fog to cloud computing architectures. Similar to iCanCloud and GreenCloud, the detailed simulation of network traffic has a negative impact on performance and scalability.

3.3 Analytical Models on Fog Energy Consumption

Besides simulators, researchers also investigate on the energy consumption of different fog computing architectures via analytical models. These models usually lead to a comprehensive understanding of the problem space and allow a detailed analysis of the effect of input parameters, see Section 2.2.2. Furthermore, they are suitable for modeling large-scale scenarios since they are very fast to execute. Nevertheless, they introduce a high level of abstraction - sometimes to an unrealistic extend. All presented approaches fail to model heterogeneous infrastructures by making idealizing assumptions, such

as expecting every fog node to have the same processing power. Moreover, since analytical models cannot represent change over time, none of the presented approaches in this section allows the modeling of mobility or enables online decision making. This constraint also prevents research on energy-saving mechanisms.

Sarkar & Misra [15] proposed an analytical model for fog computing architectures in order to compare their service latency and energy consumption to traditional cloud computing. Although the energy consumption of all individual components is described by mathematical equations, the authors do not distinguish between static and dynamic energy consumption. This greatly limits the informative value of the analyses. Furthermore, there exists no concept for applications. Consequently, the model is only suitable for a high-level examination of different infrastructures. The researchers state that according to their model, fog computing architectures can improve the mean energy consumption by 40.48 % compared to traditional cloud infrastructures.

Jalali *et al.* [6] took a more granular approach towards modeling the energy consumption of fog computing environments. Their model consists of two components: The first part is a flow-based model for highly shared equipment such as core routers and switches, which calculate the energy consumption based on the data throughput. The second part is a time-based model for end-user equipment that is usually not shared among many clients. This model calculates the consumption based on the amount of time it spends providing access to a service. The application model of Jalali *et al.* focuses on offloading applications to fog nodes for content delivery. They validate it by measuring real-life data from a Wordpress deployment that can offload content delivery to fog nodes. However, the model does not support the simulation of different applications at the same time. The authors conclude that fog computing can complement cloud computing and lead to lower energy consumption, depending on design factors such as network architecture and the type of applications. A decisive factor is the type of access network: offloading calculations over 4G LTE are orders of magnitude more expensive than, for example, over passive optical networks. Another interesting finding is that the number of hops in the WAN does not have a significant impact on overall energy consumption.

Ahvar *et al.* [9] introduce an analytical model for assessing the total energy consumption of various fog- and cloud-related architectures. Their model differentiates between static and dynamic power consumption and includes the consumption of network devices and cooling. They compared four different architectures from fully centralized to fully distributed. According to the authors, the fully distributed architecture consumes between 14 % and 25 % less energy than the fully centralized because it avoids intra-data center traf-

fic and large-size cooling systems. Even though this is the most exhaustive of the three analytical models presented in this section, it also incorporates several oversimplifications like an even distribution of VMs among the data centers or assuming that all fog nodes connect to the same amount of devices. Moreover, there is no real concept of applications with interdependent tasks, nor is there a way to calculate the relative power requirements of VMs.

4 Simulation Model

This chapter introduces LEAF, a simulator that aims to eliminate the shortcomings of existing models, thus enabling comprehensive modeling of Large Energy-Aware Fog computing environments. The most evident deficiency of existing fog computing simulators is that, if at all, only the energy consumption of data centers and end devices is simulated. However, for obtaining a holistic energy consumption model, the network must be considered too. The simulators presented in Chapter 3, which feature network modeling, do this very thoroughly. They model every single network package individually, sometimes even implementing entire communication protocols like TCP. This degree of detail negatively impacts performance. Since Requirement (4) demands scalability, LEAF takes a different approach: Network links are considered resources, similar to compute nodes, that have certain properties and constraints and a particular performance per watt. Data flows between tasks allocate bandwidth on these links.

Modeling networks in this more abstract fashion is inspired by the analytical approaches presented in Section 3.3. It enhances the performance of the DES because fewer events get created. Furthermore, the analysis of a model's state at a particular time step becomes very easy: The user can comprehend the current network topology, all data flows, and allocated resources of running applications solely by observing the simulation state at any point in time. A disadvantage of this kind of modeling is the higher level of abstraction, and therefore, possibly less accurate results.

Although some components of LEAF are based on analytical approaches, at its core it remains a numerical model - to be exact a DES. The user can adjust the infrastructure's topology, the placement of applications, and the parameterization of resources over time. This flexibility has two decisive advantages over purely analytical models: First of all, fog computing environments can be represented in a much more realistic way. LEAF allows for dynamically changing network topologies, mobility of end devices, and varying workloads. Secondly, LEAF enables online decision making during the simulation, enabling research on energy-aware task placement strategies, scheduling algorithms, or traffic routing policies.

The following sections provide details on the proposed simulator. Sections 4.1 and 4.2 present the infrastructure and application model of LEAF, respec-

tively. Section 4.3 explains how power and energy are being modeled. Section 4.4 provides a large number of examples of how individual power models can be parameterized and what needs to be considered. Section 4.5 puts a special focus on different kinds of energy-saving mechanisms. Last, Section 4.6 explains how LEAF can trace back the infrastructure's power usage to the running applications.

4.1 Infrastructure Model

The infrastructure model of LEAF is designed to meet the Requirements (1.1) and (1.2) set out in Section 1.1. Therefore, the infrastructure model must enable:

- the modeling of geographically distributed, heterogeneous, and resource-constrained compute nodes, interconnected with different kinds of wired and wireless network links, including WAN
- location-aware edge devices that may be mobile and can join or leave the network topology at any point in time

Moreover, we want the model to remain relatively abstract to avoid interfering with Requirement (4) on performance and scalability. In order to accomplish that, network and computing infrastructure are represented as a weighted, undirected graph

$$I = (N, L),$$

where

N is the set of compute nodes. A compute node $N^i \in N$ can have certain resource constraints such as maximum processing capacity and can consist of several hosts. Nodes can have a location and may be mobile. Hence, a compute node can represent a data center with various hosts, a fog node with limited computing capacity, as well as a mobile sensor without any processing capabilities.

L is the set of network links in the topology. A network link $L^i \in L$ can represent individual wired or wireless connections as well as entire network routes such as WAN connections, which include a multitude of shared networking equipment. Network links can be constrained by bandwidth and may have additional properties like a latency that users can utilize for implementing routing policies. Random effects like package loss have to be incorporated in the network link's parameters directly, for example, by decreasing the bandwidth while increasing the energy per bit.

Figure 4.1 depicts an exemplary infrastructure graph with a multitude of different compute nodes and network links. In the bottom left, several sen-

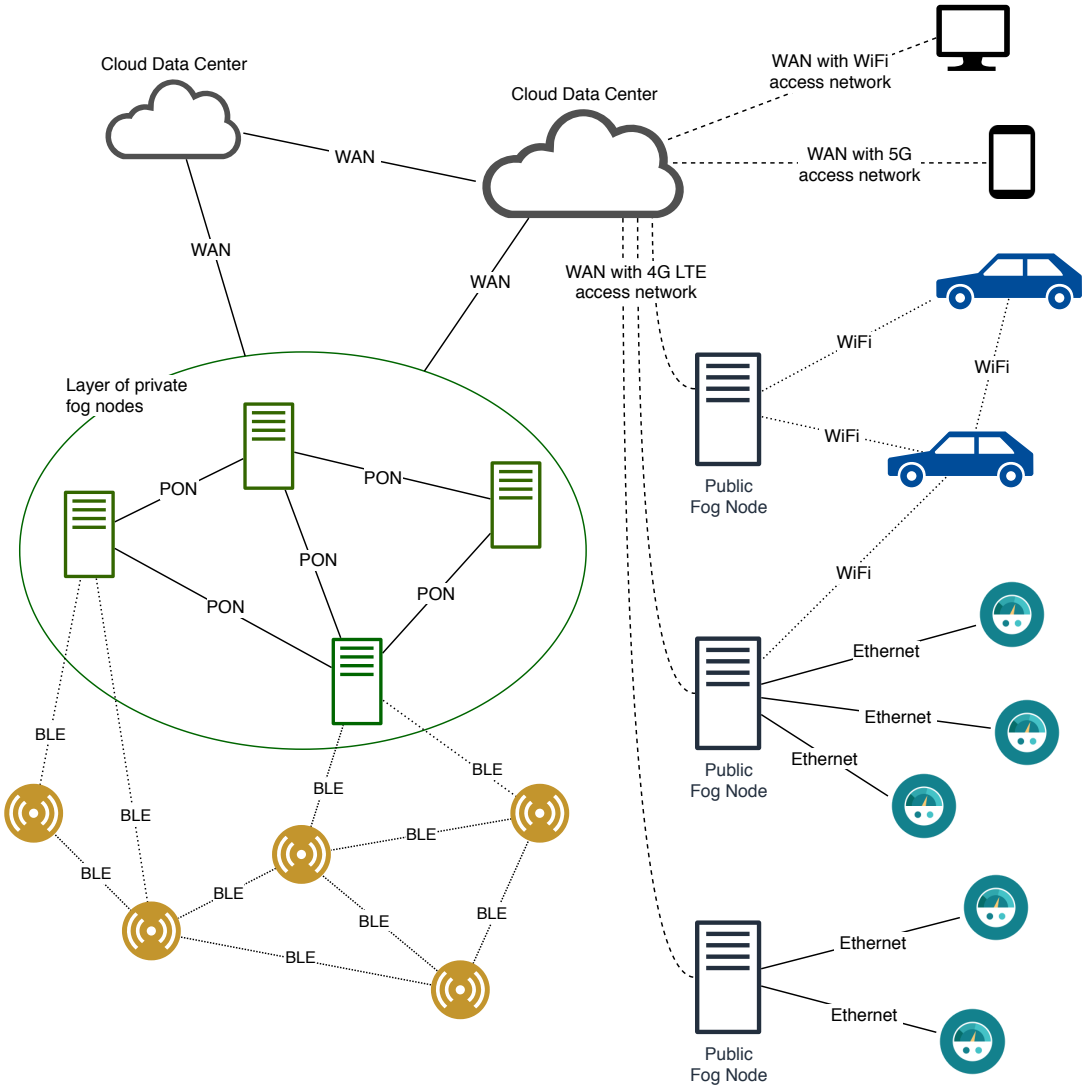

Figure 4.1: Complex infrastructure graph with various compute nodes and network links.

sors form a Bluetooth Low Energy (BLE) mesh network, which connects to a fog computing layer. The fog nodes in this layer are interconnected via passive optical network (PON) and can communicate with cloud nodes via WAN. The bottom right shows several smart meters that connect to public fog infrastructure via Ethernet. Furthermore, connected cars are linked to these fog nodes, as well as to each other for vehicle-to-vehicle (V2V) communication via WiFi. The public fog nodes connect the cloud via WAN with a 4G LTE access network. Lastly, in the top right, two devices are directly connected to the cloud via WAN using a WiFi and 5G access network.

Note that the access network, for example, wireless routers or mobile base stations, is usually not explicitly modeled in LEAF. Instead, the user should determine the overall bandwidth and power usage of the entire WAN link,

as shown in Figure 4.6. This reduces the complexity of the modeling process and computations.

An infrastructure graph can change during the simulation, as nodes are free to move around and join or leave the network. Furthermore, any property, like the latency between two nodes, can change. Some networking equipment and compute nodes, such as cloud data centers, may be shared among multiple tenants that are not all part of a simulation. This circumstance makes a difference when calculating power usage; however, the proposed infrastructure model does not need to distinguish between private and public infrastructure.

4.2 Application Model

Most simulators, especially the ones based on proper network DES, model network traffic in great detail. Some even implement entire networking protocols like TCP [26–28]. The proposed application model remains more abstract and regards all communication between different application tasks as continuous data flows. Consequently, LEAF considers all applications to be some kind of streaming applications. Similar to the infrastructure model, every application is expressed as a directed graph

$$A = (T, F),$$

where T is the set of tasks in the application, and F is the set of data flows between tasks. Graph modeling allows an abstract but realistic representation of different streaming applications; thus, the application model complies with Requirement (1.3) without interfering with Requirement (4). Tasks, as well as data flows, have certain resource requirements. Applications can be placed on the fog by mapping its tasks to compute nodes, and its data flows to network links. Task placements are described in more detail below.

Every task $T^i \in T$ requires certain computational resources from its host, denoted in MIPS. A compute node can host as many tasks as its resources allow. There exist three different kinds of tasks that are responsible for generating, processing, and storing or displaying data. In particular, it is possible to divide every set of tasks T into three distinct subsets:

Source tasks are bound to a specific compute node that is usually a sensor or mobile device. They are constantly emitting data and have at least one outgoing edge and no incoming edges.

Processing tasks can be freely placed on any compute node $N^i \in N$ that fulfills their resource requirements. Processing tasks always have at least one incoming edge and at least one outgoing edge.

Sink tasks are again bound to a specific compute node and have at least one incoming edge and no outgoing edges. Examples for data sink compute nodes are cloud storages or end-user terminals for data visualization.

A data flow $F^i \in F$ represents a continuous stream of data between two tasks and is stated in bits/s. The data rate can change during the simulation. A network link can "host" as many data streams as its bandwidth allows. However, LEAF is not suitable to model more complex effects resulting from network congestion like queuing delay or package drops.

When placing an application, first, each task has to be mapped onto a compute node. As stated above, the mapping of source and sink tasks is fixed; a task placement strategy determines the placement of processing tasks. Next, a routing policy searches for an optimal path in the network between every pair of connected tasks. If no path can be found that offers sufficient available bandwidth, the placement failed. Mathematically expressed, an application placement is a non-injective, non-surjective function that assigns each task $T^i \in T$ a compute node $N^i \in N$ and each data flow $F^i \in F$ a set of network links $L^i \in L$. Tasks and data flows allocate processing power and bandwidth on their hosts. The power model of the specific resource will base its calculations on this allocation, refer to Section 4.3.

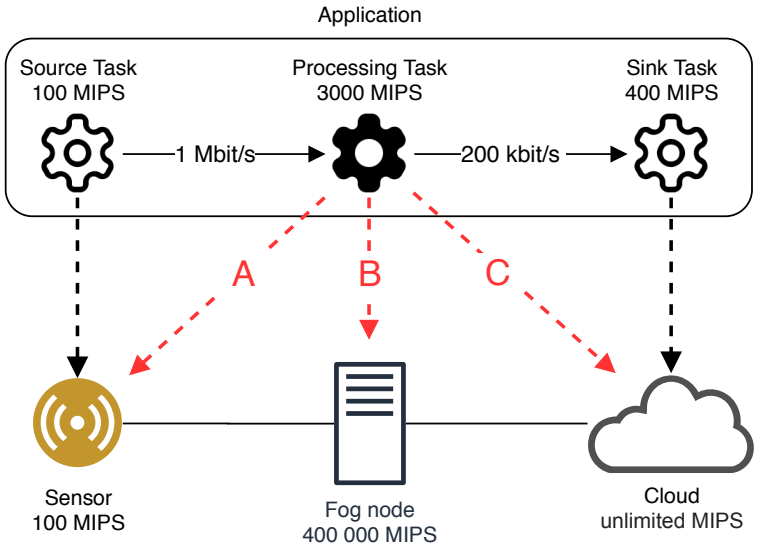

Figure 4.2: Different possibilities for placing a processing task.

Figure 4.2 illustrates an exemplary application placement with resource constrained compute nodes. The source task is running on a sensor and sends 1 Mbit/s to the processing task. The processing task requires 3000 MIPS and emits 200 kbit/s to the sink task, which, for example, stores the results in a cloud storage. A task placement strategy can now decide on a placement of the processing task, having three options:

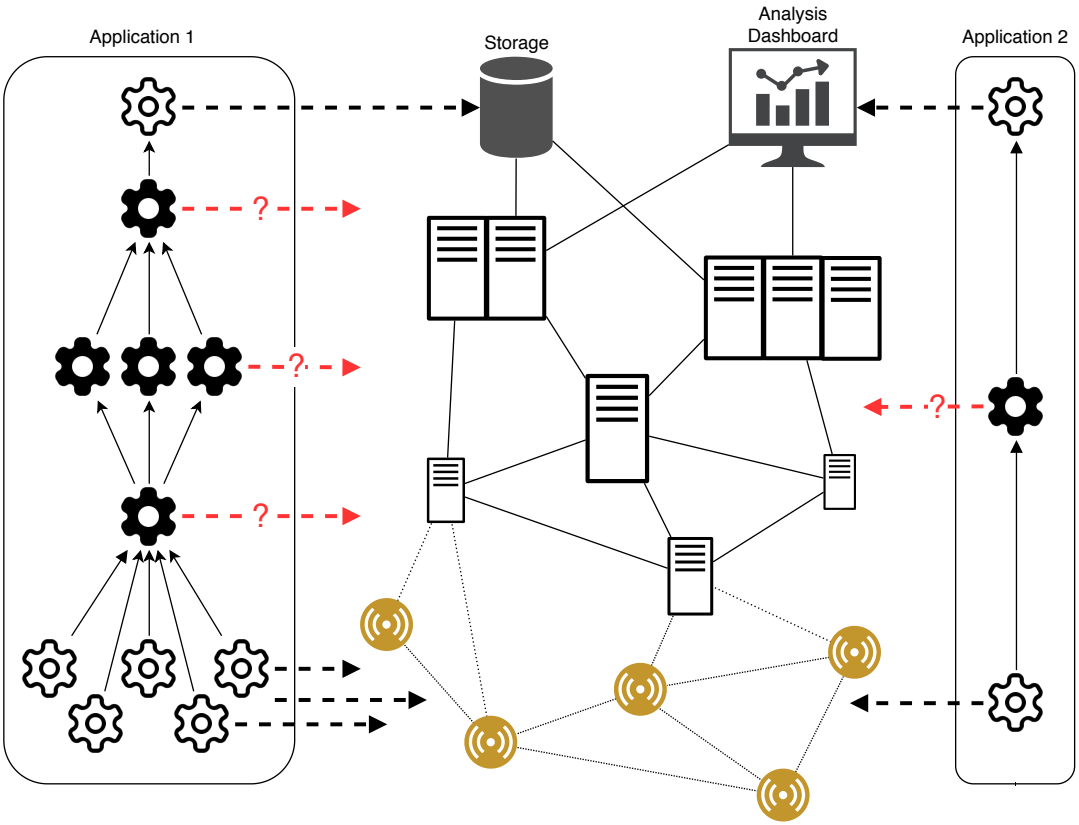

Figure 4.3: Complex application placement.

A Placing the task on the sensor: Since the sensor does not have the computational capacities to host a task that requires 3000 MIPS, this placement is not possible.

B Placing the task on the fog node: If the fog node still has enough remaining capacity, it can host the task. This placement effectively reduces the amount of data sent via WAN.

C Placing the task in the cloud: The cloud has unlimited processing capabilities in this example. Consequently, a placement is always possible.

Typically, processing tasks aggregate and filter the data stream in a way that outgoing data flows require less bandwidth than incoming data flows. Consequently, the placement of processing tasks close to the source tasks generally reduces network usage and, thus, energy consumption. Figure 4.3 shows a more complex example with two different kinds of applications. Finding optimal placements in dynamic, resource-constrained, distributed, and heterogeneous environments is a complicated problem and a focus of research [57]. LEAF enables application placement strategies that take the infrastructure's energy consumption into account.

4.3 Power Model

Building upon the infrastructure and application model, LEAF models power usage by assigning each compute node and network link an individual power model. Therefore it meets Requirements (2.1) - (2.3): LEAF can assess the power usage of all data centers, edge devices, and network links individually. A power model $P(t)$ is a function that returns the current power usage of the corresponding entity at any point in time t during the simulation. Hence, they are not only suitable for periodic measurements for subsequent analysis of power usage profiles, but can also be called at runtime by other simulated hardware or software components. Examples of such simulated entities are batteries that update their state of charge or energy-aware task placement strategies. Hence, LEAF supports online decision making based on power usage and complies with Requirement (3).

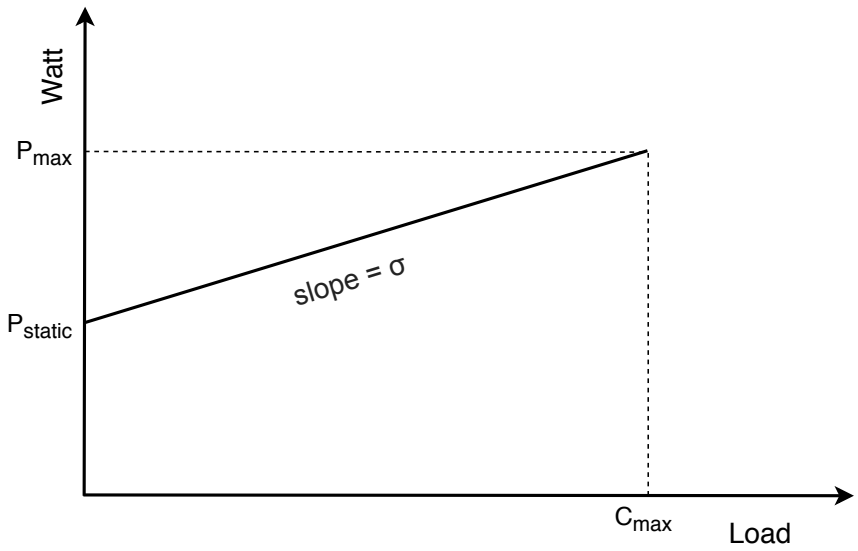

Figure 4.4: Linear power model. Adapted from Jalali *et al.* [6], Fig. 3.

In theory, a power model can be any mathematical function and may depend on multiple input parameters besides the load. However, as described in Section 2.3.1, the power usage of infrastructure can usually be modeled with sufficient accuracy as a linear function that is only based on the entity's current load $C(t)$ as depicted in Figure 4.4. This linear power model can be defined as

$$P(t) = P_{static} + C(t)\,\sigma, \tag{4.1}$$

where P_{static} is the static and $C(t)\,\sigma$ the dynamic fraction of the energy consumption. Section 4.4 explains ways to determine meaningful values for P_{static} and σ for different kinds of infrastructure. For network equipment,

σ represents the energy consumed per bit of transferred data. The unit of measurement is W/(bit/s), which can be also written as J/bit. For compute nodes σ is stated in W/MIPS, the the power required to perform one million operations per second. When modeling resource-constrained compute nodes or network links, σ can be expressed as:

$$\sigma = \frac{P_{max} - P_{static}}{C_{max}} \tag{4.2}$$

When modeling shared infrastructure, it serves no purpose to break down energy consumption into a static and dynamic fraction because other tenants and their applications also use these resources but are not included in the simulation scope. For instance, it is impossible to know at which point a data center will need to switch on or off a new physical machine, thereby changing static energy consumption. Additionally, it may not be possible to declare a reasonable P_{max} and C_{max}, as many resources, such as cloud nodes or WAN connections, can be assumed to scale at will. For these reasons, $P_{static} = 0$ for power models of shared infrastructure. Users must estimate a meaningful σ that directly incorporates a fraction of the static consumption, see Section 4.4.

Linear power models are desirable because they are easy to analyze and fast to compute. However, users can make use of additional input parameters and more complex functions if more precise modeling is required. An example of such a model is the wireless transmission model proposed by Heinzelman *et al.* [12]. The following presents a slightly adapted version of this model:

$$P_{Tx}(t, d) = P_{static} + C(t)\,\sigma + C(t)\,\gamma \cdot d^2$$

Equivalent to the linear power model, the transmitter has a static power usage and consumes a certain energy per bit σ for running the transmitter circuitry. Different from the linear model, the energy consumption of the transmit amplifier is modeled separately. It depends on the distance d between the devices. The incremental energy consumption of the transmit amplifier is γ ($\mathrm{J}/(\mathrm{bit} \cdot \mathrm{m}^2)$). We assume a quadratic energy loss over the distance, due to channel transmission.

Evidently, this model improves the accuracy of the simulation. Nonetheless, when simulating hundreds or thousands of nodes, more complex power models come at a cost. Calculating the distance between that many interconnected devices at each time step of the simulation can become computationally expensive. If the performance gets affected too much, an alternative approach, in this case, could be to estimate an average distance and incorporate the energy dissipation directly into the slope σ of the linear model.

Whether more complex power models than the linear power model should be used, strongly depends on the use case of the simulation. Computing the

distance between two points is still a comparably cheap operation, but in general, one must keep in mind that every additional input parameter needs to be modeled and assessed. When the goal is to simulate the energy consumption of battery-constrained sensors in a wireless mesh network, it makes sense to opt for a precise power model. The exact lifetime of individual sensors has a significant impact on the simulation and should be approximated as accurately as possible. However, when the objective is to compare the overall energy consumption of WiFi versus WAN traffic, it may be appropriate to use well parameterized linear power models only. The estimation of WAN parameters inevitably involves uncertainties, see Section 4.4.5. For this reason, it is pointless to model other parts of the network with a high degree of accuracy and thereby sacrifice performance.

4.4 Parameterizing Linear Power Models

The parameterization of more complex power models is often straightforward since the consumption got broken down into well-defined input parameters. Nevertheless, as explained in the previous section, it is often preferable to describe power usage as linear functions in order to maintain simplicity and scalability. Because of that, this section presents several approaches for parameterizing linear power models for different kinds of compute nodes and network links.

Section 4.4.1 explains how to determine meaningful parameters for compute nodes. Section 4.4.2 lines out what needs to be considered when modeling shared compute nodes. Section 4.4.3 covers network link parameterization in general, while Section 4.4.4 focuses on wireless network links in particular. Section 4.4.5 breaks down a WAN connection into its components and, based on this, proposes an exemplary parameterization for WAN links with a 4G LTE access network.

4.4.1 Compute Nodes

This section covers the power model parameterization of compute nodes, which are used exclusively by the simulated applications. Examples are sensors, mobile devices, or private fog data centers that are not shared with other parties. The deployed hardware in these nodes and their energy consumption are known to the user.

Parameters for these kinds of compute nodes can be easily determined by observing and measuring the power usage of real hardware. As stated in Section 2.3.1, most systems show a clear correlation between CPU load and memory and storage access. Therefore, it is reasonable to scale the dynamic

power usage only based on the CPU load. P_{static} and P_{max} can be assessed via measurements: An exemplary host may require 20 W being idle and 140 W under full load. The maximum capacity of the CPU is 83.000 MIPS. Used in Equations 4.1 and 4.2 this results in:

$$P_{host}(t) = 20\,\text{W} + C(t)\frac{140\,\text{W} - 20\,\text{W}}{83\,000\,\text{MIPS}}$$

For example, performing $C(t) = 30\,000$ MIPS this host requires about 63.4 W.

More complex compute nodes such as data centers have many hosts and require additional power for overhead like cooling and lighting. Assuming that all hosts in the data center are always in operation, the simplest approach for modeling this kind of infrastructure is to aggregate the current power usage of all underlying hosts H. To account for the operational overhead, we multiply the result with the data center's PUE, see Section 2.3.3:

$$P_{dc}(t) = \text{PUE} \cdot \sum_{H_i \in H} P_{H_i}(t)$$

In reality, most data centers try to consolidate workload on a few hosts and shut down others when they are idle to achieve a more energy-proportional power profile. Modeling this kind of energy-saving mechanism is explained in detail in Section 4.5.

4.4.2 Shared Compute Nodes

A large number of nodes, such as cloud data centers, are shared among different tenants and often assumed to be virtually infinitely scalable. As explained in Section 4.3, there is no way to know how many data center hosts are currently active, since the other tenants are outside the scope of the simulation. Consequently, for shared infrastructure power models, we define $P_{static} = 0$. A meaningful value for σ has to be determined that takes into account static power usage. Thereby we implicitly assume shared infrastructure to be energy-proportional. Nevertheless, there are ways to incorporate inefficiencies into the parameters, as shown below.

The following presents a possible method for estimating σ_{cloud}, the required power per MIPS of a shared cloud data center. First, σ_{rack} of a single data center rack is determined. We assume that an average rack draws 8 kW under full load and has a maximum capacity of 16 000 000 MIPS. As mentioned before, we assume the rack to be power-proportional. Used in Equation 4.2, this results in:

$$\sigma_{rack} = \frac{8\,\text{kW}}{16\,000\,000\,\text{MIPS}} = 500\,\mu\text{W/MIPS}$$

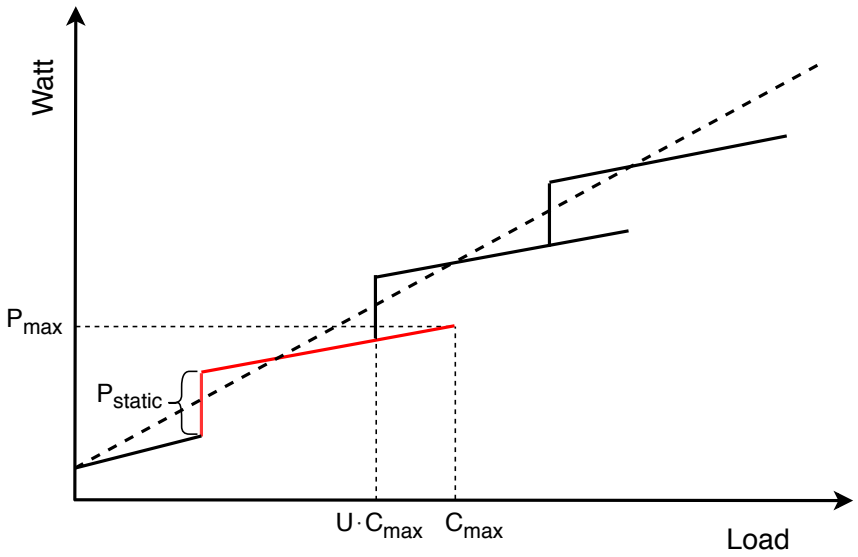

Figure 4.5: Incremental performance per watt of shared, scalable data centers: The red part visualizes a single host in the data center, which is only utilized until $U \cdot C_{max}$ of its full capacity before the next host gets activated. Adapted from Jalali *et al.* [6], Fig. 4.

One possible approach would be to simply set $\sigma_{cloud} = \sigma_{rack}$. In practice, however, servers are rarely used to full capacity but are slightly overprovisioned to ensure performance and availability. Users need to estimate at which utilization the cloud provider is going to scale up the infrastructure. Equivalently to the energy model proposed by Jalali *et al.* [6] this can be modeled by the operational capacity fraction U. In our example, U shall be 0.9. The effect of U is illustrated in the staircase curve depicted in Figure 4.5.

Furthermore, the data center's PUE must be incorporated. In this example, we assume a relatively energy-efficient data center with a PUE of 1.3. By taking the overprovisioning and PUE into account, the overall energy efficiency of the cloud data center can be calculated:

$$\sigma_{cloud} = \sigma_{rack} \cdot \frac{1}{U} \cdot \text{PUE} = 722.22\,\mu\text{W/MIPS}$$

4.4.3 Network Links

For network link power models, σ is stated in J/bit and represents the incremental energy per bit transferred. Although one could technically define network links with static power usage, it is not meaningful from a modeling perspective. Static power consumption should instead be directly attributed to the P_{static} of the responsible, adjacent compute nodes. This has the ad-

vantage that it correctly drops to zero when the node is being put to sleep, see Section 4.5. Intermediate routing devices should be considered shared infrastructure and accordingly have no static power usage. Summarizing, we define $P_{static} = 0$ for all network link power models.

It is important to understand that a network link, in reality, does not consume any power by itself. It is always the networking equipment, such as routers, switches, or wireless transmitters, that consumes energy. Therefore, the direct mapping of power models to edges in the infrastructure graph is an abstraction LEAF makes to avoid having to model all the individual components of a network. However, when parameterizing network links, these components should indeed be lined out and taken into account to ensure realistic simulations. The σ of network links is always the sum of the σ of each network equipment involved.

There exist a variety of studies and models for the power usage of wired and wireless networks with partly varying results [10, 45, 46, 58–60]. As shown below, it is not possible to define generally applicable parameterizations for different network types. This section is intended to provide some examples and reference points. The exact parameterization always depends on the goal and context of the simulation.

The first example covers a simple Ethernet adapter. Ethernet is a comparably power-hungry protocol since it was initially not designed around energy efficiency. Wired devices are generally expected to have power available nearby. The `ENC28J60` is a common Ethernet controller which operates at around 3.3 V and requires 180 mA under load and 120 mA being idle[1]. Following, its P_{static} is around 400 mW, while P_{max} is around 600 mW. The maximum throughput of a `ENC28J60` is 10 Mbit/s. Used in Equation 4.2 this results in:

$$\sigma_{ENC28J60} = \frac{600\,\text{mW} - 400\,\text{mW}}{10\,\text{Mbit/s}} = 20\,\mu\text{J/bit}$$

Needless to say, such small Ethernet adapters are comparably inefficient. Modern hardware deployed at internet service providers is indeed optimized for energy-efficiency. A Cisco CRS-3 router can manage throughputs of 4480 Gbit/s at a maximum power usage of 12.3 kW. The overall σ of this router - including static power usage - is 2.75 nJ/bit. This example is intended to show that energy usage of wired connections that use the same protocol can vary by several orders of magnitude and strongly depends on the deployed hardware and cable type. For parameterizing such power models, much attention must be paid to configuration and structure of the network. Besides the hardware, also throughput, package size, and bit error rate can have an influence on required power.

[1] `http://ww1.microchip.com/downloads/en/DeviceDoc/39662e.pdf`, accessed 2020-04-28

As stated above, the σ of a network link is always the sum of all network equipment involved. Thus, if we assume that two compute nodes are directly connected, and both use this Ethernet adapter, their network link consumes

$$\sigma_{ENC28J60 \leftrightarrow ENC28J60} = 40\,\mu\text{J/bit}.$$

The static consumption of 400 mW per adapter should be attributed to the adjacent compute nodes.

4.4.4 Wireless Network Links

Much like wired connections, the energy-effectiveness of wireless connections strongly depends on the used hardware. To obtain a precise σ, one has to sum up the required energy per bit for receiving and transmitting data of both communicating devices. Further determining factors in wireless power consumption are the distance between devices and the characteristics of the environment [61]. On short distances and with a free line of sight, a transmitter has to use way less energy to send packets than over larger distances or with elements absorbing or reflecting the radio waves. This is a general characteristic of wireless networks: Sending packages is more costly than receiving because the transmitter has to account for the energy dissipation.

However, LEAF does currently not allow the modeling of asymmetrical bandwidth and power consumption characteristics. Since edges in the infrastructure graph are undirected, network connections are expected to provide the same bandwidth and consume the same power for uplink and downlink - although, in reality, this is rarely the case. To obtain valid results, the user must determine how the network is mostly used in the simulated scenario and choose an adequate parameter. For example, if the mobile base stations in a simulation solely upload data to the cloud, the user should pick a σ, which represents a realistic uplink consumption. A future version of LEAF can remedy this deficiency by not utilizing an undirected graph as the basis for the infrastructure model but a directed multigraph. This way, uplink and downlink can be modeled separately. Moreover, it would become possible to have multiple different connections between two compute nodes. This would additionally enable energy-aware routing algorithms that, for example, could decide whether it is better so send data via WiFi or Bluetooth.

The following provides a few examples of wireless power model parameterizations. Huang *et al.* [62] propose a detailed model for 4G LTE power characteristics and compare it to 3G and WiFi. Table 4.1 shows the incremental energy per bit for transmitting σ_{Tx} and receiving σ_{Rx} data, which they determined by performing measurements on a mobile phone. We can see that

Table 4.1: Energy per bit for transmitting and receiving data on a mobile phone according to Huang *et al.* [62].

Type	σ_{Tx} (nJ/bit)	σ_{Rx} (nJ/bit)	σ_{Tx}/σ_{Rx}
4G LTE	438.39	51.97	8.44
3G	868.98	122.12	7.12
WiFI	283.17	137.01	2.07

the difference between sending and receiving is especially large on mobile communication technology.

Besides the mobile phone, one also needs to model the counterpart of the wireless connection in order to ascertain a σ for the network link. For WiFi access points, the hardware specifications of network equipment are known in most scenarios. An enterprise wireless access point like the Cisco Aironet 1572 consumes 25 W idle, 31 W under full load[2], and provides up to 1.3 Gbit/s. This kind of access point is deployed to cover large areas with many devices; hence, we consider it shared infrastructure. However, WiFi is not bidirectional like Ethernet, so the access point cannot provide 1.3 Gbit/s uplink and downlink at the same time. Again, modeling this circumstance is not possible because of the earlier mentioned limitations of LEAF. Nevertheless, it can be assumed that the access point draws the maximum of 31 W when transmitting at full load since this is the more costly operation. Thus, the incremental energy per bit for transmitting is:

$$\sigma_{Tx\ Aironet1572} = \frac{31\,\mathrm{W}}{1.3\,\mathrm{Gbit/s}} = 23.8\,\mathrm{nJ/bit}$$

In order to determine a meaningful σ_{Rx} for this access point, further measurements are necessary. However, based on the results presented so far, the transmission of data from an Aironet to a mobile phone can already be parameterized:

$$\sigma_{Aironet1572 \to Phone} = 23.8\,\mathrm{nJ/bit} + 137.01\,\mathrm{nJ/bit} = 160.81\,\mathrm{nJ/bit}$$

Determining parameter values for mobile base stations is even more challenging. First of all, there exist a variety of different types and sizes of cell sites. Macrocells are usually mounted on rooftops or hills and cover areas of 1 to 20km. Microcells have a range of less than 2km and are mostly used in areas with dense phone usage, such as shopping malls or train stations.

[2] `https://www.cisco.com/c/en/us/td/docs/wireless/access_point/1570/installation/guide/1570hig/1570_axa.html`, accessed 2020-05-09

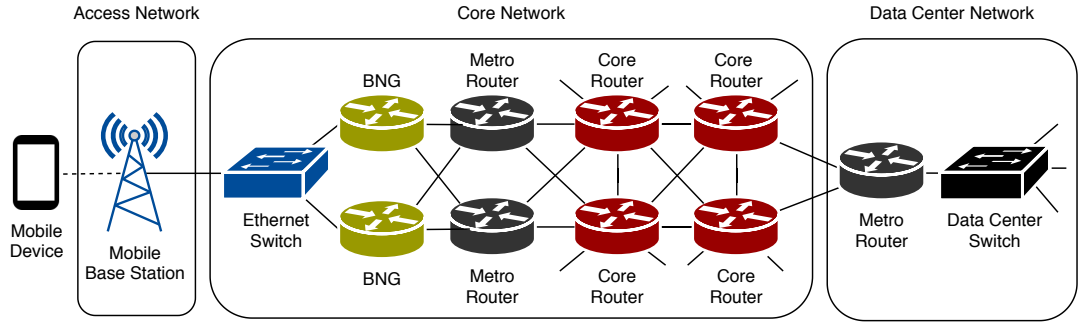

Figure 4.6: Topology of WAN between mobile device and cloud data centre. Adapted from Vishwanath *et al.* [59], Fig. 1.

Picocells cover less than 200m and are usually deployed indoors. Lastly, femtocells operate on the order of 10 meters and can, for example, be used to provide mobile network to offices with poor network coverage. Each of these mobile base station types has a different power profile [63]. Additionally, Abdulkafi *et al.* [61] showed that the transmit power of cell sites highly depends on the distance of communicating devices and the characteristics of the environment. Furthermore, for precise results, one would not only need to model the transmit power of base stations but also how it degrades over the distance since mobile devices usually connect to the base station with the strongest signal.

To summarize, if no further information is available on the type and location of base stations or if different kinds of base stations are deployed, it is tough to provide reasonable estimates on their consumption. If possible, the parameterization should be based on measurement data collected in the real existing environment. In fully conceived scenarios, the consumption of base stations remains a significant factor of uncertainty.

4.4.5 Wide Area Network

The principal challenge of modeling WAN power usage is that many determining variables are unknown and not under the user's control. There is no way to know how many routers a package passes through the network or how energy-efficient the hardware at internet exchange points operates. For highly shared, third-party infrastructure, estimates are becoming increasingly vague because there is only little official data available.

Nevertheless, one can make elaborate estimates on the energy consumption of networking equipment along the path of the data through the WAN. To provide an example, we will look at a WAN connection between a mobile phone connected via 4G LTE and a cloud data center. The topology of such

a connection was described by Vishwanath *et al.* [59] and is depicted in Figure 4.6. We group the networking equipment in four categories, which are outlined in detail below, and describe the overall incremental energy per bit as:

$$\sigma_{WAN} = \sigma_{device} + \sigma_{access} + \sigma_{core} + \sigma_{dc} \tag{4.3}$$

First we determine σ_{device}, namely how much power the mobile phone requires to send data to the access network, in this case the next mobile base station. We use the findings of Huang *et al.* [62], who state the energy per bit consumed on a mobile device for sending data via 4G LTE to be:

$$\sigma_{device} = 438.4\,\text{nJ/bit}$$

Next, the data has to travel through the access network, which is connecting end devices to their immediate service provider. In this example, the phone is connected to a mobile base station via 4G LTE. Vishwanath *et al.* [59] derived the incremental energy per bit for uplink of a 3-sector 10-MHz 2×2 MIMO LTE base station as 6200 nJ/bit. As explained in the previous section, the actual consumption of base stations depends on a variety of factors, so this is just an exemplary value. We define the upstream consumption of the access network as:

$$\sigma_{access} = 6200\,\text{nJ/bit}$$

Following, the data travels through the core network. In the described example, this includes the metropolitan area network, as well as the actual core network that builds the backbone of the internet and connects the different internet service providers. Networking equipment in the metropolitan and core network is usually interconnected with optical fiber. The data passes several switches, a broadband network gateway (BNG), and different kinds of routers on its way from the access network to the data center.

According to the topology depicted in Figure 4.6, for our example, we assume a single Ethernet switch, one BNG, one metro router, and five core routers on the way through the WAN. Table 4.2 depicts the energy per bit consumed by different network devices that are representative of these categories. The power usage has been determined via the Cisco Power Calculator[3]. Analogous to the shared compute nodes, we expect the shared networking equipment to not be utilized at 100 % and define a operational capacity fraction $U = 0.9$. The overall energy per bit for the core network is:

$$\sigma_{core} = \frac{1}{U}\left(\sigma_{es} + \sigma_{bng} + \sigma_{mr} + 5\,\sigma_{cr}\right) = 38.2\,\text{nJ/bit}$$

[3] `http://tools.cisco.com/cpc`, accessed 2020-05-17

Table 4.2: Power consumption of Cisco networking equipment.

Type	Model	Max capacity	Max power	Energy per bit
Core router	CRS-3	4480 Gbit/s	12300 W	$\sigma_{cr} = 2.7\,\text{nJ/bit}$
Metro router	7603	560 Gbit/s	4550 W	$\sigma_{mr} = 8.1\,\text{nJ/bit}$
BNG	ASR 9010	320 Gbit/s	1890 W	$\sigma_{bng} = 5.9\,\text{nJ/bit}$
Ethernet switch	Catalyst 6509	256 Gbit/s	1766 W	$\sigma_{es} = 6.9\,\text{nJ/bit}$
Data center switch	Nexus 9500	102.4 Tbit/s	15954 W	$\sigma_{dcs} = 0.2\,\text{nJ/bit}$

Last, we need to determine the power usage of the data center network - the processing of data in the servers is already modeled in the compute nodes. We assume one more router of the size of a metro router and two data center switches. Again, all equipment is utilized at 90 % of its full capacity.

$$\sigma_{dc} = \frac{1}{U}\left(\sigma_{mr} + 2\,\sigma_{dcs}\right) = 8.5\,\text{nJ/bit}$$

Used in Equation 4.3 we calculate the total energy per bit for streaming data from the mobile device to the data center via 4G LTE:

$$\begin{aligned}\sigma_{WAN} &= \sigma_{device} + \sigma_{access} + \sigma_{core} + \sigma_{dc} \\ &= (438.4 + 6200.0 + 38.2 + 8.5)\,\text{nJ/bit} \\ &= 6685.1\,\text{nJ/bit}\end{aligned}$$

To illustrate, for every watt-hour, the mobile phone can transfer 67.31 MB of data to the cloud. Interestingly, the core and data center network consumption only accounts for a minor fraction of the overall consumption. Therefore, many unknown factors like the number of package hops or the deployed networking equipment do not have a substantial impact on the result. In contrast, the access network contributes significantly to WAN power consumption. This is in line with the results from previous research [6, 58, 59].

4.5 Energy-saving Mechanisms

Energy-saving mechanisms aim to reduce static power usage of hardware during periods of low utilization by throttling or powering off parts of the system. They can significantly influence overall energy consumption by pushing infrastructure towards operating in an energy-proportional way. Consequently, they can only be applied to resources that are not fully uti-

lized. This section explains two different kinds of energy-saving mechanisms and how they can be represented in LEAF.

The first type of energy-saving mechanism is targeted at reducing static energy consumption of individual system components. An example of this are techniques related to CPU throttling like dynamic voltage and frequency scaling (DVFS). As the name suggests, DVFS dynamically adjusts voltage and frequency of a microprocessor depending on its actual load. This reduces static energy consumption during low utilization and, hence, conserves power. DVFS is widely used in mobile device processors but also has a noticeable impact in data centers as it decreases the need for cooling and, as a side effect, improves the data center's PUE.

Figure 4.7 displays two simple ways to model DVFS in LEAF. In this example, the CPU reduces frequency and therefore static power usage once its utilization drops below 80 % and 20 %. The blue line illustrates a simple approach where P_{static} and σ of the original linear power model were adapted to display a more power-proportional usage profile. This is simplistic but performant and suitable for large-scale experiments. The red line represents a non-linear, but more accurate power model where the "static" energy consumption depends on the CPU's utilization $C(t)$, for example:

$$P_{static} = \begin{cases} 20\,\text{W}, & \text{if } C(t) >= 0.8\,C_{max} \\ 15\,\text{W}, & \text{if } 0.8\,C_{max} > C(t) >= 0.2\,C_{max} \\ 10\,\text{W}, & \text{otherwise} \end{cases}$$

Nevertheless, also the red power model is not ideal. In reality, the Linux kernel offers five different modes that define under which circumstances the frequency gets changed - frequencies are usually not switched the same moment utilization drops below a certain threshold. Guérout *et al.* [48] presented a precise simulation as well as an analytical model for DVFS [48]. However, their model depends on a multitude of different parameters that the simulator has to assess continuously, which is computationally expensive when there are many CPUs to simulate.

The second type of energy-saving mechanisms dynamically shuts down idle resources entirely, also called putting them "to sleep". This approach is widely adopted for servers within cloud data centers and shows dramatic improvements in energy efficiency. For instance, Meisner *et al.* state that their energy-conservation approach PowerNap can reduce data center power consumption by 74 % [64]. Gupta & Singh show that for traffic loads for up to 5 %, network equipment can be powered off 60 % to 80 % of the time to conserve energy [65]. Fog computing could highly profit from these approaches by putting fog nodes to sleep during periods of reduced load.

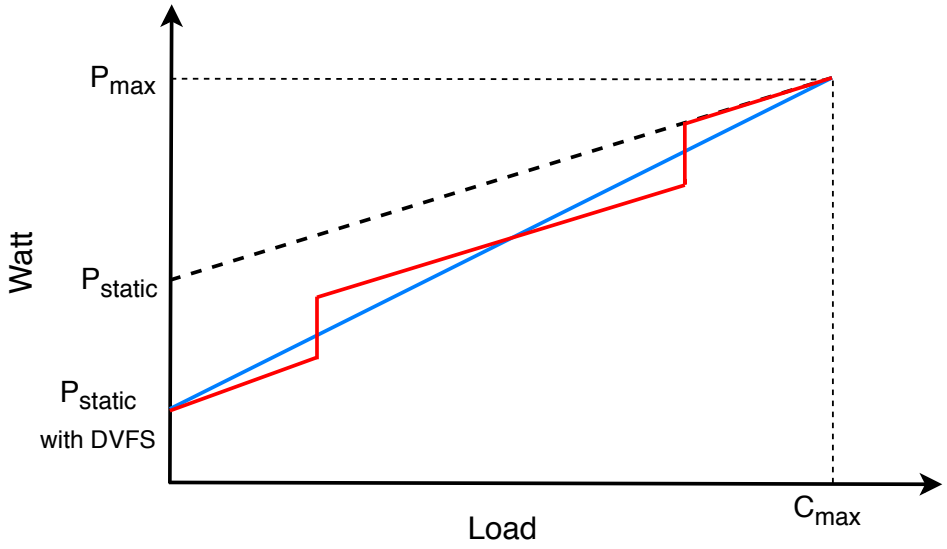

Figure 4.7: Exemplary CPU power models: The black, dashed line represents a CPU without DVFS. The red line represents a non-linear DVFS power model comprising frequency changes at certain loads. The blue line represents a more abstract, linear DVFS power model.

Incorporating these techniques into the fog environment simulation is of varying difficulty. Shutting down hosts within compute nodes can be modeled by tracking how long they are idle and marking them as powered-off after some time. For powered-off hosts, power models return $P(t) = 0$. It is significantly more complicated to switch off entire compute nodes or network links in the simulation because this changes the network's topology. Since the proposed model is relatively abstract and does not model individual switches or routers, it is not suitable for modeling the dynamic shutdown of single network devices.

To make full use of energy-saving mechanisms, users should adapt their application placement strategies and routing policies. For example, workload can be consolidated at a minimum number of nodes to maximize the number of idle nodes that can be powered off. Among many established consolidation algorithms for cloud computing [54, 66, 67] there are already some approaches targeted at fog computing [57]. A smart consolidation of correlated VMs can also highly reduce network traffic and conserve energy.

4.6 Tracing Back Power Usage to Applications

When implementing energy-aware application placement strategies, it is helpful to know not only the infrastructure's power usage but also its cause, namely the relative power requirements of applications. For example, an

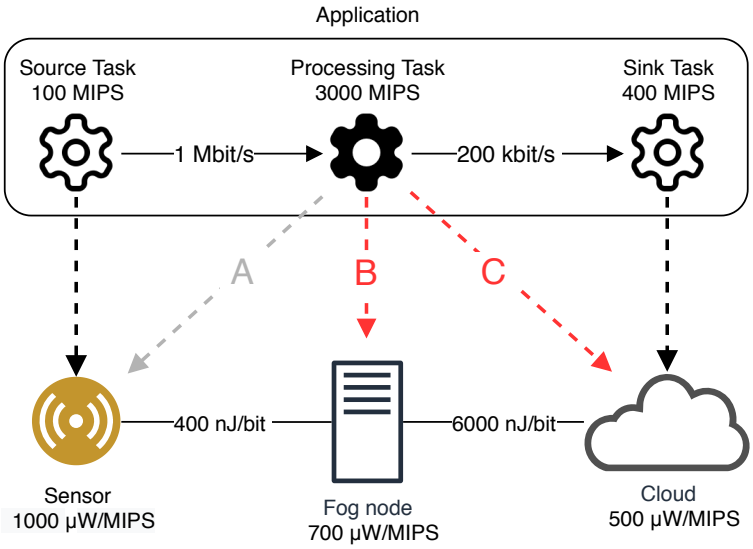

Figure 4.8: Extension of Figure 4.2 with power usage characteristics of compute nodes and network links.

application running on a server all by itself is responsible for a lot of static energy consumption since it is the sole reason this server is up and running. The application's power requirements will improve by migrating it to another server that is already under load. To support the implementation of application placement strategies that intend to reduce inefficiencies and conserve energy, LEAF meets Requirement (2.4), tracing back the infrastructure's power usage to the responsible applications.

To illustrate, Figure 4.8 takes up the example of Figure 4.2. In this example there is no static power consumption. In Section 4.2 it has been determined that Placement A is impossible because the resource-constraint sensor cannot host the processing task. Consequently, an energy-aware task placement strategy has to decide between Placement B and Placement C in this example.

In LEAF, calculating the dynamic power consumption of applications is straightforward. It is the sum of all dynamic power usage that its tasks and data flows caused on their allocated resources. The derivation of the relative application power usage for both placements is shown in Table 4.3. Although the processing task consumes more power when placed in the cloud, the overall power usage of Placement B is only half of Placement C. This is, because streaming 1 Mbit/s from the source task to the sink task, namely from the sensor to the cloud, is very power-intensive.

LEAF's approach towards attributing static energy consumption is as follows: For every task, it computes the fraction of how much resources the task attributes on a compute node in relation to the overall reserved capacity. Note that the overall available capacity of the compute node does not play any role.

Table 4.3: Application power usage for Placement B and C.

	Placement B	**Placement C**
Source Task	0.1 W	0.1 W
Processing Task	2.1 W	1.5 W
Sink Task	0.2 W	0.2 W
Source Task → Processing Task	0.4 W	6.4 W
Processing Task → Sink Task	1.2 W	0 W
Total	4 W	8.2 W

For example, if a task allocates 1000 MIPS on a node that was idle before, all static energy consumption gets attributed to the application. It does not matter if the overall capacity of the node is 10 000 MIPS or 500 000 MIPS. This is meaningful because the placement is highly inefficient: If the task was placed somewhere else, the compute node would be idle and could be turned off. However, if the task is placed on a node that already performs 50 000 MIPS, the application would only be attributed 1/6 of its static consumption.

5 Evaluation

To evaluate LEAF and present its capabilities, different architectures and task placement strategies were simulated in a smart city scenario. The results of these experiments are analyzed and discussed in this chapter. Note, that the evaluation is not based on real measurement data - the simulated scenarios are designed to be realistic but simple. Thus, the objective of this evaluation is not to provide exact results on how to design future fog environments. It rather aims to demonstrate how researchers can use LEAF to effectively analyze and improve the overall energy consumption of fog computing environments. Furthermore, it will be shown that the model fulfils the requirements denoted in Section 1.1.

Section 5.1 presents the prototypical implementation of LEAF. Section 5.2 describes the experimental setup of the evaluated scenarios. Section 5.3 explains, analyzes and discusses the results of the conducted experiments. Section 5.4 takes a closer look at the performance and scalability of the presented simulator. Last, Section 5.5 confirms that the proposed simulator meets all requirements denoted in Section 1.1.

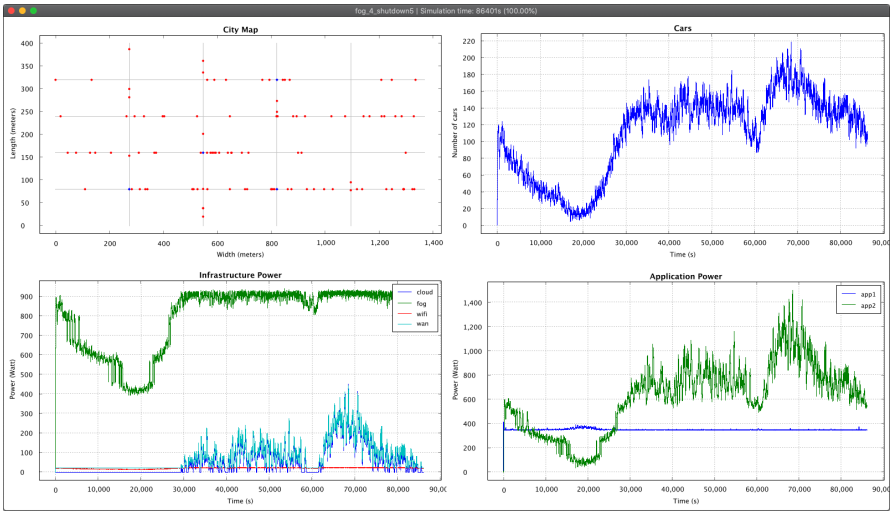

Figure 5.1: Live visualization of the prototype: (1) A map of the city, (2) the number of taxis on the map, (3) the power usage of infrastructure parts and (4) the power usage of applications over time.

5.1 Prototype

A prototype of LEAF was implemented in Java based on the cloud computing simulator CloudSim Plus [33] and the graph library JGraphT[1]. All code is available on GitHub[2]. Several contributions were made in three abstraction layers: The CloudSim Plus core, a package that implements the proposed infrastructure and application model and a package that contains the code used for this evaluation. This section provides some further information about the contributions made.

Multiple modules were added or replaced in the CloudSim Plus core. These changes are not directly related to simulating fog environments but are general improvements to the library. In particular, the CloudSim Plus core was improved in the following ways:

- The existing network topology was highly coupled to BRITE topology files, had an unintuitive interface and a $\mathcal{O}(n^3)$ worst-case complexity for finding shortest paths, implementing only the Floyd–Warshall algorithm. It was replaced with a more flexible interface which can be implemented by users using libraries such as JGraphT to make use of the many features and algorithms implemented in this library.
- The existing power models were only suitable for use with data center hosts. They were replaced with a general power model that can be used for hosts and network links but is entirely decoupled from their implementation.

The LEAF project itself depends on this patched CloudSim Plus core and contains two packages: First, the *org.leaf* package contains the proposed infrastructure and application model as well as the related power models. The infrastructure and application graphs were implemented on top of JGraphT. Second, the *org.cityexperiment* package implements all functionality required in the evaluation. It contains the city scenario, mobility model, compute nodes, network link types and the applications presented in Section 5.2. Additionally a live visualization has been implemented with XChart[3] in order to provide insights on running experiments. The visualization updates itself according to a user-defined frequency and contains four charts: 1. A map of the city and the moving taxis, 2. the number of taxis that were present on the map at a given time, 3. the infrastructure power usage over time and 4. the application power usage over time. An example screenshot can be seen in Figure 5.1. Each experiment run outputs two CSV files containing the measurements. The analysis of the results was performed in Python.

[1] `https://jgrapht.org/`, accessed 2020-05-07
[2] `https://github.com/birnbaum/LEAF`, accessed 2020-05-29
[3] `https://github.com/knowm/XChart`, accessed 2020-04-28

5.2 Experimental Setup

This section describes the experimental fog computing environment on which the evaluation was carried out. Section 5.2.1 describes the structure of the simulated smart city scenario and how taxi traffic was modeled. Section 5.2.2 presents the infrastructure topology and applications that were modeled in the experiments. Section 5.2.3 explains which parts of the topology get analyzed regarding their energy consumption, and it covers the parameterization of the infrastructure, application, and power models.

5.2.1 Smart City Scenario

The evaluation is based on an intelligent transportation system scenario located in a smart city center. The urban structure is inspired by Manhattan, namely a rectangular street grid which each block measuring 274 m by 80 m. The simulation covers an area containing 16 crossings, each equipped with a smart traffic light system that optionally has a fog node attached. Figure 5.2 shows an exemplary map of the city. Two taxis are on the map, connected to nearby traffic lights, and two traffic lights have fog nodes attached. Note that this is just an example; the number of taxis and fog nodes varies in the experiments.

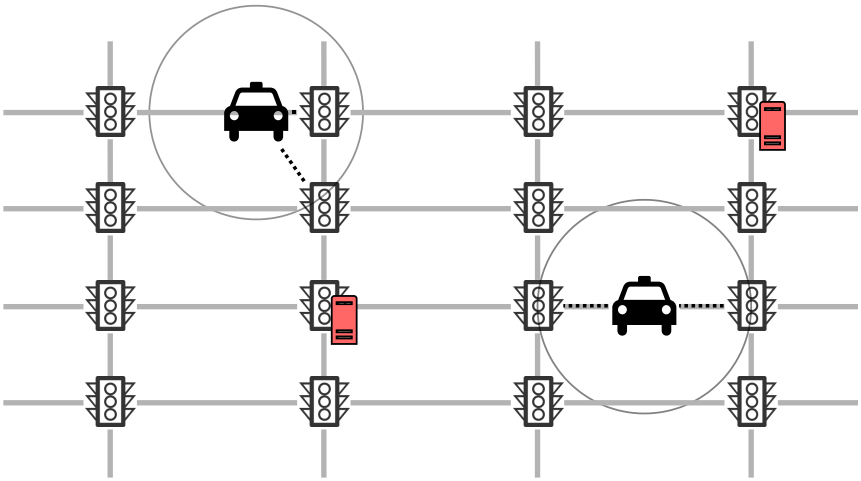

Figure 5.2: Map of the simulated city center with two taxis and two fog nodes.

All experiments simulate the traffic of taxis within this city center over a period of 24 hours. In order to obtain a realistic environment, the quantity and speed of taxis was modeled according to a dataset published for the 2015 DEBS Grand Challenge competition[4] which contains information on approximately 173 million taxi trips that took place in New York in 2013.

[4] `http://www.debs2015.org/call-grand-challenge.html`, accessed 2020-05-05

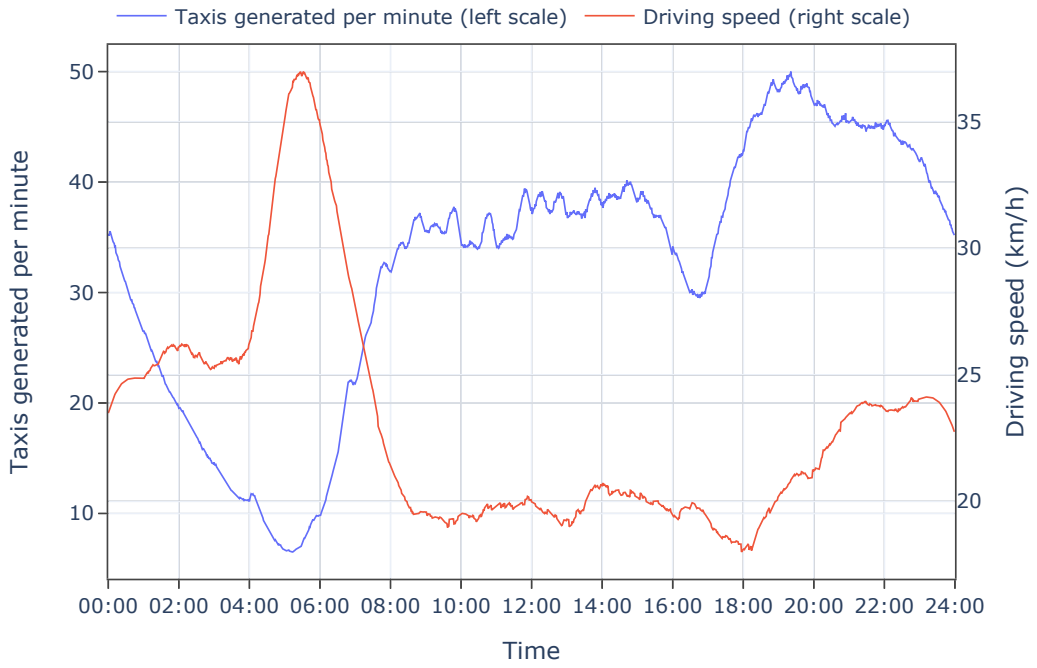

Figure 5.3: Number of taxis generated per minute vs. average driving speed.

First, all samples were grouped, based on the time of the day the passengers were picked up. The time window is one minute; hence, there are $60 \cdot 24 = 1440$ groups. Based on the frequency distribution, the simulator generates new taxis by periodically calculating the current probability of arriving cars following a Poisson distribution. Figure 5.3 shows the average number of taxis generated per minute.

We can observe that during early morning hours, only around five taxis enter the map every minute, while at the evening rush hour, it is up to 50. The taxis arrive on one of the edges of the map and follow a random path to another edge where they leave. The speed of the taxis gets determined via the 2015 DEBS Grand Challenge data, too: The median of distance and duration of every group was used to calculate the average speed at a given day time, likewise depicted in Figure 5.3. At night, taxis can drive at more than 30 km/h; during day time, they do not even reach 15 km/h.

5.2.2 Network Topology and Applications

In the simulated scenario, the infrastructure graph consists of four different kinds of compute nodes: The cloud, fog nodes, traffic light systems and taxis.

Taxis are connected via WiFi to nearby traffic light systems. Since the taxis' location gets updated during the simulation, the access points within reach

are recalculated periodically, and the mapping of data flows to network edges is updated accordingly. Traffic lights can also communicate with other close-by traffic lights via WiFi, effectively forming a mesh network. Additionally, all traffic lights are equipped with a 4G LTE module connecting them to the cloud node via WAN. Although fog nodes are modeled as separate nodes in the infrastructure graph, they are directly attached to a single traffic light. Hence, the connection between fog nodes and their respected traffic light has no bandwidth limitation nor power usage.

The experiments simulate an intelligent transportation system comprising two kinds of applications:

CCTV applications process the data recorded by cameras deployed at each traffic light system. The source task, deployed at the traffic light system itself, sends large amounts of video data to a processing task that is responsible for identifying vehicles and pedestrians for traffic monitoring, enforcing traffic laws and automatic incident detection. The results are sent to a cloud server for further analysis and storage. In the explained scenario, 16 of these applications, one for each traffic light system, are running.

STM are Smart Traffic Management, vehicle-to-Infrastructure (V2I), applications that control traffic lights to ensure maximum throughput of public transportation, in this case, taxis. Each taxi on the map sends sensor data to a processing task, which decides how to control the traffic lights. The processing task then notifies all traffic light systems on the way of the taxi. The number of applications running depends on the number of taxis on the map.

5.2.3 Modeling Focus and Parameterization

All experiments involve the same number of traffic light systems. Since every traffic light system only hosts the source task of a single CCTV application, their power usage is identical in all experiments. Similarly, taxis are generated according to the same distribution in every experiment. Taxis only host a STM application's source task, hence, also the energy consumption of taxis is identical in all experiments. Because of this, the power usage of traffic light systems and taxis is not explicitly modeled. Rather, the simulation focuses on the parts in the system that can be influenced by different network architectures, task placement strategies or energy save mechanisms: The fog and cloud compute nodes as well as the network traffic.

Table 5.1 displays the selected parameter values for the experiment's infrastructure. The cloud data center is expected to provide unlimited computing capabilities and consumes 722 µW/MIPS, as derived in Section 4.4.2. Fog

Table 5.1: Parameterization of infrastructure.

	Max load	P_{static}	σ
Cloud	∞	-	722 µW/MIPS
Fog node	400 000 MIPS	100 W	375 µW/MIPS
$WAN_{4G\ LTE}$	100 Mbit/s	-	6685 nJ/bit
$WiFi_{Taxi \to AP}$	1.3 Gbit/s	-	307 nJ/bit
$WiFi_{AP \to AP}$	1.3 Gbit/s	-	48 nJ/bit

nodes are equipped with a single powerful processor that provides roughly 400 000 MIPS. Fog nodes consume 100 W being idle and 250 W under full load. This results in a σ of 375 µW/MIPS, which, at first sight, seems to be a lot less than cloud nodes. However, note that fog nodes are not shared infrastructure in this scenario: Every active fog node consumes an additional 100 W of static power. Additionally, fog nodes only get loaded to roughly 85 % of their maximum capacity by the placement algorithm, so the impact of static power usage is even more significant.

Section 4.4.5 derived a power usage of 6685 nJ/bit to transfer data over a WAN with 4G LTE access network. For transmitting WiFi from taxis to access points we assume 283.17 nJ/bit based on the measurements of Huang *et al.* [62]. Traffic light systems are equipped with access points of the model Cisco Aironet 1572 and require 23.8 nJ/bit for transmitting data, refer to Section 4.4.4. Because of the lack of real measurement data, we assume the same consumption for receiving data. As a result, sending data from taxis to access points requires

$$283.17\,\text{nJ/bit} + 23.8\,\text{nJ/bit} \approx 307\,\text{nJ/bit},$$

and from access point to access point

$$23.8\,\text{nJ/bit} + 23.8\,\text{nJ/bit} \approx 48\,\text{nJ/bit}.$$

Each CCTV application continuously sends 10 Mbit/s of video data to its processing task. This processing task requires 30 000 MIPS to process this data and outputs 200 kbit/s to the sink task which is always located in the cloud. Each STM application streams 100 kbit/s of sensor data to its processing task, that decides which traffic lights to notify. The processing task requires 7000 MIPS and forwards 50 kbit/s to every traffic light on the way of the taxi.

5.3 Experiments

A series of eight different experiments was conducted in the described smart city environment. Table 5.2 and Figure 5.4 provide an overview of the results. The following sections explain all experiments and analyze and discuss their outcome in detail.

Table 5.2: Aggregated energy consumption (Wh) of infrastructure components of all conducted experiments.

Experiment	Total	Cloud	Fog (dynamic)	Fog (static)	WAN	WiFi
Cloud Only	60 086	21 659	0	0	38 352	74
Fog 1	40 821	15 419	3239	2400	19 485	276
Fog 2	31 047	9841	6136	4800	9873	395
Fog 3	26 754	5147	8573	7200	5413	419
Fog 4	24 049	1517	10 458	9600	1950	522
Fog 5	24 431	98	11 195	12 000	606	530
Fog 6	26 689	0	11 246	14 400	513	529
Fog 6s	22 545	0	11 246	10 274	513	511

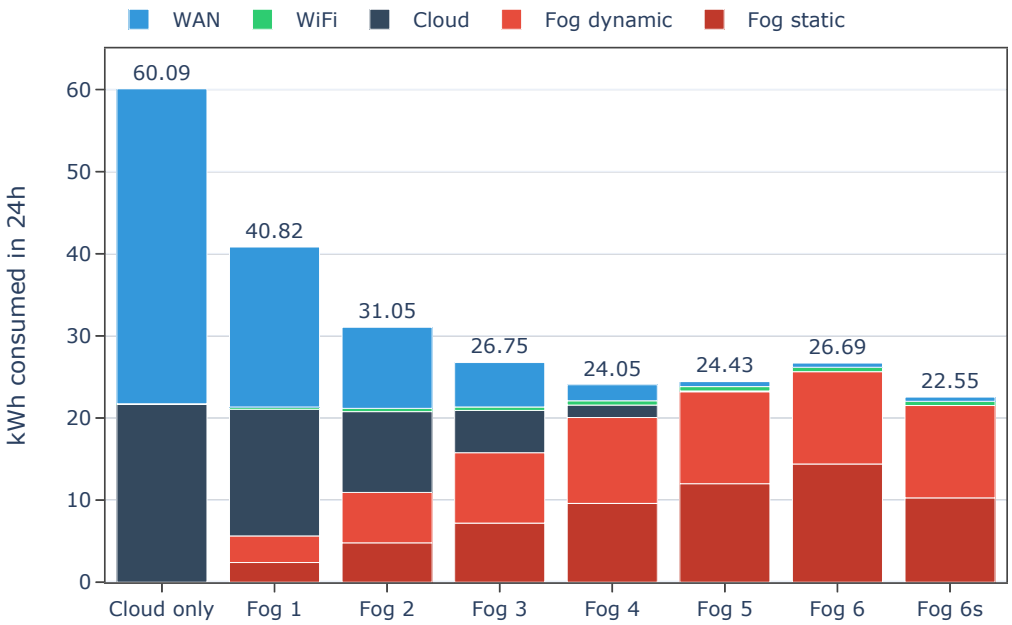

Figure 5.4: Table 5.2 visualized as a stacked bar plot.

5.3.1 Cloud Only

In the first experiment, called *Cloud Only*, all processing tasks of both application types are executed in the cloud. There exist no fog nodes. This results in overall energy consumption of 60 086 Wh over the course of a day. Since LEAF models the consumption of all infrastructure components separately, it provides a clear understanding of what parts of infrastructure were responsible for power usage at which point in time, illustrated in Figure 5.5.

It can be observed that the power usage of the cloud and WAN directly correlate to the number of taxis on the map. The major fraction of the power consumption, namely 63.8 %, can be attributed to the WAN. The reason for this is that all raw sensor data has to travel via WAN to the processing tasks placed in the cloud.

To understand and improve on these results, we take a look at the causes of power usage. As depicted in Figure 5.6, all CCTV applications together require a constant 1416.2 W of power, totaling to 56.5 % of the overall energy consumption. Since LEAF uses an analytical approach for modeling, it is straightforward to calculate where this power usage originates from: 16 traffic light systems constantly stream 10 Mbit/s to the cloud over WAN, resulting in a power usage of

$$16 \cdot 10\,\text{Mbit/s} \cdot 6685\,\text{nJ/bit} = 1069.6\,\text{W}.$$

Additionally, the processing tasks of these applications consume

$$16 \cdot 30\,000\,\text{MIPS} \cdot 722\,\mu\text{W/MIPS} = 346.6\,\text{W}.$$

Based on this analysis, it can be concluded that placing processing tasks closer to the data sources could drastically reduce the energy required by CCTV applications since less data would need to be transferred over the WAN. The accumulated power required by STM applications correlates with the number of taxis on the map. Although the amount of transferred data is not as high as in CCTV applications, this type of application could also benefit from a more geo-proximate processing task placement. Source and sink tasks of STM applications are located at the edge of the network. This implies that the data does not need to be sent to the cloud but could be processed locally, avoiding data transfer over the power-intensive 4G LTE connection.

5.3.2 Fog Only

The previous analysis concluded that a processing task placement close to the source tasks might reduce overall energy consumption. To confirm this, six

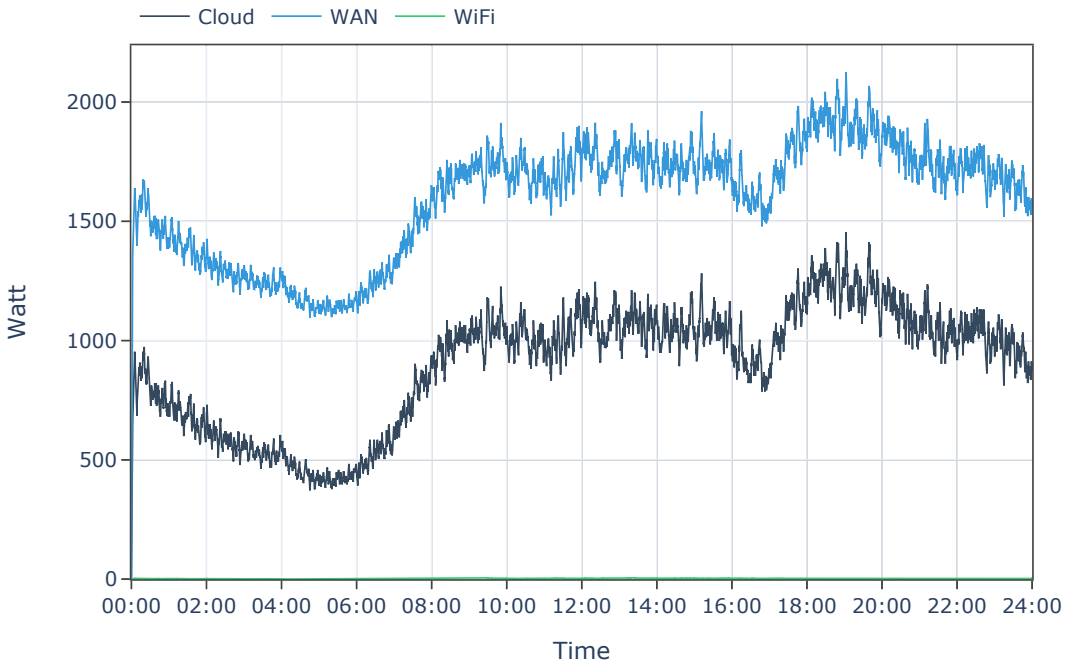

Figure 5.5: Power usage of infrastructure in experiment *Cloud Only*.

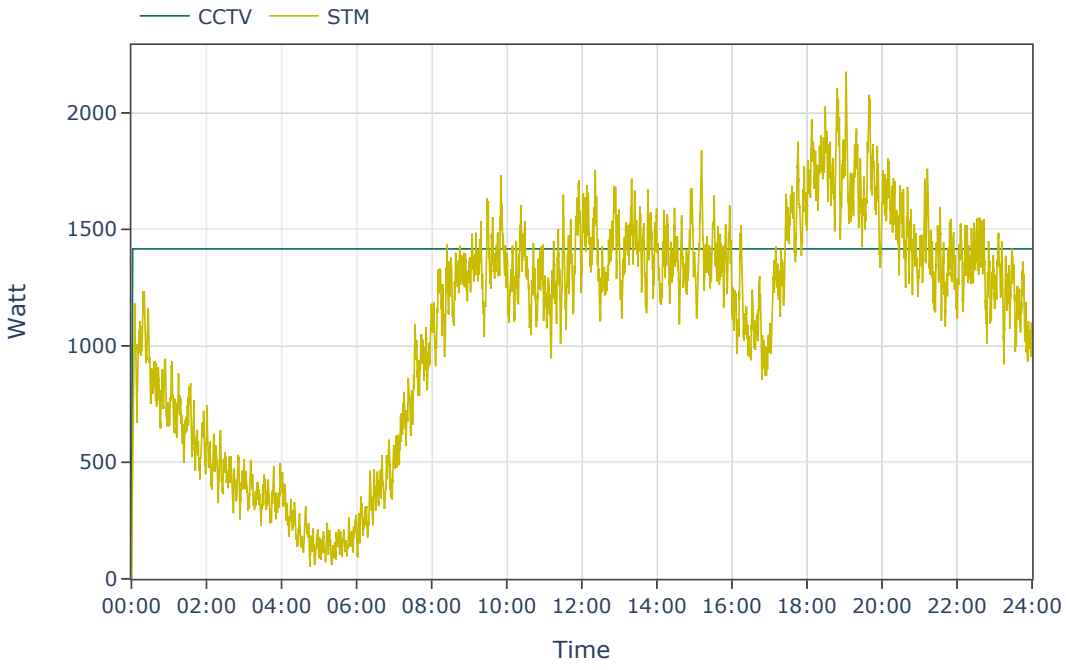

Figure 5.6: Power usage of applications in experiment *Cloud Only*.

random traffic light systems were equipped with fog nodes in the next experiment, called *Fog 6*. These fog nodes provide sufficient computing capacity; no tasks have to be offloaded to the cloud. Processing tasks are distributed evenly across the fog nodes. The resulting infrastructure power usage over time is presented in Figure 5.7.

The power usage of network decreased sharply: By avoiding large and unnecessary data flows, WAN energy consumption dropped from 38 352 Wh to 513 Wh. At the same time, WiFi consumption only increased from 74 Wh to 529 Wh. As a consequence, the fog nodes themselves are now the only relevant cause of power usage, accounting for 96.0 % of the total energy consumption in this experiment. Figure 5.7 reveals that a major fraction of this is static power consumption: The six fog nodes require 100 W of static power each, totaling in 14.4 kW of energy in a day.

The accumulated application power usage of the *Fog 6* experiment, depicted in Figure 5.8, shows an interesting picture. Although the CCTV applications require constant MIPS and bit/s during the entire simulation, the reported power consumption varies over time. This effect occurs because LEAF allocates static power consumption of fog nodes proportionally to applications running on them. However, fog nodes are not utilized efficiently in this experiment, especially at night when only a few taxis are on the road. During the day, the CCTV processing tasks share the fog nodes with more STM processing tasks. Consequently, power usage attributed to CCTV applications drops.

The analysis of infrastructure and application power usage in experiment *Fog 6* yielded that the many deployed fog nodes impose a high static power usage on the system. Moreover, the fact that these fog nodes are not always fully utilized is an evident inefficiency. This inevitably raises the question if fewer fog nodes that offload tasks to the cloud at peak times could lower overall energy consumption. Further experiments were therefore conducted to demonstrate how LEAF can help to plan more energy-efficient infrastructures.

5.3.3 Fog and Cloud

To determine the optimal number of fog nodes for minimal overall energy consumption, experiment *Fog 6* was repeated with only one to six available fog nodes. These experiments are called *Fog 1* to *Fog 5*, respectively. Tasks are still distributed evenly across the available fog nodes, but once all fog nodes are running at more than 85 % capacity, tasks are being offloaded to the cloud. Refer to Figure 5.4 and Table 5.2 for the results of experiments *Fog 1* to *Fog 5* in comparison to *Cloud Only* and *Fog 6*.

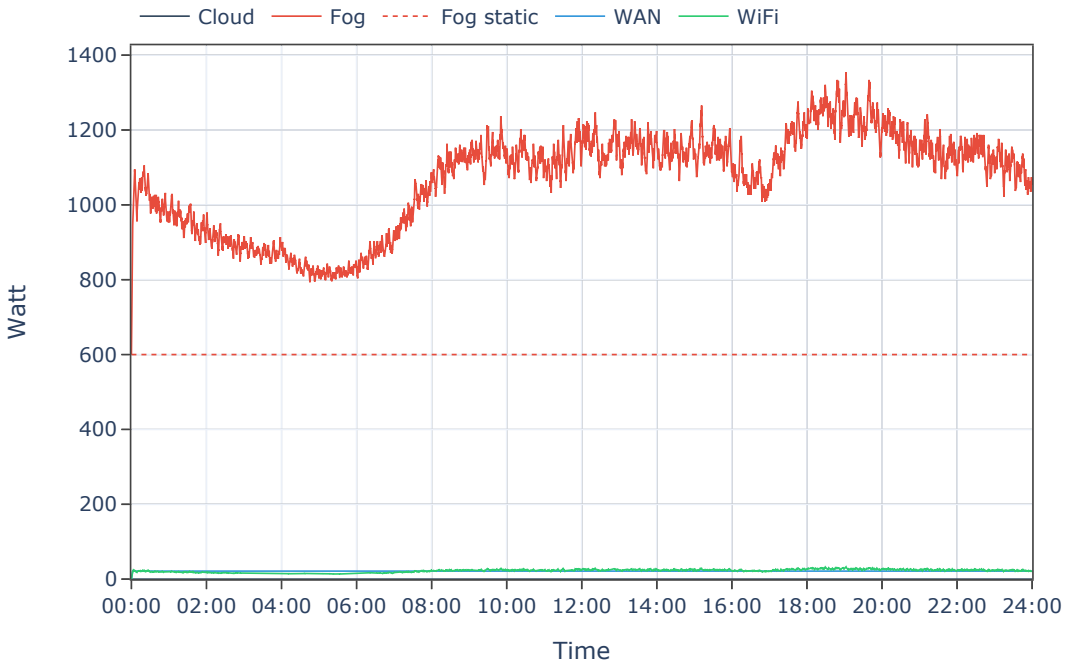

Figure 5.7: Power usage of infrastructure in experiment *Fog 6*.

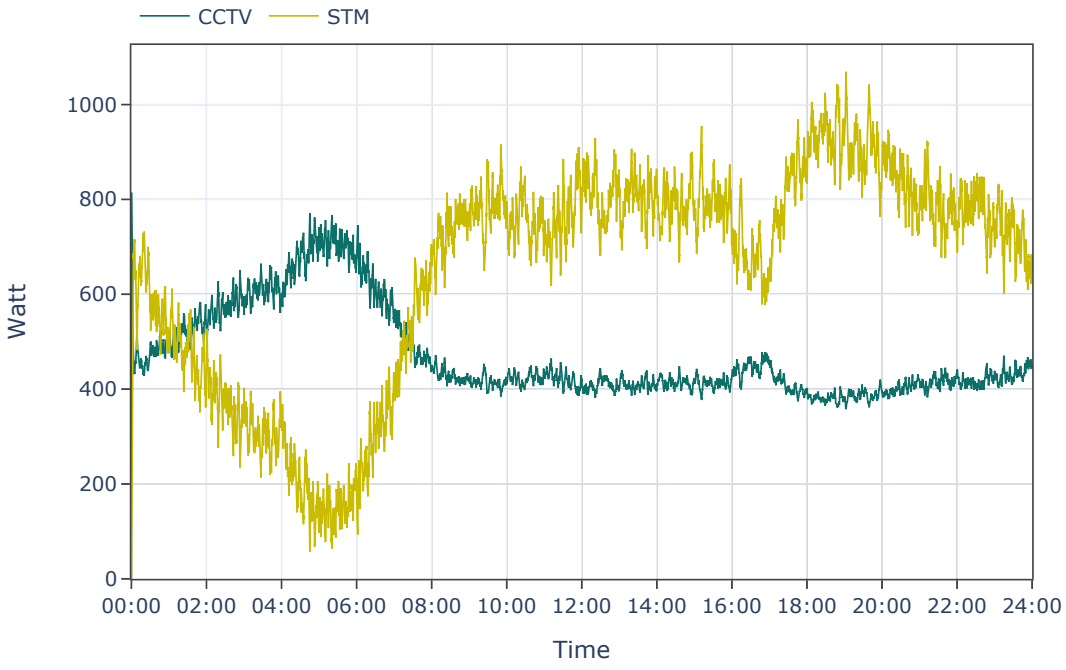

Figure 5.8: Power usage of applications in experiment *Fog 6*.

Interestingly, the lowest overall energy consumption was achieved by deploying only four fog nodes, saving 2640 Wh compared to *Fog 6*. The fog nodes in experiment *Fog 4* can provide roughly

$$4 \cdot 400\,000\,\text{MIPS} \cdot 0.85 = 1\,360\,000\,\text{MIPS},$$

which is enough to host all CCTV processing tasks (480 000 MIPS) and around 125 STM processing tasks (875 000 MIPS). However, over long periods of the day, more than 125 taxis are on the map simultaneously - during rush hour, more than 200 - so many processing tasks have to be executed in the cloud. This placement adds 1517 Wh of cloud energy consumption and increases the WAN energy consumption by 1437 Wh. On the other hand, 5588 Wh are saved by reducing the number of fog nodes, leading to the lower consumption overall. Figure 5.9 illustrates how the fog nodes are fully utilized most of the time.

Despite the improvements, the configuration used in experiment *Fog 4* has drawbacks: First, during the night, much energy continues to get wasted on the static energy consumption of barely utilized fog nodes. Second, for a large portion of the day, tasks have to be offloaded to the cloud, resulting in increased power consumption for data transfer. Moreover, the biggest shortcoming of this solution is that it is over-optimized. A configuration of four fog nodes may lead to minimal consumption during an average day of the year. However, there are certainly days with much more or much less taxi traffic. On such days, the fog computing layer will either not provide sufficient capacity or be entirely underutilized, resulting in poorer energy efficiency.

5.3.4 Fog with Energy-Saving Mechanism

The last experiment, called *Fog 6s*, aims to remedy the deficiencies of experiment *Fog 4* and demonstrates how LEAF supports the implementation of adaptive task placement strategies and energy-saving mechanisms.

In this experiment, once a fog node is not utilized for 5 seconds, it will automatically be put to sleep. Powered-off fog nodes have a static power consumption of zero. In order to fully exploit this energy-saving mechanism, *Fog 6s* uses a different task placement algorithm than the previous experiments. Instead of distributing the workload evenly across the fog nodes, it tries to consolidate as much work as possible on a minimum number of nodes before allocating resources on further nodes. Put another way, the placement algorithm tries to maximize the number of idle fog nodes during the simulation.

Experiment *Fog 6s* was conducted with a maximum of six fog nodes available. Individual fog nodes are still only loaded up to around 85 %, to ensure

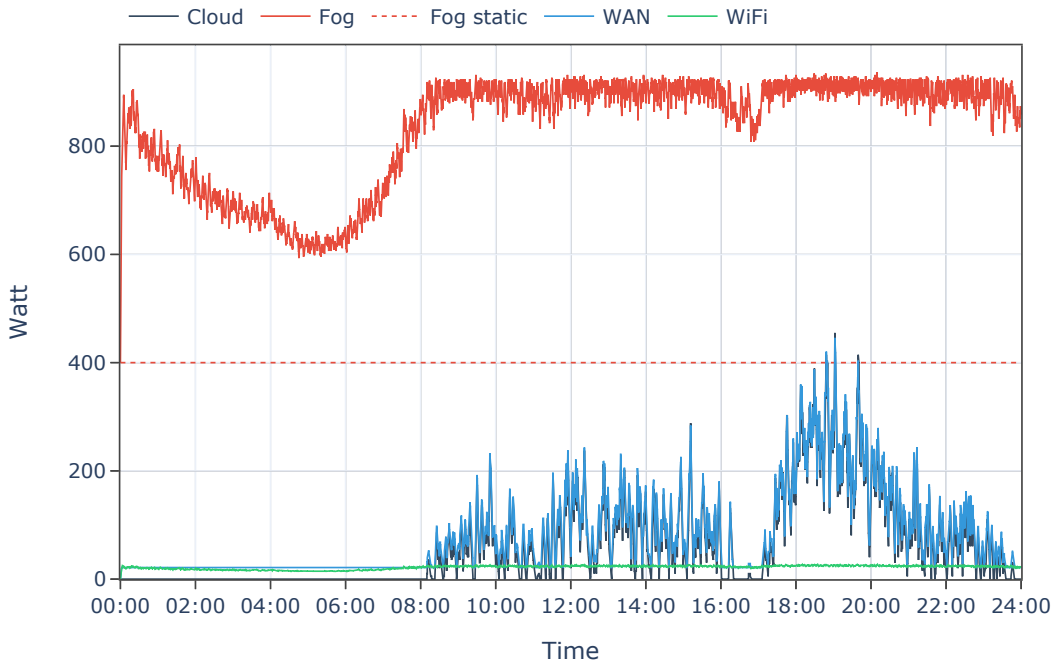

Figure 5.9: Power usage of infrastructure in experiment *Fog 4*.

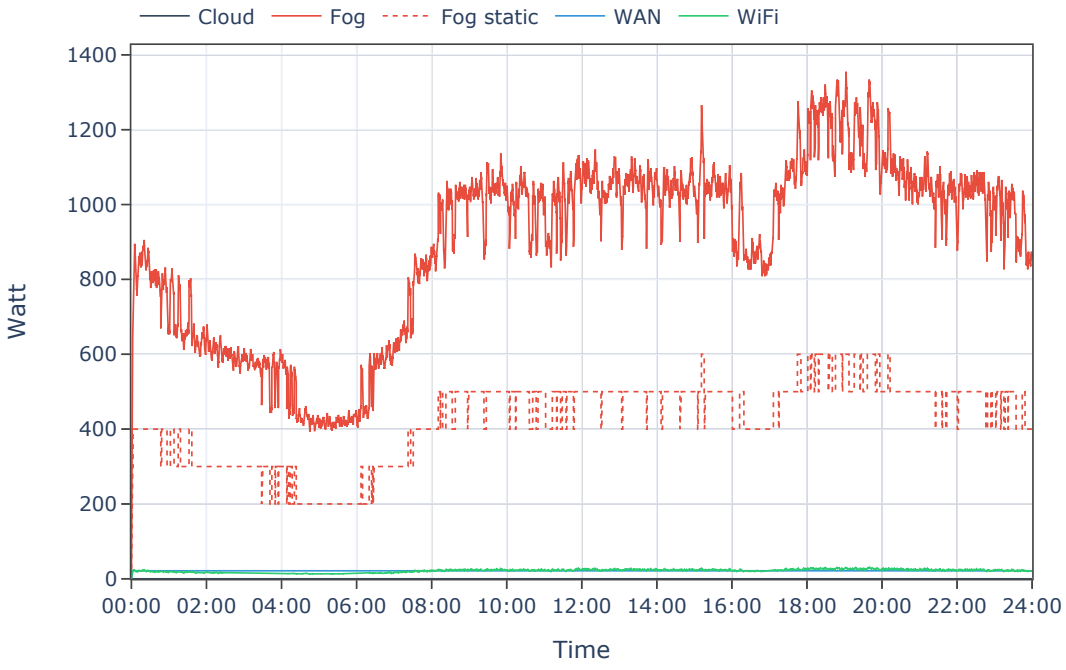

Figure 5.10: Power usage of infrastructure in experiment *Fog 6s*.

comparability with the other experiments. Figure 5.10 displays the infrastructure's power usage of this experiment.

It becomes evident that the overall static power usage of fog nodes is no longer constant. In fact, all six fog nodes are only active for around 1 hour and 42 minutes during the entire day. Between 04:00 and 06:00 in the morning, only two out of six nodes are active, resulting in static power usage of 200 W at that time. The network power usage profile is roughly equivalent to the original *Fog 6* experiment.

Consequently, the power usage of applications was reduced too, see Figure 5.11. Although all processing tasks are now executed on fog nodes, the CCTV applications have very constant power usage during the day, similar to the *Cloud Only* experiment. The big difference is that the power usage of CCTV applications now averages around 350 W, compared to 1416.2 W in the *Cloud Only* placement. The constant power usage profile is an effect of the improved placement strategy, which consolidates tasks on few fog nodes. Only in the early morning hours, when there are almost no taxis on the road, the attributed power usage of CCTV applications slightly rises.

On the other hand, the power usage profile of STM applications is now again very dynamic; one can spot when fog nodes had to be activated, and when they got switched off again. The STM applications benefit from the energy-saving mechanism by becoming more power proportional compared to the other experiments that include fog nodes. At night the attributed power usage of all STM applications drops below 80 W.

Lastly, we want to study the performance of the new experiment in comparison to the others. In total, only 22 545 Wh get consumed in experiment *Fog 6s*. It thereby undercuts the previously best performing experiment *Fog 4* which required 24 049 Wh. To illustrate how the applied energy-saving mechanism was able to conserve energy, Figure 5.12 displays the overall power usage of experiments *Fog 4* and *Fog 6s* over time. Smoothing was applied to make the results simpler to distinguish.

It is apparent that the mechanisms applied in experiment *Fog 6s* eliminate the drawbacks of experiment *Fog 4*, mentioned at the end of Section 5.3.3. When there is little traffic in the city, *Fog 6s* undercuts the consumption of *Fog 4* by reducing static power usage of fog nodes. In times of high load, *Fog 6s* continues to provide sufficient resources to process all data in the fog layer. This avoids the additional power usage that *Fog 4* is facing when offloading tasks to the cloud. At times of average load, the power usage profile of both experiments is almost identical. Moreover, the strategy applied in the *Fog 6s* experiment is suitable for guaranteeing an optimal utilization of fog infrastructure even on days with more or less taxi traffic than on the simulated, average day.

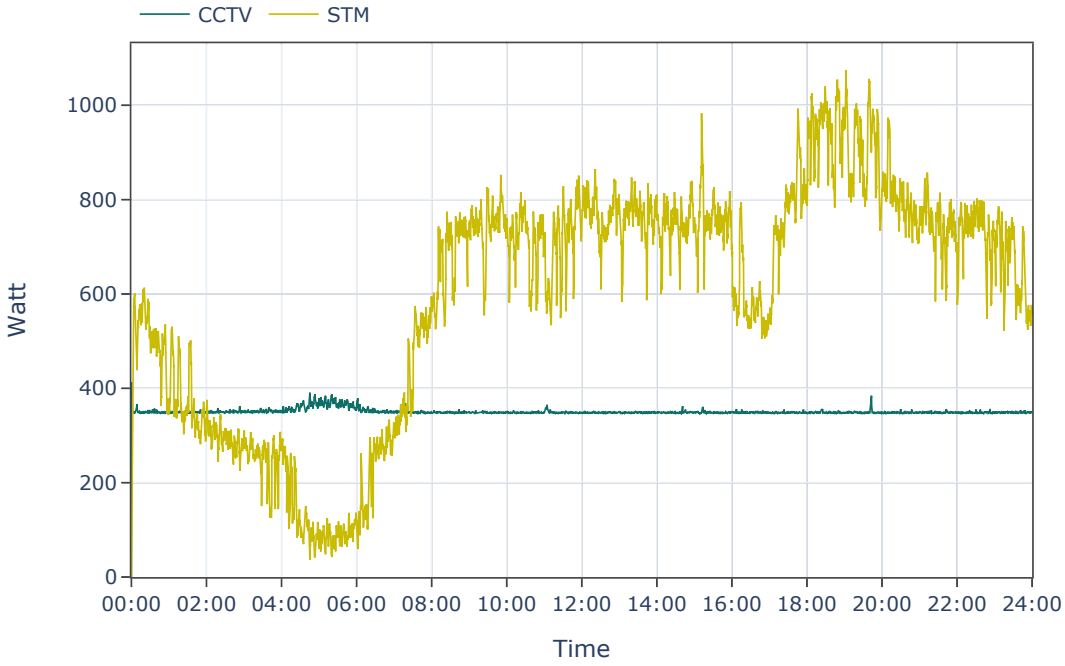

Figure 5.11: Power usage of applications in experiment *Fog 6s*.

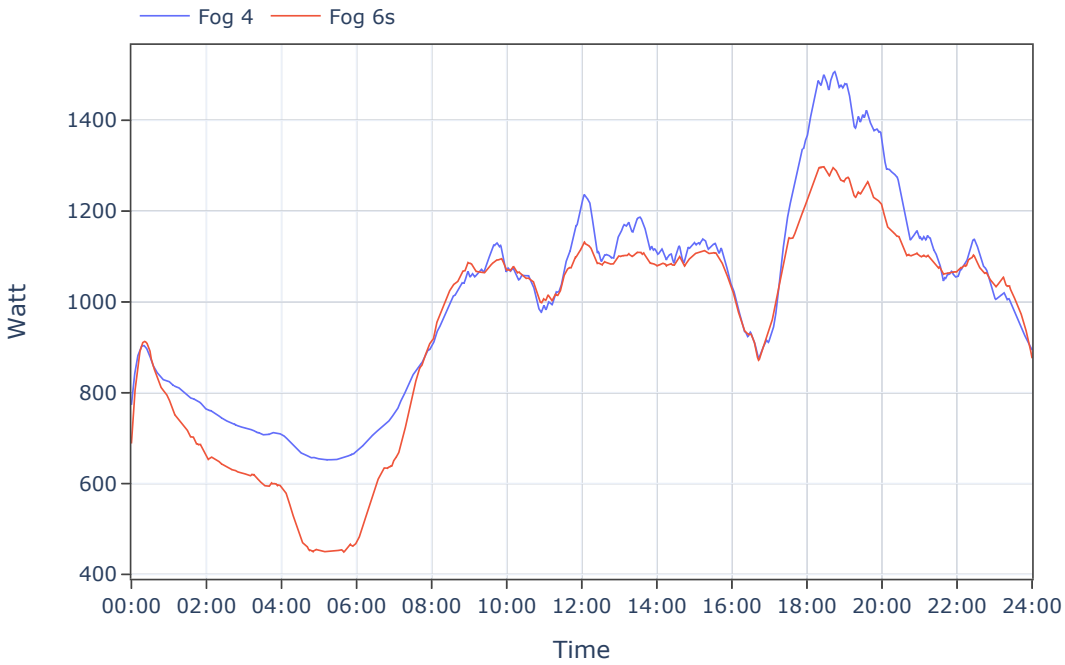

Figure 5.12: Comparison of overall power usage in experiments *Fog 4* and *Fog 6s* (smoothing applied).

5.4 Performance and Scalability

The evaluation has shown that LEAF is a useful tool to model, analyze, and reduce energy consumption in fog computing scenarios. Since performance and scalability are explicit requirements placed on the proposed model, this section focuses on these aspects.

During the course of every experiment, 46 737 taxis crossed the map, and a total of 329 876 tasks was generated and executed. On average, 110 taxis were present on the map simultaneously, during peak times up to 219. The experiments were carried out on a single core of a 2.7 GHz Intel Core i7-6820HQ and each took around 350 seconds to complete. Accordingly, the model was able to traverse through nearly 250 seconds of simulated time in one second of real-time. Therefore, it can be concluded that LEAF meets Requirement (4) because it enables the simulation of experiments with hundreds or thousands of devices and tasks and executes orders of magnitude faster than real-time.

In order to further investigate the scalability of LEAF, an experiment with twice the number of generated taxis was conducted. In other words, the λ parameter of the Poisson distribution responsible for generating taxis at each time step was doubled. As a result, a total of 93 217 taxis crossed the map, and 658 355 tasks were created and destroyed.

This extended simulation took 1757 seconds to complete - five times longer than the previous experiments. The main reason for the non-linear growth is a series of memory leaks in the CloudSim Plus implementation on which the current prototype is based. Most of these issues could be resolved by fixing the CloudSim Plus core or re-implementing LEAF on another platform.

Nevertheless, also the LEAF model itself contains some non-linear complexity growth related to the proposed infrastructure graph. First, with a growing infrastructure graph, routing becomes more complicated, namely, the mapping of application data flows to network connections. Even heuristic algorithms for finding shortest paths have a higher-than-linear run time. This problem could be mitigated by using domain knowledge to trim the graph before searching shortest paths. For instance, in the explained scenario, we know beforehand that directly connecting two access points is always faster and more energy-efficient than routing the traffic though a taxi. Hence, removing all taxis from the graph before searching for a fog node is possible, which significantly reduces the search space.

The second difficulty is that the infrastructure topology is dynamic, and nodes are mobile. Wireless connections have to be re-evaluated periodically to see what nodes are in range of each other. These calculations can have a severe performance impact once there are many possible network connections, for example, when taxis were able to communicate with each other too. In this case, the number of comparisons would raise quadratically with the

number of taxis on the map. One way to deal with this limitation is applying more advanced methods than linear search such as fixed-radius near neighbor algorithms. Furthermore, domain knowledge can be applied to reduce the number of comparisons. For instance, in the provided example, if a taxi is very far away from another taxi at a specific time step, there is a minimal time until these two taxis may get in range of each other. No further comparisons need to be made during this period.

5.5 Compliance with Requirements

The previous sections have demonstrated the usefulness of LEAF by showing how it can be used to effectively model, analyze, and reduce the energy consumption of fog computing scenarios. To summarise the evaluation, it will be confirmed that LEAF meets all requirements listed in Section 1.1.

Requirement (1), LEAF's ability to model realistic fog computing environments, was demonstrated in the experimental setup. (1.1) The experiments simulated a distributed fog computing environment with different types of nodes, of which some, the fog nodes, are resource-constrained. The compute nodes were interconnected with different protocols, namely WiFi and WAN, with a 4G LTE access network. (1.2) Taxis, as well as traffic lights and fog nodes, were location-aware. Taxis were mobile and could dynamically join and leave the simulation. (1.3) The experimental setup covered two different types of streaming applications. CCTV applications were linear data pipelines while STM applications had multiple sink tasks, one sink per traffic light on the route of the taxi.

The experiments conducted in Section 5.3 were thoroughly analyzed regarding their infrastructure and application energy consumption. Therefore it has been shown that Requirement (2), the holistic energy consumption model, is met. LEAF captures the power usage of (2.1) data centers, (2.2) edge devices, and (2.3) network links. (2.4) Furthermore, it is possible to trace back the infrastructure's power usage to the responsible applications.

Experiment *Fog 6s* contained an energy-conserving online decision making algorithm. The applied energy-saving mechanism dynamically puts fog nodes to sleep during off-peak periods. Besides, this experiment comprised an advanced task placement strategy that consolidates workload on a minimal number of nodes. Consequently, LEAF complies with Requirement (3), the support of energy-aware online decision making.

Lastly, Section 5.4 covered the performance and scalability of the implemented prototype. The results confirm that LEAF meets Requirement (4), namely, it enables the simulation of complex scenarios with hundreds of parallel existing devices that execute thousands of tasks substantially faster than

real-time. Furthermore, scalability limitations were identified, and suggestions were made on how these could be avoided or at least reduced in future versions.

6 Conclusion

This thesis introduced LEAF, a simulator for modeling Large Energy-Aware Fog computing environments. LEAF features a holistic but detailed energy consumption model that covers the individual infrastructure components and the applications running on it. Unlike most existing simulators, which focus exclusively on energy consumption of data centers or battery-constrained end devices [13, 14, 23], LEAF additionally takes the power requirements of network connections into account. This is particularly relevant in the context of IoT since mobile communication technologies such as 4G LTE or 5G are comparatively power-hungry.

LEAF allows for the modeling of large-scale fog computing scenarios that execute thousands of streaming applications on a distributed, heterogeneous infrastructure. Compute nodes can be interconnected with different types of wired or wireless networking protocols, and edge devices can be mobile and join or leave the topology during the simulation. This level of realism permits research on energy-conserving fog computing architectures leading to more informed decisions in the planning of future infrastructure. Furthermore, the proposed model enables online decision making based on power usage, which can be used to implement energy-aware task placement strategies or routing policies. These algorithms can make direct use of LEAF's ability to trace the power usage of infrastructure back to the responsible applications in order to identify and mitigate potential inefficiencies. Moreover, different kinds of energy-saving mechanisms can be integrated into simulations.

What further distinguishes LEAF from existing fog computing simulators is the combination of analytical and numerical modeling approaches. Instead of modeling network traffic in detail, all data flows, and power models are represented by parameterizable, mathematical equations. This method leads to results that are easy to analyze and ensures scalability to hundreds or thousands of devices and applications. The thesis collects findings of various papers on the energy usage of different compute and networking equipment, including a detailed derivation of WAN connection parameters, to provide the reader with examples on how to model and parameterize LEAF experiments.

A series of experiments was carried out on a prototypical implementation to show that the model is useful and meets its stated requirements. The eval-

uation demonstrated that LEAF enables the comprehensive analysis of the infrastructure's power usage and can reveal the cause-effect relationship between task placement and energy consumption. The integration of an adaptive energy-saving mechanism illustrated how the energy consumption of fog computing environments can be further reduced by consolidating workloads and switching off unused computing capacity. Furthermore, the evaluation results indicate that fog computing may indeed be able to conserve energy in the future, mainly by reducing WAN usage. This is in line with findings from previous studies [6, 9, 10, 15]. However, in order to obtain more accurate results, further investigations should be conducted based on real measurement data.

From an environmental point of view, a general issue with mere technological efficiency gains is that they are often canceled out by so-called rebound effects. The Jevons paradox describes that improved efficiency sometimes results in increased demand - up to the point where the overall effect can be detrimental. Cloud computing is a notable example of this, where energy efficiency is improving at fast rates; nevertheless, the overall power required by cloud computing is rising every year. Admittedly, improved energy efficiency is not the only reason for the increased demand in cloud computing. However, some of the business models existing today, such as offering high-quality video streaming, would certainly not be economical with less efficient hardware from 20 years ago.

Instead of aiming solely at further improvements in energy efficiency, an exciting direction for future research could be focusing on the carbon emissions caused by fog computing infrastructures directly. Existing models and simulators do not take into account the fact that the amount of green energy in the grid fluctuates throughout the day. Google recently published an article[1] on how they are planning to shift non-urgent tasks, like processing YouTube videos, to times where renewable power sources provide most of their energy. Similarly, a local lack or surplus of low-carbon energy could influence task placement and workload scheduling in fog computing environments. Extending LEAF with additional data sources like weather data, energy prices, and data on the carbon intensity of the current energy mix will enable research on architectures and algorithms that save money and tonnes of CO_2, rather than kilowatt-hours.

[1] `https://blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows`, accessed 2020-05-05

References

1. Su, K., Li, J. & Fu, H. *Smart City and the Applications* in *2011 International Conference on Electronics, Communications and Control (ICECC)* (2011).
2. Atzori, L., Iera, A. & Morabito, G. The Internet of Things: A survey. *Computer Networks* (2010).
3. Jones, N. How to stop data centres from gobbling up the world's electricity. *Nature* (2018).
4. Dastjerdi, A. V. & Buyya, R. Fog Computing: Helping the Internet of Things Realize Its Potential. *Computer* (2016).
5. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J. & Jue, J. P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* (2019).
6. Jalali, F., Hinton, K., Ayre, R., Alpcan, T. & Tucker, R. S. Fog Computing May Help to Save Energy in Cloud Computing. *IEEE Journal on Selected Areas in Communications* (2016).
7. Noor, T. H., Zeadally, S., Alfazi, A. & Sheng, Q. Z. Mobile cloud computing: Challenges and future research directions. *Journal of Network and Computer Applications* (2018).
8. Miller, R. *Hyperscale Data Centers: A Data Center Frontier Special Report* tech. rep. (Data Center Frontier, 2019).
9. Ahvar, E., Orgerie, A.-C. & Lèbre, A. Estimating Energy Consumption of Cloud, Fog and Edge Computing Infrastructures. *IEEE Transactions on Sustainable Computing* (2019).
10. Yan, M., Chan, C., Gygax, A., Yan, J., Campbell, L., Nirmalathas, A. & Leckie, C. Modeling the Total Energy Consumption of Mobile Network Services and Applications. *Energies* (2019).
11. Beloglazov, A., Buyya, R., Lee, Y. & Zomaya, A. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. *Advances in Computers* (2010).
12. Heinzelman, W. R., Chandrakasan, A. & Balakrishnan, H. *Energy-efficient communication protocol for wireless microsensor networks* in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences* (2000).

13. Mechalikh, C., Taktak, H. & Moussa, F. *PureEdgeSim: A Simulation Toolkit for Performance Evaluation of Cloud, Fog, and Pure Edge Computing Environments* in *2019 International Conference on High Performance Computing & Simulation (HPCS)* (2019).

14. Hirsch, M., Mateos, C., Rodriguez, J. & Zunino, A. DewSim: A Trace-driven Toolkit for Simulating Mobile Device Clusters in Dew Computing Environments. *Software Practice and Experience* (2019).

15. Sarkar, S. & Misra, S. Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. *IET Networks* (2016).

16. Valancius, V., Laoutaris, N., Massoulié, L., Diot, C. & Rodriguez, P. *Greening the Internet with Nano Data Centers* in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies* (2009).

17. Google. *Google Environmental Report 2019* tech. rep. (Google, 2019).

18. Masanet, E., Shehabi, A., Lei, N., Smith, S. & Koomey, J. Recalibrating global data center energy-use estimates. *Science* (2020).

19. Varshney, P. & Simmhan, Y. L. Demystifying Fog Computing: Characterizing Architectures, Applications and Abstractions. *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)* (2017).

20. Mayer, R., Graser, L., Gupta, H., Saurez, E. & Ramachandran, U. *EmuFog: Extensible and scalable emulation of large-scale fog computing infrastructures* in *2017 IEEE Fog World Congress (FWC)* (2017).

21. Coutinho, A. A., Greve, F., Prazeres, C. & Cardoso, J. *Fogbed: A Rapid-Prototyping Emulation Environment for Fog Computing* in *2018 IEEE International Conference on Communications (ICC)* (2018).

22. Hasenburg, J., Grambow, M., Grünewald, E., Huk, S. & Bermbach, D. *MockFog: Emulating Fog Computing Infrastructure in the Cloud* in *2019 IEEE International Conference on Fog Computing (ICFC)* (2019).

23. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. & Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience* (2017).

24. Zeng, X., Garg, S. K., Strazdins, P., Jayaraman, P. P., Georgakopoulos, D. & Ranjan, R. IOTSim. *Journal of Systems Architecture* (2017).

25. Sonmez, C., Ozgovde, A. & Ersoy, C. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies* (2018).

26. Qayyum, T., Malik, A., Khan, M., Khalid, O. & Khan, S. FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment. *IEEE Access* (2018).

27. Núñez, A., Vázquez-Poletti, J., Caminero, A., Castañé, G., Carretero, J. & Llorente, I. ICanCloud: A Flexible and Scalable Cloud Infrastructure Simulator. *Journal of Grid Computing* (2012).

28. Kliazovich, D., Bouvry, P., Audzevich, Y. & Khan, S. *GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers* in *The Journal of Supercomputing* (2010).

29. Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C. & Buyya, R. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software Practice and Experience* (2011).

30. Flores, H., Tarkoma, S., Nurmi, P. & Hui, P. *MobileCloudSim: A Context-Aware Simulation Toolkit for Mobile Computational Offloading* in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers* (2018).

31. Kecskemeti, G., Casale, G., Jha, D. N., Lyon, J. & Ranjan, R. Modelling and Simulation Challenges in Internet of Things. *IEEE Cloud Computing* (2017).

32. Svorobej, S., Takako Endo, P., Bendechache, M., Filelis-Papadopoulos, C., Giannoutakis, K., Gravvanis, G., Tzovaras, D., Byrne, J. & Lynn, T. Simulating Fog and Edge Computing Scenarios: An Overview and Research Challenges. *Future Internet* (2019).

33. Filho, M. C. S., Oliveira, R. L., Monteiro, C. C., Inácio, P. R. M. & Freire, M. M. *CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness* in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (2017).

34. Cisco Systems, Inc. *Cisco Annual Internet Report (2018–2023)* tech. rep. (Cisco Systems, Inc, 2020).

35. Yi, S., Qin, Z. & Li, Q. *Security and Privacy Issues of Fog Computing: A Survey* in *International Conference on Wireless Algorithms, Systems, and Applications* (2015).

36. Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. *Fog Computing and Its Role in the Internet of Things* in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing* (2012).

37. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* (2017).

38. Gupte, S. & Younis, M. *Vehicular networking for intelligent and autonomous traffic management* in *2012 IEEE International Conference on Communications (ICC)* (2012).

39. Dolui, K. & Datta, S. K. *Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing* in *2017 Global Internet of Things Summit (GIoTS)* (2017).
40. Brogi, A., Forti, S. & Ibrahim, A. *How to Best Deploy Your Fog Applications, Probably* in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)* (2017).
41. Maria, A. *Introduction to Modeling and Simulation* in *Proceedings of the 29th Conference on Winter Simulation* (1997).
42. Basmadjian, R., Ali, N., Niedermeier, F., de Meer, H. & Giuliani, G. *A Methodology to Predict the Power Consumption of Servers in Data Centres* in *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking* (2011).
43. Barroso, L. A. & Hölzle, U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* in *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* (2008).
44. Fan, X., Weber, W.-D. & Barroso, L. A. *Power Provisioning for a Warehouse-Sized Computer* in *Proceedings of the 34th Annual International Symposium on Computer Architecture* (2007).
45. Baliga, J., Ayre, R., Hinton, K., Sorin, W. V. & Tucker, R. S. Energy Consumption in Optical IP Networks. *Journal of Lightwave Technology* (2009).
46. Hinton, K., Jalali, F. & Matin, A. Energy Consumption Modelling of Optical Networks. *Photonic Network Communications* (2015).
47. Barroso, L. A. & Hölzle, U. The Case for Energy-Proportional Computing. *IEEE Computer* (2008).
48. Guérout, T., Monteil, T., Costa, G. D., Calheiros, R. N., Buyya, R. & Alexandru, M. Energy-aware simulation with DVFS. *Simulation Modelling Practice and Theory* (2013).
49. Vishwanath, A., Jiazhen Zhu, Hinton, K., Ayre, R. & Tucker, R. S. *Estimating the energy consumption for packet processing, storage and switching in optical-IP routers* in *2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)* (2013).
50. Malone, C. & Belady, C. *Metrics to Characterize Data Center & Equipment Energy Use* in *Proceedings of Digital Power Forum, Richardson, TX.* (2006).
51. V. Avelar, D. A. & French, A. *PUE: A Comprehensive Examination of the Metric* tech. rep. (The Green Grid, 2012).
52. Institute, U. *2019 Annual Data Center Survey Results* tech. rep. (Uptime Institute, 2019).

53. Castañé, G., Núñez, A., Llopis, P. & Carretero, J. E-mc2: A formal framework for energy modelling in cloud computing. *Simulation Modelling Practice and Theory* (2013).

54. Beloglazov, A. & Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation: Practice and Experience* (2012).

55. Garg, S. K. & Buyya, R. *NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations* in *2011 Fourth IEEE International Conference on Utility and Cloud Computing* (2011).

56. Jiang, X., Ye, K. & Huang, P. *DartCSim+: Enhanced CloudSim with the Power and Network Models Integrated* in *IEEE International Conference on Cloud Computing, CLOUD* (2013).

57. Brogi, A., Forti, S., Guerrero, C. & Lera, I. How to place your apps in the fog: State of the art and open challenges. *Software: Practice and Experience* (2019).

58. Baliga, J., Ayre, R., Hinton, K. & Tucker, R. S. Energy consumption in wired and wireless access networks. *IEEE Communications Magazine* (2011).

59. Vishwanath, A., Jalali, F., Ayre, R., Alpcan, T., Hinton, K. & Tucker, R. *Energy consumption of interactive cloud-based document processing applications* in *2013 IEEE International Conference on Communications (ICC)* (2013).

60. Yan, M., Chan, C. A., Gygax, A. F., Yan, J., Campbell, L., Nirmalathas, A. & Leckie, C. Modeling the Total Energy Consumption of Mobile Network Services and Applications. *Energies* (2019).

61. Abdulkafi, A., Sieh, K., Koh, J., Chieng, D., Ting, A. & Ghaleb, A. *Energy efficiency of LTE macro base station* in *2012 International Symposium on Telecommunication Technologies* (2012).

62. Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S. & Spatscheck, O. *A Close Examination of Performance and Power Characteristics of 4G LTE Networks* in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (2012).

63. Desset, C., Debaillie, B., Giannini, V., Fehske, A., Auer, G., Holtkamp, H., Wajda, W., Sabella, D., Richter, F., Gonzalez, M. J., Klessig, H., Gódor, I., Olsson, M., Imran, M. A., Ambrosy, A. & Blume, O. *Flexible power modeling of LTE base stations* in *2012 IEEE Wireless Communications and Networking Conference (WCNC)* (2012).

64. Meisner, D., Gold, B. & Wenisch, T. *PowerNap: Eliminating Server Idle Power* in *ACM SIGPLAN Notices* (2009).

65. Gupta, M. & Singh, S. *Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links* in *IEEE International Conference on Communications (ICC) Glasgow* (2007).

66. Choi, J., Govindan, S., Urgaonkar, B. & Sivasubramaniam, A. *Profiling, Prediction, and Capping of Power Consumption in Consolidated Environments* in *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems* (2008).

67. Srikantaiah, S., Kansal, A. & Zhao, F. Energy-Aware Consolidation for Cloud Computing. *Cluster Computing - CLUSTER* (2008).