End-to-end simulation of the energy consumption of Fog infrastructures and their applications

Clément Courageux-Sudan

▶ **To cite this version:**

Clément Courageux-Sudan. End-to-end simulation of the energy consumption of Fog infrastructures and their applications. Networking and Internet Architecture [cs.NI]. Université de Rennes, 2023. English. NNT : 2023URENE008 . tel-04496167

HAL Id: tel-04496167
https://theses.hal.science/tel-04496167

Submitted on 8 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

école normale supérieure rennes

THÈSE DE DOCTORAT DE

L'ÉCOLE NORMALE SUPÉRIEURE DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *INFO*

Par
Clément COURAGEUX-SUDAN

End-to-end simulation of the energy consumption of fog infrastructures and their applications

Thèse présentée et soutenue à Rennes (FRANCE), le 8 Décembre 2023
Unité de recherche : IRISA

Rapporteurs avant soutenance :

Isabelle GUERIN-LASSOUS	Professeur des Universités à l'Université Lyon 1.
Pierre SENS	Professeur des Universités à Sorbonne Université.

Composition du Jury :

Examinateurs :	Adrian FRIDAY	Professor à Lancaster University
	Isabelle GUERIN-LASSOUS	Professeur des Universités à l'Université Lyon 1.
	Pierre SENS	Professeur des Universités à Sorbonne Université.
	François TAIANI	Professeur des Universités à l'Université de Rennes
Dir. de thèse :	Anne-Cécile ORGERIE	Directrice de recherche au CNRS.
Dir. de thèse :	Martin QUINSON	Professeur des Universités à l'ENS Rennes.

Remerciements

Je tiens tout d'abord à remercier Anne-Cécile Orgerie et Martin Quinson pour leur encadrement tout au long de cette thèse. Ce travail a été possible grâce à leur présence, à leurs questions, ainsi qu'aux encouragements que j'ai reçus tout au long de ces trois années. Travailler en leur compagnie a été un réel plaisir.

Je souhaite également remercier Isabelle Guérin-Lassous et Pierre Sens pour avoir accepté d'être rapporteurs de mon manuscrit, ainsi que François Taïani et Adrian Friday pour leur participation au jury de cette thèse.

Un grand merci à tous les membres de l'équipe Myriads, avec qui j'ai pu partager de merveilleux moments.

Enfin, je tiens à remercier mes parents Marie-Pierre et Olivier, mes frères Baptiste et Paul, et mes amis.

Table of Contents

List of Acronyms

ACK Acknowledgement

AI Artificial Intelligence

AP Access Point

AR Augmented Reality

CDN Content Delivery Network

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

DAG Directed Acyclic Graph

DC Datacenters

DCF Distributed Coordination Function

DIFS Distributed Inter Frame Space

GHG Greenhouse Gas

IoT Internet of Things

ICT Information and Communication Technologies

ITU International Telecommunication Union

LCA Life Cycle Assessment

M2M Machine-to-Machine communications

MAC Medium Access Control

MCS Modulation and Coding Scheme

MIMO Multiple-Input Multiple-Output

MPI Message Passing Interface

nDC nano Datacenter

NIC Network Interface Card

PER Packet Error Rate

PUE Power Usage Effectiveness

RTS/CTS Request To Send/Clear To Send

Rx Reception

SIFS Short Inter Frame Space

SNR Signal to Noise Ratio

STA Stations

TCP Transmission Control Protocol

Tx Transmission

UDP User Datagram Protocol

VM Virtual Machine

VR Virtual Reality

Résumé en français

Contexte

Les Technologies de l'Information et de la Communication (TIC) permettent la génération, la distribution, le stockage et le traitement de données. Afin de répondre aux besoins croissants des utilisateurs d'Internet, les infrastructures associées évoluent constamment. Le Cloud Computing offre à ses utilisateurs une quantité importante de ressources au sein de Datacenters. Cependant, certaines applications comme la réalité virtuelle présentent de nouveaux défis, comme la nécessité de latences très faibles. Les datacenters Cloud, regroupés dans un nombre limité de zones géographiques, ne permettent pas de répondre à ces exigences. C'est pourquoi de nouvelles approches complémentaires au Cloud sont proposées, comme le Fog Computing. Avec le Fog, des ressources de calcul et de stockage sont ajoutées en bordure de réseau, dans des micro-Datacenters. Cela permet une réduction des temps de communication pour les utilisateurs qui communiquent avec des machines plus proches géographiquement. Cependant, les micro-Datacenters ont des capacités de calcul, de communication et un rendement énergétique très variables. Il est ainsi nécessaire d'évaluer les performances, mais aussi l'impact de ces infrastructures sur l'environnement.

En effet, les coûts des infrastructures TICs sont importants. Différentes études estiment que les émissions de gaz à effet de serre liées aux TICs représentent entre 1.9% et 3.8% des émissions mondiales [1]. Ces estimations varient du simple au double à cause de grandes incertitudes quant aux émissions des TIC durant les différentes phases de leur cycle de vie. Pour les phases de fabrication et de recyclage, des méthodologies telles que l'Analyse de Cycle de Vie (ACV) permettent des estimations se basant sur des bases de données accessibles à tous. Cependant, il est très complexe d'estimer les émissions durant la phase d'usage en l'absence d'une méthodologie de mesure rigoureuse. Ces estimations permettraient pourtant aux fournisseurs de services Cloud et Fog d'optimiser leurs dépenses, et à leurs utilisateurs de réduire les coûts de déploiement et de gestion de leurs applications.

Plusieurs approches peuvent être utilisées pour tenter d'évaluer ces coûts. Certaines permettent d'obtenir des résultats à grande échelle peu précis, alors que d'autres approches

plus fines ne permettent que d'évaluer des infrastructures de taille limitée. Par exemple, les estimations des émissions mondiales reposent sur des modèles analytiques calibrés à l'aide de données obtenues à plus petite échelle. Mais des données de calibration réalistes sont complexes à obtenir et peuvent biaiser les résultats. De nombreux facteurs doivent être pris en compte : applications, matériel, technologies de communication utilisées, infrastructure réseau, entre autres.

D'autres résultats peuvent être obtenus à l'aide d'expérimentations in-situ. Dans ce cas, les scientifiques utilisent des infrastructures réelles, et des outils de mesures comme des wattmètres physiques ou logiciels. Cependant, cette approche est limitée par les infrastructures existantes, ne proposant pas forcément le matériel de mesure nécessaire. D'autres expériences se font à l'aide de plateformes dédiées aux expérimentations (testbed). Ces testbeds permettent une meilleure reproductibilité des résultats, avec les mêmes limites matérielles. Enfin, les expériences sur de longues durées nécessitant de nombreuses ressources sont limitées par leur coût économique et environnemental.

Une dernière approche est la simulation. En utilisant des modèles de simulation, les scientifiques peuvent étudier un système du monde réel de manière reproductible. Il existe une littérature foisonnante sur les modèles de simulation des infrastructures TIC et de leurs émissions : des modèles réseau pour différentes technologies de communication, des modèles d'application et de matériel. Cependant, la plupart de ces modèles fonctionnent en autonomie et ne considèrent pas les interactions possibles entre différents composants des infrastructures distribuées. De plus, le niveau de granularité des modèles permet rarement d'étudier des infrastructures de grande taille à cause du coût en temps et en calcul pour exécuter ces simulations. Une approche intéressante serait de regrouper des modèles spécialisés dans l'étude des émissions des infrastructures de grande taille au sein d'un même outil de simulation.

Problématique

La taille, l'hétérogénéité et l'évolution rapide des infrastructures Fog rendent difficile l'évaluation de leur impact sur l'environnement. Différentes approches sont utilisées dans la littérature pour mieux les comprendre, mais nous observons que la plupart ne permettent pas d'obtenir des mesures reproductibles sur des infrastructures de grande taille. Il est possible d'adapter les modèles de simulation en fonction du but des expériences, entre le niveau de précision des modèles et leur passage à l'échelle. Ainsi, la simulation pourrait

être utilisée pour étudier des infrastructures de bout-en-bout, de l'utilisateur jusqu'aux machines hébergées dans les datacenters. En se basant sur les travaux existants et sur nos contributions, nous tentons de répondre à la question suivante :

Quel est le compromis entre le passage à l'échelle et la précision des modèles de simulation pour estimer l'impact environnemental des infrastructures Fog et de leurs applications ?

Pour répondre à cette problématique, notre approche consiste à développer des modèles de simulation pour évaluer la performance et la consommation d'énergie de différents éléments des infrastructures TIC. Ces modèles doivent permettre d'étudier les relations entre différentes technologies. Par exemple, nous étudions des infrastructures Fog utilisant des communications filaires en Ethernet dans le coeur du réseau et sans fil via Wi-Fi au niveau des utilisateurs. Afin d'obtenir des résultats réalistes, il est également nécessaire de proposer une méthodologie de calibration des modèles visant à simplifier la simulation d'infrastructures réalistes. La combinaison des modèles et méthodologies proposés permet d'étudier la consommation énergétique et les émissions d'une infrastructure fog réaliste exécutant des applications réelles.

Contributions

La première contribution de cette thèse est l'extension et la validation d'un modèle de simulation des communications Wi-Fi. Ce modèle permet d'étudier des réseaux de grande taille en représentant les communications en tant que flux de données. Ce modèle peut être combiné avec d'autres modèles de simulation pour les communications filaires, permettant l'étude de réseaux hétérogènes. La validation de ce modèle montre que sa précision est proche de modèles de l'état de l'art reposant sur une modélisation plus fine des communications. En même temps, notre modèle permet un meilleur passage à l'échelle en réduisant grandement la durée des simulations.

En se basant sur le modèle de communication Wi-Fi, notre seconde contribution est un modèle de mesure de la consommation de l'énergie utilisée par les cartes réseau Wi-Fi. En comparant ce modèle aux prédictions de consommation énergétiques faites par le simulateur ns-3, nous montrons que notre modèle fourni des estimations précises tout en réduisant considérablement les durées et l'utilisation mémoire des simulations.

Notre troisième contribution est un modèle d'application microservices pouvant être calibré de manière semi-automatique grâce à des traces d'exécutions de l'application à

étudier. Cette approche nous permet de considérablement réduire le travail d'instanciation des modèles comparé à d'autres travaux de l'état de l'art. Nous comparons la précision des prédictions de notre approche à plusieurs benchmarks d'applications microservices de la littérature, exécutés sur une plateforme expérimentale.

Enfin, nous finissons en étudiant, à l'aide de nos modèles, la consommation d'une infrastructure Fog réaliste de bout-en-bout exécutant une application microservice. Nous montrons que la configuration de l'infrastructure et des applications qui l'utilisent peuvent fortement impacter la consommation d'énergie, et les performances des applications.

Chapter 1

Introduction

1.1 Context

ICT infrastructures provide computing resources interconnected using various communication technologies for the creation and storage of data, the execution of applications, and the distribution of network services. Infrastructures continuously evolve to meet the demands of present-day Internet users. The development of cloud computing led to centralizing large amounts of computing and storage resources within large Datacenters (DC). However, emerging applications such as IoT and video streaming create novel network constraints such as ultra-low latency requirements. Such constraints require switching from a centralized data storage and processing cloud model to a more distributed approach. Fog computing considers small computing facilities at the edge of networks. Distributing data and processes across different geographical locations can help to reduce the latency between data producers and processing facilities. But fog infrastructures are more heterogeneous than clouds: fog DC nodes have varying processing capacity and energy consumption, while the connectivity between the fog and its users can rely on either wired or wireless communication technologies such as Ethernet and Wi-Fi.

Despite their benefits, ICT infrastructures have a cost. The increasing demand, the number of devices, and their limited lifetime raise concerns about their impact on the planet. As an example, the 2015 Paris Agreement [2] to limit global temperature rise to 1.5°C requires a reduction of GHG emissions by 50% by 2030, down to net zero by 2050. In this context, the impact of ICT on the world's GHG emissions is subject to debates [1]. While some studies consider ICT as a source of additional emissions, others consider it as one of the major levers to decrease global GHG emissions. Current estimations of the impact of ICT are comprised between 1.9% and 3.8% of the world's GHG emissions [1]. In France, ICT accounts for 10% of the total electricity used and 2.5% of the total energy consumption according to a 2022 report from the ADEME and ARCEP [3]. This impact is non-negligible and can be compared to the aviation sector, estimated to be 2% of the

world's emissions in 2016 [4]. Estimating this impact is a complex process that requires studying the cost of all life-cycle steps of devices from DC servers to the end-users.

Evaluating the impact of manufacturing and recycling phases of devices is possible using publicly available Life Cycle Assessment (LCA) databases. Estimations of the impact during the use phase is a difficult task due to the variability that arises from varying device usage patterns and the absence of a standardized assessment methodology. Nevertheless, estimating the impact during the use phase can significantly benefit cloud and fog providers as they allow for the comparison of diverse hardware and network configurations. Similarly, these estimations can also assist service providers to optimize their resource allocation strategies and to reduce GHG emissions and economic costs.

Existing works rely on varying assumptions and estimation methodologies. These diverse methodologies are a result of the balance between achieving scalability for large-scale measurements while ensuring accurate results. The assessment of the world's impact of ICT relies mainly on analytical models. These models are calibrated using data derived from smaller-scale measurements and then extrapolated to estimate the outcomes at a global scale. However, the values used as input for analytical models are complex to obtain. Many factors must be considered to provide valuable information: different types of applications, heterogeneous hardware, communication protocols, and network architectures, among others.

To obtain these values, one approach is to execute in-situ experiments. In-situ experiments leverage existing platforms and monitor real applications using techniques such as hardware or software wattmeters. However, this approach suffers from technical limitations such as the availability of power sensors in a limited number of hardware components (CPU and memory), and they are limited to the study of the platform's hardware. Other works leverage experimental testbeds to deploy experiments using real hardware and applications. Again, the results obtained with testbeds are limited by their hardware configuration, and compared to production environments they rely on the use of artificial loads instead of real users. Additionally, modern ICT infrastructures involve large numbers of nodes and applications. The scalability of real experiments is limited by the size of the platform under study and the cost of running experiments in platforms representative of the target environment.

To overcome these limitations, scientists make heavy use of simulation. This approach can be used to study real-world systems in a reproducible manner. Different models can be combined to estimate the energy consumption of network devices and processing

nodes. Extensive literature is available on the simulation of different parts of ICT infrastructures: network models for various communication technologies or execution models for different types of applications. But most models only focus on a single part of the ICT infrastructures. Additionally, while some models provide accurate results, their granularity requires heavy computations that increase simulation times and reduce the size of the platforms under study. Finally, existing models are often implemented in distinct frameworks. The availability of valid communication, execution, and energy models in the same framework could ease the study of the interdependencies between different parts of the infrastructures.

1.2 Research problem

The size, heterogeneity, and uncertain evolution of ICT infrastructures make complex the estimation of their impact. Several approaches enable estimations of this impact, such as in-situ experiments, testbeds, emulation, and simulation. With simulation, models of various levels of granularity can be used. However, studying the tradeoff between the results' accuracy and the scalability of the simulations is necessary to obtain sound results. Based on already developed models from the literature and our contributions we try to answer the following question:

What is the tradeoff between the accuracy and the scalability of simulation models to estimate the impact of large-scale end-to-end ICT infrastructures running real applications?

To answer this research question, our approach is to design simulation models to evaluate the performance and energy consumption of the different parts of large-scale networks. To study the interactions between heterogeneous devices and diverse communication technologies, these models' designs must be compatible with one another. In this thesis, we focus on wired communications with Ethernet and wireless communications using Wi-Fi. The choice of Wi-Fi communications is motivated by their popularity compared to other wireless technologies, as stated in a Cisco report *"there will be nearly 628 million public Wi-Fi hotspots by 2023, up from 169 million hotspots in 2018, a fourfold increase"* [5]. Since the calibration of simulation models can be complex, the design of our models has to enable calibration using a well-defined methodology to replicate the behavior of real-world systems. The combination of the proposed models properly calibrated will enable studying the energy consumption of realistic fog infrastructures running distributed

applications.

1.3 Contributions

The contributions of this thesis are the following:

— The extension and validation of a Wi-Fi performance model based on flow-level simulation. This model implemented in SimGrid eases the simulation of networks composed of Ethernet and Wi-Fi communications at scale.

— The design and validation of an energy model for Wi-Fi devices. This model is built using the flow-based Wi-Fi performance model and enables estimating the energy consumption of Wi-Fi infrastructures at scale. The validity of this contribution is evaluated against another model from the literature.

— The proposition of a microservice execution model and a methodology to semi-automatically transpose real microservice applications into their simulation equivalent using application traces. We assess the validity of this contribution by comparing simulated executions of a microservice benchmark to real executions.

— The combination of performance and energy models from the literature and our contributions to study the performance and GHG emissions of an end-to-end fog infrastructure running a microservice application. This study compares the energy requirements of different deployment policies and their impact on application performance.

1.4 Organization of the document

In Chapter 2, we present the state of the art of the works related to this thesis. We start by reviewing studies estimating the current and future impact of ICT, and we discuss the challenges explaining the uncertainties in these works. Then we describe the large-scale network infrastructures studied in this work and the applications they execute. Finally, we introduce the principal performance evaluation methodologies before focusing on existing simulation models to evaluate end-to-end ICT infrastructures.

In Chapter 3, we extend a model to estimate the duration of Wi-Fi communications to study communications in fog infrastructures more accurately. Then we validate the model in several scenarios. This model uses flow-level simulation, and we show the ability of this approach to scale while providing close accuracy compared to more fine-grained

network simulation approaches.

In Chapter 4, we propose an energy model for Wi-Fi devices based on the Wi-Fi performance model. We compare the accuracy of the predictions and the scalability of this model to the predictions of the Wi-Fi energy model available in the ns-3 simulator.

In Chapter 5, we propose a model to simulate realistic microservice applications. We introduce a methodology to transpose semi-automatically real applications into simulators using application traces. We show the ability of this approach to provide accurate application performance metrics compared to real microservice benchmarks executions.

In Chapter 6, we combine these models and other models from the literature to simulate a microservice application deployed in a fog infrastructure. We show the ability of our approach to provide several end-to-end metrics about the application's performance, energy consumption, and the GHG emissions of the infrastructure.

Finally, Chapter 7 concludes this work and provides research directions to improve further the study of these large-scale infrastructures.

1.5 Publications

— "Automated performance prediction of microservice applications using simulation.", Clément Courageux-Sudan, Anne-Cécile Orgerie, and Martin Quinson. IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). 2021.

— "A flow-level Wi-Fi model for large scale network simulation.", Clément Courageux-Sudan, Loic Guegan, Anne-Cécile Orgerie, and Martin Quinson. Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM). 2022.

— "A Wi-Fi Energy Model for Scalable Simulation.", Clément Courageux-Sudan, Anne-Cécile Orgerie, and Martin Quinson. 24th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). 2023.

Chapter 2

State of the Art

The continuous ICT improvements characterized by significant architectural changes and frequent material replacement come at a price. The manufacturing, usage, and disposal of devices, along with the rising user demand lead to increased ICT related energy consumption and GHG emissions. The existing scientific literature reveals difficulties in enhancing the energy efficiency of ICT, ensuring accurate measurements, and aligning with political climate commitments. In this chapter, we start in Section 2.1 by motivating our work through a review of existing literature regarding the use of ICT infrastructures and their GHG emissions, highlighting the uncertainties and difficulties of performing accurate predictions. To overcome these difficulties, we review the state-of-the-art of modern ICT infrastructures in Section 2.2 from the end-user to the premises of cloud providers, emphasizing the decentralization of application towards the edge of networks. After a discussion on performance evaluation methodologies, we review the literature of simulation models adapted to the study of fog infrastructures in Section 2.3. Finally, Section 2.4 examines additional material to simulate end-to-end infrastructures at scale.

2.1 ICT: current trends and future prospects

As the quantity of interconnected devices rises, disputes emerge concerning the ecological impact of ICT infrastructures in the world's GHG emissions and their ability to lower emissions in other sectors. In this section, we study literature forecasts about the present state and the future evolutions of ICT. We highlight the challenges motivating our work to obtain accurate forecasts of environmental impacts.

2.1.1 The energy consumption and GHG emissions of ICT

In pace with an increased number of devices, the energy consumed by ICT raised to concerning levels. We review the literature on the current footprint of ICT.

Estimating the current footprint of ICT

The world's GHG emissions due to ICT [1, 6, 7] is estimated in the range of 1.8%-3.9% of the world's emissions in 2020. To obtain these estimations, studies have to consider:

Life-cycle of devices: The impact of ICT devices does not only depend on energy consumption during the use phase. The life-cycle of a device considers **a)** embodied GHG emissions (to extract materials, manufacture and transport devices), **b)** use-phase emissions (energy consumption of the device when used), **c)** end-of-life emissions (disposal, recycling). Methodologies such as LCA aid the estimation of the impact of each phase [8]. While LCA helps to estimate the footprint of devices, studies highlight the absence of up-to-date metrics due to rapid evolutions and companies' unreleased data [1, 9, 10].

Energy mix: While the electricity used by infrastructures is often identical between geographical locations, their GHG footprint can vary a lot. GHG emissions required to manufacture and run devices highly depend on the energy mix of the host countries. As an example, a device in the USA emits $388gCO2e/kWh$ [11] while the same device emits $56gCO2e/kWh$ in France [12]. Only a few contributions provide information on GHG emissions, most works only provide power usage data [13], while others lack clarity on the specific energy mix they are using.

Rebound effects: Rebound effects in ICT happen when the increase in efficiency of technologies leads to higher usage. This increased usage can, in turn, lead to an increased footprint [14]. Rebound effects are complex to evaluate given their large scope, but have been observed in ICT for the last 20 years [15]. The authors of [1] mention that many studies concluding on a reduction of the footprint due to ICT efficiency improvements such as a report from the GeSI [16] do not consider or minimize rebound effects.

In addition to these factors, the evolution of the number of devices and their applications impacts the estimations of ICT–related emissions.

2.1.2 Evolutions of ICT infrastructures

According to a report published by CISCO in 2020 [5], the number of Internet users was estimated to increase from 3.9 to 5.3 billion between 2018 and 2023. Similarly, this report predicted the number of devices per capita to increase from 2.4 to 3.9, corresponding to 29.3 billion networked devices [5]. Additional devices lead to a greater volume of communications, resulting in increased data storage and processing.

According to industry actors such as Malmodin from Ericsson Research in [1], the

rapid increase in the number of networked devices will result in near-term saturation in specific domains, smartphones in particular. While some sectors may reach saturation, others continue to grow. Machine-to-Machine communications (M2M) allows machines to communicate and process data without human intervention, while Internet of Things (IoT) equips physical objects with sensors connected to the Internet. According to ICT forecasts, M2M communications in conjunction with IoT devices are going to comprise a significant portion of the overall number of networked devices by 2030 [5, 17, 18, 19]. However, there are important uncertainties between different studies. For instance, projections of the number of IoT devices by 2025 range from 20 billion according to CISCO [5] to 100 billion according to ARM [19], a prominent IoT chip manufacturer. Despite uncertainties, studies agree on the key role of IoT and M2M in the future of ICT.

The data generated by ICT devices is propagated through network infrastructures that continuously improve to obtain larger throughput. Average fixed broadband speeds were predicted to increase from 45.9Mbps in 2018 to 110.4Mbps in 2023, according to CISCO [5]. Faster networks lead to new applications such as high-quality content streaming or low-latency cloud gaming.

Increased bandwidth also leads to bigger volumes of data. The IDC's Global Datasphere Forecasts [20] published in 2020 estimations of the *"total volume of data created and replicated worldwide"*. They forecast an almost three-fold data increase between 2021 and 2025, from 64.2 to 181 zettabytes.

The data generated by ICT devices is usually stored and processed by high-capacity Datacenter (DC) servers. A DC is an infrastructure providing end-users with networked computer servers that process, store, and distribute data. A scientific report on the energy usage of DCs in the USA [21] published in 2016 sheds light on the growth in DC sizes. This growth has led to the creation of hyper-scale DCs which are DCs *"covering up to more than 400,000 sq ft"* with efficient *"cooling systems and redundant power"*. Despite efficiency improvements to optimize DCs' resources, the estimated number of servers in the United States continuously increased from 12 million in 2008 to 18 million in 2020 according to [21]. The persistent rise in demand coupled with the recent deceleration of Moore's Law [1] tend to show that this increase in the number of servers will continue.

Evolution of the demand

Some uncertainties in ICT forecasts are due to the uncertain evolution of networked devices usages and emerging applications.

IoT devices are used in many domains: health, agriculture [22], animal populations monitoring, among others. For example, the city of Angers in France invested 178 million euros in 2019 to improve the efficiency of public services and to deploy 50,000 sensors [23]. Some sensors produce metrics a few times a day, others create much more data like video surveillance cameras. The city claims they will contribute to reducing by 30% the amount of water used for plants and by 66% the electricity used by public lights by 2026 [24]. However, little information is available on the energy used by the sensors themselves, the energy used to send and process the data over the network.

As of 2023, Artificial Intelligence (AI) is a topic of interest for researchers and the industry. A 2021 report on the evolution of AI supported by actors such as Google and OpenAI [25] estimates the growth of AI peer-reviewed publications from 40,000 in 2014 to 120,000 in 2019. Industrial AI applications are also widespread. Large-scale models such as GPT-3 [26] and Stable Diffusion [27] are trained and process requests using specialized hardware. A report from Google published in 2022 evaluates the energy consumption of AI during training and inference phases [10]. This report describes different hardware solutions, such as NVIDIA V100 Graphics Processing Units (GPUs) or custom Tensor Processing Units. End-user devices such as smartphones also include AI-specific hardware [28]. Amid an increase of AI hardware and software, the precise impact of this rapid evolution in the years to come is uncertain. While some works predict a plateau in the future carbon footprint of AI [10], most studies anticipate increased emissions [1, 5].

With cryptocurrencies and smart contracts, blockchains can induce major changes in the future usage of ICT. Blockchains require many devices to perform computations and store large amounts of data as in the case of Bitcoin's *proof-of-work* [29, 30]. Cryptocurrencies' impact on the energy consumption of ICT was estimated to account for 68.7TWh in 2020, or the equivalent of 7 million US households according to [1]. Alternative transaction validation techniques such as *proof-of-stake* could significantly reduce the energy consumed by blockchain such as Ethereum [31].

Despite the lack of a formal definition, the metaverse aims at easing communication between Internet users [32]. Large-scale adoption of this concept would impact ICT infrastructures with the use of Virtual Reality (VR) hardware, and the need for real-time, low-latency communications. Similarly, complete offloading of the games' executions to cloud DCs with cloud gaming could reshape ICT infrastructures. To reach sufficient Quality of Experience, data streams must achieve very low latency, while high-quality graphics require large bandwidth [33]. To enable such technologies, the research community stud-

ies the possibility of offloading computations to the edge of networks, reducing latencies and core network usage [32, 34], but increasing the power used by edge nodes.

While this review is not exhaustive, it illustrates the growing number of connected devices. While efficiency improvements can reduce the energy consumed by devices, the majority of forecasts concerning the impact of ICT suggest an overall increase in emissions, despite uncertainties on the precise magnitude of this rise.

2.1.3 Forecasting the footprint of ICT infrastructures

Figure 2.1 compares GHG emissions forecasts of ICT made by different studies, taken from [1]. These predictions vary by a factor of 3, with results for 2030 comprised between 1.5GtCO2e and 4.8GtCO2e. We can categorize existing literature into two groups: **a)** works that estimate ICT will enable reduced GHG emissions in the future, **b)** works that plan on an increase of GHG emissions due to ICT.

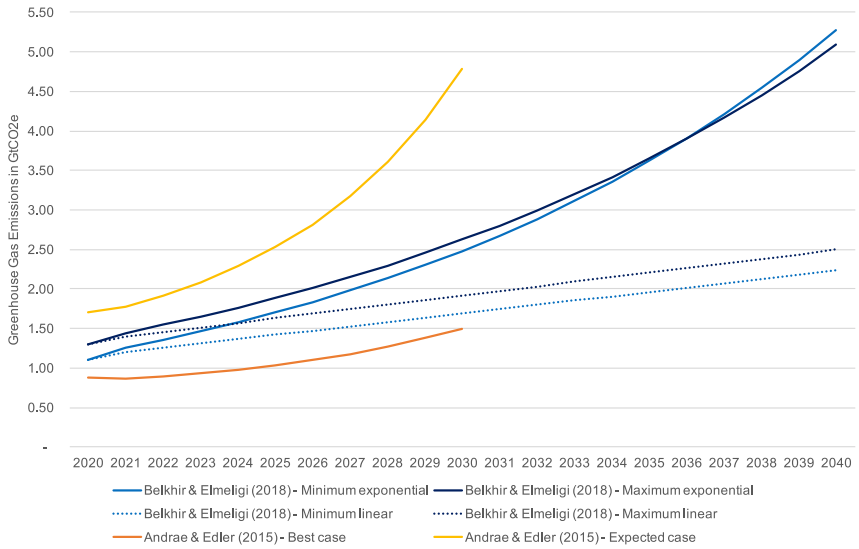


Figure 2.1 – Projections of ICT's GHG emissions from 2020. Source: [1]

In 2020, the International Telecommunication Union (ITU) published a standard [35] on the evolution of ICT to stay in line with the 1.5°C agreements with important industry actors: GeSI (AT&T, Deutsche Telekom, Huawei...), GSMA (Microsoft, Cisco, Deutsche Telekom...) [1], SBTi [36]. Many companies engaged to reduce their GHG footprint by 45% by 2030, mainly through the use of renewable energies. Moreover, ICT could also help to

1. `https://www.gsma.com/betterfuture/egdc`, last accessed December 2023

gain energy in other sectors. In 2015 the GeSI published a document [16] estimating a possible reduction of 20% of the world's GHG emissions with ICT efficiency improvements.

However, other works observe the necessity of estimating the footprint of ICT infrastructures considering different evolution scenarios, without omitting part of the infrastructures and factors such as rebound effects. Most predictions concluding on the reduction of ICT emissions do not consider any rebound effect despite their likelihood [37]. More cautious approaches need to be adopted considering all parameters [1, 15, 38].

To estimate more accurately the future footprint of ICT, some works propose to establish scenarios for the evolution of infrastructures and applications to estimate the footprint of each scenario [15, 39, 40]. Adopting this approach requires deciding on a set of scenarios representing the evolution of ICT hardware and software, and using a transparent and open methodology to estimate the GHG emissions of ICT.

ICT evolution scenarios

Based on prospective ICT studies such as [1, 5], one can estimate the evolution of the infrastructures and their GHG footprint. Then, use cases permit to study the use of these infrastructures. Experimental use-case scenarios are proposed in many works in the literature for different ICT infrastructures and applications. For example, the authors of [41] propose scenarios to study vehicular networks, while the authors of [42] propose a scenario to study the electricity demand of Japanese telecommunication networks. These scenarios enable the analysis of the interactions between different parts of infrastructures (DC, network equipment, end-user devices). However, to the extent of our knowledge, most scenarios focus on a single type of application at a limited scale, hindering the understanding of systemic effects. The lack of a methodology to evaluate the end-to-end impacts of ICT scenarios during use-phase also complexifies these evaluations.

Improving impact evaluations

Based on prospective ICT scenarios, one can estimate their energy and GHG footprint. The obtained estimations could help industrials and politics to make decisions towards more reasonable use of ICT. Although embodied and end-of-life emissions are evaluated using standardized methodologies, no standard approach exists for assessing use-phase emissions. Furthermore, considering uncertain device usage patterns adds complexity to the prospective evaluation of GHG emissions during this phase. For these reasons, solutions are proposed to evaluate the impact of ICT infrastructures and applications

from the end-users to the premises of cloud providers. There are several challenges to the production of accurate predictions during the use phase:

— Modeling the architecture of current and future end-to-end ICT infrastructures;
— Accounting for the heterogeneity between different types of devices;
— Capturing the interactions between different ICT components at scale;
— Studying versatile applications.

In this work, we try to answer these challenges. Doing so requires solutions that enable accurate footprint estimations of ICT infrastructures at scale. These solutions need to enable the study of networks mixing between wired network links and wireless technologies at the edge of the network in a reproducible manner. In this thesis, we restrain our study of network technologies to Ethernet and Wi-Fi because of their popularity. Additional technologies could also be considered as future work.

2.2 ICT infrastructures: towards the fog

The increasing number of Internet applications, large data transfers, and computationally intensive tasks require efficient infrastructures. This led to the development of large DCs and cloud computing solutions. However, applications introducing novel challenges such as the need for ultra-low latencies benefit from alternative approaches like fog computing. This section outlines the adaptations of current infrastructures and their applications to obtain lower network latency and reduce the use of the core network.

2.2.1 From Cloud to Fog computing

Cloud Computing

Cloud computing consists in the on-demand availability of computing resources (storage, processing, applications) within remote infrastructures [43]. When comparing cloud computing to self-hosting, cloud computing provides virtually infinite computing power, reduces maintenance costs, and can increase resource usage efficiency. The energy efficiency of cloud DCs depends on the usage of servers and the efficiency of additional datacenters' equipment (cooling, lighting). The impact of this additional equipment is estimated using the Power Usage Effectiveness (PUE) indicator, a ratio between the total energy consumed by the infrastructure and the energy consumed solely by ICT devices. According to a report from uptime intelligence in July 2023, *"the average annual power*

usage effectiveness (PUE) reported in 2022 was 1.55" [44]. The PUE of self-hosting can exceed 2.0, while the PUE of current hyperscale DCs approximates 1.1 according to various reports [21, 45].

Despite the ability of large-scale DCs to execute processing tasks very efficiently, emerging ICT applications requirements show the limits of the centralized cloud computing model. Challenging technical issues arise to achieve very low latencies, for example below 20ms for Augmented Reality (AR) applications [46]. The geographical distribution of DCs in a limited number of locations can impact communication latencies and exceed this threshold, especially when users are far away from DCs [47]. Similarly, transmitting the data generated by devices such as IoT to cloud servers is costly and may benefit from preprocessing to reduce the data volume sent to the cloud. Solving these issues has led to the development of an extension of the cloud model, where the execution of tasks can take place near the end users.

Edge and Fog computing

Edge and fog computing have been proposed to solve the latency and congestion issues of cloud computing [48]. In contrast to the centralized cloud model, edge computing places storage and compute resources close to the data sources. Groups of resource-limited nodes such as Raspberry PIs form a nano Datacenter (nDC). While such nDCs are less powerful and more heterogeneous than large cloud DCs, their distribution at the edge of the network reduces network latencies. This reduced latency can meet the requirements of applications such as AR as noted by the authors of [49]. Despite the advantages of edge over cloud computing, the scarcity of edge servers combined with their reduced processing capacity makes difficult the execution of compute-intensive tasks compared to clouds. Similarly, small-scale nDC infrastructures are less optimized than large hyperscale DCs, reducing their energy efficiency [21].

Compared to edge computing, fog computing creates several layers of computing resources between the data source and the cloud DC [48]. Figure 2.2 illustrates a fog infrastructure composed of three layers: **a)** end-users producing data and connected to the network using wireless links, **b)** fog DCs located at a short distance to the users, **c)** the cloud DC that can be anywhere in the world. These layers enable the use of low-latency edge resources and efficient cloud DCs simultaneously. Processes can be placed depending on application requirements, between low latency but resource-constrained nodes and fast executions of tasks on far-away servers [50]. While edge computing is

already widely adopted, for instance, to cache videos using the Akamai Content Delivery Network (CDN) [51], real-world use cases for fog computing are mostly experimental [52].

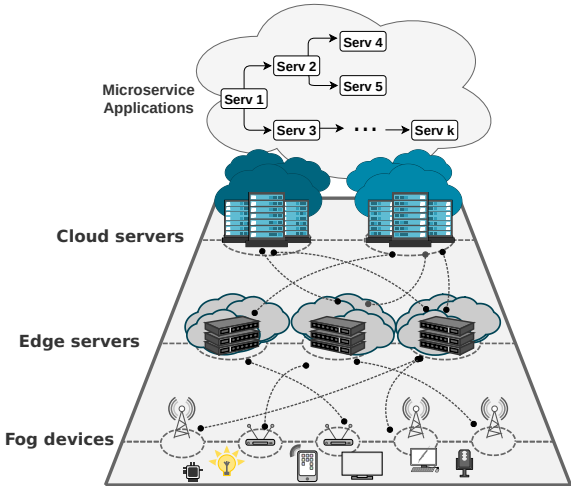


Figure 2.2 – Overview of the actors of a fog infrastructure. Inspired from [53].

Edge and fog computing tackle the latency and network congestion issues of cloud computing. However, they also bring new challenges. The geographical distribution of servers requires separating the applications between compute-intensive tasks in cloud servers and smaller tasks close to the data sources. In this context, applications are distributed, despite uncertainties on the tradeoffs between optimized performance and energy consumption.

Microservice applications: distributed and scalable

Modern applications can benefit from the design of fog infrastructures. A prevalent approach is to transform monolithic applications into interconnected microservices. The authors of [54] characterize microservice applications as a set of independent services, where each service can be maintained, deployed, and tested without modifying the others. Services can be developed independently, reducing the development costs and duration. When a service receives a request, it can be propagated to other services situated on separate nodes. Request executions in a microservice application form a Directed Acyclic Graph (DAG) of function calls, as shown in Figure 2.3 for a synthetic application. Table 2.1 summarizes the main differences between microservice and monolithic applications' architectures.

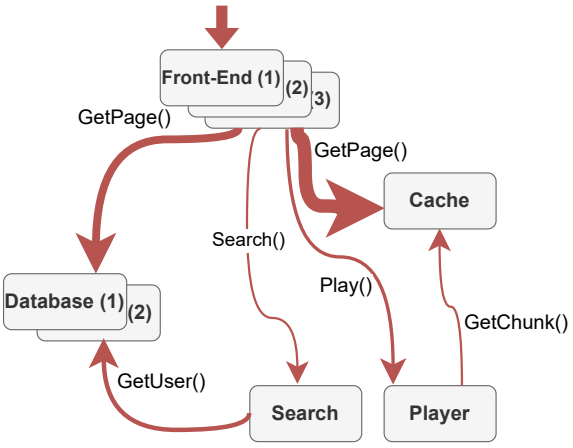


Figure 2.3 – Synthetic example of the DAG of a microservice application

Prominent Internet actors embraced microservices, such as Twitter, Netflix, Uber, and WeChat [55]. While the exact architecture of popular microservice platforms remains undisclosed, reports suggest that Netflix employs over 600 services, Uber utilizes more than 1,000 services, and WeChat incorporates over 3,000 [56].

Although individual microservices are simple and lightweight, multiple services are combined to fulfill a single request. It results in intricate network interactions and additional processing overhead for each service. Additionally, cautious orchestration of the application is necessary to obtain good performance, as we see in the next section.

Table 2.1 – Comparing monolithic and microservice applications, inspired by [57, 58, 59]

Monolithic application	Microservice application
Single large application	Independent light services
Complex application architecture	Light interfaces between services
Full application replicas	Individual service replicas
Full application deployment	Independent service deployment
Execution of requests in a single location	Network communications between services
No network overhead	Overheads between services
Few dependencies	Many dependencies (orchestration)

Performance of cloud and fog applications

The emergence of fog, edge computing, and microservice applications has mitigated some constraints associated with traditional clouds. Nevertheless, there is an ongoing

debate regarding the precise advantages and potential drawbacks of these approaches on performance and energy consumption.

Some studies [57, 58] compare the performance of monolithic and microservice applications deployed in cloud environments. These studies conclude on the benefits of microservice replication at the service level, which can handle larger volumes of requests with reduced failures compared to monolithic applications. However, these contributions do not precisely measure the overhead of microservices compared to monolithic applications. Network messages have to be received and processed by each service, while a monolithic application only receives requests once.

The authors of the *DeathStarBench* microservice benchmark [60] compared the ratio of time spent processing network requests in monolithic and microservice deployments of the benchmark. While execution times are faster with microservices, the time spent by services processing network requests is three times higher than with a monolithic application (2ms in the monolithic deployment against 6ms for microservices). The authors of [61] also study the overhead of microservices, showing that the context switching and network operations overheads depend on the number of services.

Another debate in this context is the real impact of fog infrastructures and applications on energy consumption and GHG emissions. Since the fog requires the deployment of many small nDCs, its impact in all stages of the life-cycle of devices could be major. In this section, we only focus on use phase emissions.

The authors of [62] propose a theoretical model to evaluate the fog's latency and energy improvements compared to cloud computing. Their results show that a fog approach can help reduce the response time of their application and the energy consumed to transmit and process data by *"40.48% compared to the conventional cloud computing model"* [62]. However, the proposed model needs precise calibration and could be completed. This model does not consider **a)** heterogeneous communication and processing technologies, and **b)** the PUE despite its higher value in fogs than in clouds [21].

Similarly, the authors of [63] simulate microservice applications in the fog. Again, fog deployments give better performance and energy consumption than a cloud environment. However, no information is available on the PUE of the infrastructure, we have no information on the energy consumed by the core and access networks, and the application is small-scale (5 services). Finally, the simulation models used do not differentiate between communication technologies (wireless in the edge and wired in the core).

Other works regarding fog and edge energy consumption have similar limitations. The

authors of [50] reviewed the experimental methodologies of 99 fog placement publications. Their analysis states that: **a)** the evaluated platforms are often self-generated by the authors of the papers with no justification (77/99), **b)** the infrastructure's parameters (latency, bandwidth, CPU, memory), were generated with no justification for 61 papers, while **c)** only 19 experiments evaluated their contribution with more than one cloud datacenter and 100 fog nodes. This survey highlights the need for realistic networks and application evaluations at scale without omitting parts of the infrastructure. We also note the importance of some power parameters (PUE) that can impact the results.

In addition to end-to-end infrastructure parameters, the authors of [64] provide information on the overhead of the kubernetes [65] orchestrator for geo-distributed fog cluster federations. The results show that the processing of data and the number of metrics sent to the monitoring servers can be important within large fogs (27MiBps of cross-cluster traffic in their evaluation). This can be a problem for network-constrained environments.

To overcome missing end-to-end metrics for the energy consumed by fog and edge applications and the limited scalability of most contributions, the next section compares the evaluation methodologies to obtain end-to-end energy and performance metrics for large-scale IT platforms.

2.2.2 End-to-end performance evaluation methodologies

Scientific contributions in computer science evaluate their results using different approaches. The advantages of different methodologies depend on the goal of the research questions under study, as explained in [66, 67, 68]. We compare three experimental methodologies in this section: real experiments, emulation, and simulation. We compare their ability to answer our research question: estimating the energy consumption of end-to-end large-scale infrastructures. We base our comparison on the tradeoffs between:

— **Accuracy:** As detailed in [66], experiments are performed to answer a specific research question. The research question under study thus determines the necessary level of accuracy. For instance, comparing the communication bandwidth between different Wi-Fi standards necessitates very fine-grained metrics, while studying network bandwidths in a large-scale end-to-end application may not require the same level of granularity. Different methodologies enable different levels of granularity.

— **Scalability:** The evaluation of computer systems is possible at different scales. While very fine-grained experiments provide accurate outputs, investigating the footprint of networks at scale poses significant challenges when relying on fine-

grained approaches. Fined-grained studies at scale have prohibitive costs and require using large amounts of resources over long timespans.

— **Reproducibility:** The term reproducibility has many definitions at different degrees [69]. An evaluation is reproducible when the authors provide enough information to allow others to run and observe the same conclusions. Reproducibility consists mainly of the transparency of models, a clear definition of the evaluation methodology and scenarios, available datasets, and open access to the hardware and software used.

One experimental approach cannot maximize these three properties simultaneously. While accurate experiments tend to have limitations in terms of scalability, scalable experiments are challenging to reproduce and offer broader, less detailed results. The optimal balance between those properties depends on the objectives of the evaluation [66, 67].

Real-world experiments

Experiments can be conducted by deploying scenarios on real hardware and monitoring application- and system-level metrics. Real experiments can run on production infrastructures equipped with monitoring tools or with the help of experimental testbeds. Testbeds are platforms built specifically for the study and validation of scientific questions. They provide computing resources, monitoring devices like Wattmeters, and software stacks. Compared to production environments, testbeds give large control to their users, making results more reproducible. Conversely, concurrent platform users can impact the results of experiments running on services such as Amazon's AWS or Microsoft Azure clouds.

When physical wattmeters are not available, an alternative is to use software-based powermeters for different devices' components. The authors of [70] review different techniques and tools to measure the power usage of CPUs, GPUs, and memory. This option mainly relies on power sensors available in the hardware. The drawback of this approach is the lack of measurement for the power consumed by other hardware components such as the network card, the entire device, and the rest of the infrastructure such as cooling.

Various scientific testbeds exist to conduct experiments in different domains. In France, Grid'5000 [71] is very popular for cloud-related experiments, with 800 nodes in 8 separate locations. More than 2,000 publications use this testbed to study topics such as the orchestration of fog applications [72]. Chameleon [73] is similar to Grid'5000, with nodes in different countries. Other testbeds such as FIT IoT-LAB [74] with more than 1,500 nodes are more specific to IoT devices, sensor networks, and mobility.

While testbeds give better control on experimental conditions compared to public clouds, there are drawbacks to this approach when evaluating end-to-end infrastructures. First, the execution of end-to-end scenarios requires interactions between separate specialized testbeds. For example, the execution of an IoT application with a back-end in the cloud may require using Grid'5000 and FIT IoT-LAB simultaneously. Second, the size of the testbeds limits the scalability of the experiments. For instance, Grid'5000 clusters are much smaller than the size of today's hyperscale DCs. They are also limited by the testbed's hardware, making it hard to study specialized devices not available on the testbed. Exploring many parameters requires extensive computing power, consumes a large amount of energy, and has a high monetary cost. Finally, the reproducibility of some experimental scenarios is very complex. For example, Wi-Fi communications can be affected by external factors such as interferences or other testbed users [75]. To summarize, real experiments provide accurate results but can be limited in scalability and reproducibility.

Emulation

Emulation consists in reproducing the behavior of a computer system within another host computer system. Different emulation solutions exist at different levels of granularity.

Hardware-level emulation with solutions such as QEMU [76] can emulate systems at the hardware level. The main benefits of such solutions are their accuracy and reproducibility. Full replication is possible with this methodology, provided the emulation software and enough computing power. However, this solution cannot simulate large-scale distributed systems because **a)** the overhead of this emulation requires much more power than the system under study, and **b)** the network topology between different nodes also needs to be emulated.

Network emulation makes it possible to study distributed systems. CrystalNet [77], Distem [78], or Kollaps [79] are network-level emulators. These emulators use real computing nodes and limit their processing and memory resources using Virtual Machines or containers. The creation of virtual routes between hosts emulates the network. The authors of Kollaps [79] managed to reproduce a distributed system composed of more than 2,600 nodes. Distem [78] uses 100 physical nodes to emulate 5,000 virtual nodes. However, these emulation techniques limit the capacity of physical nodes and network links to emulate a distributed system. Using this approach does not enable studying faster networks and larger infrastructures than the physical testbed used to run the experiments.

Alternative approaches try to overcome some of these limitations. Bonsai [80] aims at compressing networks to simplified topologies while preserving most properties. Other works simulate the network between emulated nodes like TANSIV [81] and ns-3 DCE [82].

To summarize, emulation can study both limited and large-scale infrastructures. However, the emulation of large systems requires more powerful infrastructures than the system to emulate, limiting the reproducibility of large-scale systems emulation. Emulation provides accurate results depending on the goal of the experiments, where different tools provide results at different levels of granularity.

In this thesis, we aim to study large-scale ICT infrastructures such as a fog connected to a large cloud data center. These infrastructures require the evaluation of thousands of nodes and network links. In this context, emulation methods based on limitation techniques are very expensive to use during long timespans and require access to extensive computing resources, limiting the scalability of this approach.

Simulation

A model is an abstract representation of a real-world system [66]. Simulation makes use of models to evaluate and validate or invalidate scientific questions. Different models can simulate the same phenomena but do not rely on the same hypotheses. Before using a model, the authors of [66, 67, 68] advise understanding the validity limitations of existing models and choosing a model corresponding to the hypotheses of the research question. We differentiate models by their level of granularity. We observe a tradeoff between fine-grained models that are too expensive to study large infrastructures and coarse-grained models with less precise results but better scalability. In the following, we compare three different levels of granularity for network communications simulation: link, packet, and flow-level models. Similar levels of granularity occur for other models with the same advantages and drawbacks (instruction- and system-level models for CPUs).

Link-level simulation: Link-level models can simulate network communications down to the physical layer. As explained in [83], link-level models reproduce the physical phenomena of digital data transmission: modulation and demodulation, channel access, signal loss, and the configuration of emission and reception equipment. Usually, this low-level approach uses MATLAB models, such as the WLAN toolbox for Wi-Fi communications [84]. Despite the accuracy of these models, *"the runtime complexities of these simulators limit their suitability to single link simulation"* [83]. Thus, the scalability of link-level models does not allow the simulation of scalable ICT infrastructures.

Packet-level simulation: Packet-level models abstract the physical layer and model communications at the granularity of network packets. Packet models are used to model network stacks, for instance, to compare the performance of network protocols. Packet-level models can benefit from link-level results to simplify physical phenomena, such as in the Wi-Fi models of the ns-3 network simulator where channel errors are estimated using a lookup table produced with a link-level model [83, 85]. The authors of [83] explain that this approach is more scalable than link-level models and can *"simulate networks with hundreds of nodes with reasonable runtime"*. However, the simulation of large-scale networks using packet-level models has shown to be resource intensive, simulations requiring several hours of runtime with important memory usage [86, 87]. To this extent, the scalability of packet-level models does not permit answering the research question of this thesis. Nevertheless, similarly to link-level models' results to calibrate packet-level models, a more coarse-grained simulation approach could benefit from packet-level models' information to obtain more scalable results.

Flow-level simulation: Flow-level models represent network communications at the granularity of network packets. They do not represent every packet going through the network but represent sets of network packets as communication flows. Network flows are comparable to fluids moving in pipes instead of many network packets. Similarly to packet-level compared to link models, flow models abstract part of the network communications, and can use the results of more fine-grained models. For example, flow-level simulation frameworks such as SimGrid [88] can use the results of packet-level models to abstract channel-sharing mechanisms while keeping credible outputs. However, cautious validation is required to assess the validity and the accuracy limits of these models. Regarding scalability, flow models enable the simulation of more large-scale networks with lower resource usage than packet-level models. Some flow-level models have been used to study networks with thousands of nodes within limited runtimes [87, 89, 90].

Table 2.2 – Comparison of performance evaluation methodologies (✔✔ very good; ✔ good; ∼ variable; ✘ bad)

Methodology	Accuracy	Scalability	Reproducibility
Real experiments	✔✔	✘	✘
Emulation	✔	∼	✔
Simulation	∼	∼	✔✔

In this section, we compared the advantages and drawbacks of simulation, emulation, and real experiments to study the energy consumed by networks and applications at scale. A summary of this comparison is in Table 2.2. We observe the possibility of designing simulation models depending on the question under study to match scalability and accuracy requirements. In comparison, emulation and real experiments can provide more realistic results but suffer from limited scalability. For this reason, in this work, we focus on simulation models to estimate the energy consumption of fog networks and applications at scale. Flow-level models seem a reasonable approach between packet-level models that are too fine-grained to simulate large infrastructures and analytical models that are too coarse-grained to study some network and application behaviors.

2.3 Simulation frameworks to estimate fog performance and energy consumption

In this section, we review state-of-the-art simulation frameworks for the simulation of different parts of ICT infrastructures and the missing contributions towards end-to-end infrastructures' simulation. We start by describing network simulation tools with different levels of granularity in Section 2.3.1. In Section 2.3.2 we focus on tools for the simulation of task executions and the architecture of applications. Then, we review tools to evaluate energy and GHG emission in Section 2.3.3.

2.3.1 Network simulation tools

We review existing simulation tools from the most fine-grained to the more coarse-grained solutions. We focus on simulation tools for Wi-Fi and Ethernet communications, even if other technologies can be simulated by some of the presented frameworks.

Matlab's WLAN Toolbox [84] is used to simulate IEEE 802.11 network links down to the physical layer. It models the physical transmission of data including waveforms, signal modulation, and demodulation, and has detailed propagation models. These models provide very fine-grained results but suffer from important scalability limitations. As an example, the simulations of a single Wi-Fi link in the WLAN Toolbox compared to ns-3 packet-level simulations in [91] are slower by a factor of 46 (60 minutes against 77 seconds). This approach is preferred to simulate single network links as explained in [83].

Ns-3 [92], OMNET++ [93], and Komondor [94] are packet-level network simulators.

Compared to link-level models, they can simulate in a few minutes scenarios that take hours with link models as shown in [83]. While ns-3 and OMNET++ are general-purpose packet-level frameworks, Komondor focuses on IEEE 802.11ax communications. The models of these frameworks have been developed and validated. For example, the Wi-Fi models of ns-3 have been validated against link-level simulation in [83, 85], and Komondor has been validated with identical results compared to ns-3 and analytical models in [94]. In addition to Wi-Fi models, ns-3 and OMNET++ propose models for wired communications and other wireless technologies (e.g. 3G, LoRA). Ns-3 is a very active simulator in research, used by *"thousands of publications to date"*[2], and is actively maintained. Despite better scalability than link-level models, packet-level simulators still suffer from scalability issues when simulating networks with many nodes because of an important computational cost, as shown in [53, 86].

Simgrid [95], Narses [89], and FLEO [90] are flow-level network simulation frameworks. FLEO is an OMNET++ extension for the study of CDN networks without wireless nodes, and Narses only provides models for wired communications. SimGrid is more versatile, with wired network models validated in [88, 96] against a packet-level simulator, and a Wi-Fi model introduced in 2021 in the thesis of Loic Guegan [53]. The scalability of flow models is better than packet-level frameworks. SimGrid can be used for experiments with thousands of nodes [86], Narses speeds up 45 times compared to packet-level simulation [89], while FLEO was validated with scenarios comprising 3,515 nodes [90]. To the extent of our knowledge, no other flow-level simulation frameworks than SimGrid propose validated Wi-Fi models. Additionally, this model is limited to the simulation of Wi-Fi in ideal channel conditions according to [53], and its validity limitations were too important to be used in fog scenarios before this thesis.

ClouSim Plus [97] and the fog-specialized frameworks IFogSim [63, 98] and YAFS [99] propose network models for the simulation of cloud and fog communications. However, the accuracy of the simulated network communication times of CloudSim (in the core of IFogSim) is questioned in [100] where channel sharing is shown to be invalid. For these validity concerns, we do not consider CloudSim-based simulators to study fog communications in this thesis.

A summary of the network simulators presented in this section is proposed in Table 2.3. We observe that flow-level models propose good scalability properties, but are limited by the lack of a Wi-Fi model adapted to the simulation of fog environments.

2. `https://www.nsnam.org/research/`, last accessed December 2023

Table 2.3 – Comparison of network simulation frameworks (✔ yes; ✘ no; ⚡ contribution of this thesis).

Simulator	Network			
	Wired	Wi-Fi	Energy Wired/Wi-Fi	Scalability
WLAN Toolbox [84]	✘	✔	✘/✘	Link-level
ns-3 [92]	✔	✔	✔/✔	Packet-level
OMNET++ [93]	✔	✔	✔/✔	Packet-level
Komondor [94]	✘	✔	✘/✘	Packet-level
Narses [89]	✔	✘	✘/✘	Flow-level
FLEO [90]	✔	✘	✘/✘	Flow-level
CloudSim Plus [97]	✔	✘	✔/✘	Packet-level
IFogSim [98, 63]	✔	✔	✔/✔	Packet-level
YAFS [99]	✔	✔	✔/✔	Queue-based
SimGrid [95]	✔	⚡	✔/⚡	Flow-level

2.3.2 Task processing simulation tools

Computing nodes need to process the data they receive. In this section, we review models for the execution of tasks. We start by reviewing existing resource usage models (CPU, memory). Then we focus on models for the simulation of distributed applications.

Hardware execution

The gem5 simulator [101] is extensively used to perform cycle-accurate simulations of CPU tasks. Similarly to link-level models that simulate network communications down to physical phenomena, cycle-accurate models can be used to compare processor microarchitectures. Despite the accuracy of the results of this simulator, it is not adapted to the study of large-scale systems with compute-intensive applications due to the computational costs of the models. As written in [102], one second of simulated time in gem5 can require thousands of seconds of simulation time without reducing the simulations' fidelity.

More coarse-grained simulators such as SimGrid [95] can model the execution of tasks on simulated nodes. In this case, CPU executions are modeled depending on the number of CPU cores on the nodes, the frequency, and the usage of each core. SimGrid models have been used to simulate different applications at scale. More than 3,000 Message Passing Interface (MPI) processes are simulated in [103] and 12,000 cloud VMs within two hours in [104]. But SimGrid also has some limitations. No memory model is available,

and the simulation of realistic distributed applications such as microservices requires a lot of work to instantiate correctly all the models.

CloudSim plus [97] and fog-specialized simulators such as YAFS [99] and IFogSim [63, 98] also simulate the execution of tasks with good scalability. The execution models are based on queues and can be used to simulate cloud computing environments including task executions, Virtual Machine (VM), and scaling algorithms. Similarly to their network models, the validity of the execution models of CloudSim is questioned in the literature. The authors of [105] declare *"Unfortunately, the result was very different in terms of simulated timespan. CloudSim could not even be close to it"* when comparing the results of different simulators, showing that the start time of tasks is not realistic in CloudSim.

Table 2.4 compares the features of the task execution models of this section. We can observe that no *ideal* framework exists that combines fast simulations using valid models with fine-grained results. However, in the context of this thesis, we observe SimGrid as a good compromise between the fine-grained execution models of gem5 and the coarse-grained models with validity limitations of CloudSim.

Table 2.4 – Comparison of state-of-the-art simulators for task executions and distributed applications (✔ yes; ✘ no; ⚡ contribution of this thesis).

Simulator	Network		Computing		Energy		Domain
	Wired	Wi-Fi	CPU	Memory	Network	Nodes	
gem5 [101]	✘	✘	✔	✔	✘	✔	Processes
ns-3 [92]	✔	✔	✘	✘	✔	✔	Network
OMNET++ [93]	✔	✔	✔	✔	✔	✔	Network
CloudSim Plus [97]	✔	✘	✔	✔	✘	✔	Cloud
DISSECT-CF [106]	✔	✘	✔	✔	✔	✔	Cloud
μqsim [107]	✔	✘	✔	✔	✘	✘	Microservices
YAFS [99]	✔	✔	✔	✔	✔	✔	Fog
IFogSim [63]	✔	✔	✔	✔	✔	✔	Fog
BigHouse [108]	✔	✘	✔	✘	✘	✔	Cloud
SimGrid [95]	✔	⚡	✔	✘	⚡	✔	Versatile

Application architecture

Application models can be used to study the interactions between network and processing models when running real applications.

BigHouse [108] models simulate cloud applications using statistical methods based on a discrete-event simulator. Unfortunately, recent works have shown that the accuracy of BigHouse is limited for distributed applications composed of several components deployed on different nodes due to its very high-level representation of applications [107].

Based on the limits of BigHouse, the authors of μqsim [107] use a finer-grained representation of microservice applications. μqsim allows for detailed modeling of internal microservice executions using sets of execution stages for the requests received by microservices, and communication dependencies for interactions between services. Applications form a DAG where nodes are services and edges represent the path followed by individual requests processed by the application. With a correct calibration of the different execution stages of each service and their interactions for different request types, the authors manage to reproduce the behavior of a complex microservice benchmark [60]. Whereas the calibration of the models is feasible by hand for small applications, the lack of a proper calibration methodology leads to a tedious and error-prone process with large applications. This approach also requires to re-calibrate the models each time the code of a service is modified to match the application's new behavior.

Other simulators help the study of fog and IoT-specific applications. IFogSim [98, 63] can simulate microservices in fogs with mobility, scheduling, and energy models. Again, this simulator is based on CloudSim and inherits its limited validity [100, 105]. Similar works such as IoTSim-Osmosis [109] or YAFS [99] propose models specific to IoT applications in the fog. While these tools can simulate end-to-end infrastructures, their communication and execution models remain very simplistic. For instance, these simulators do not differentiate wired and wireless communications despite their different channel-sharing mechanisms, and their different energy usage.

In the context of this state-of-the-art review, we observe a tradeoff between the accuracy of the application models and the difficulty of their calibration. A methodology to transpose real-world applications into their simulation equivalent could ease the study of real applications using simulation. We observe that other domains make use of runtime traces to extract application information and simulate tasks using these traces [105].

2.3.3 Energy simulation frameworks

Based on the performance models for nodes and communications, power models can compute the energy consumed by the infrastructure. Figure 2.4 illustrates the behavior of power consumption models for ICT devices. The power consumed by a device P_{tot} is the sum of **a)** a *static power consumption* P_{stat} (the minimum power to operate the device) and **b)** a *dynamic power consumption* P_{dyn} (depending on device's activity). Depending on the devices, the calibration of power models will vary along with the computations of P_{stat} and P_{dyn}. Since power models depend on devices' usage, their results are accurate only if the durations of communications/process executions are accurate. We assume performance models are accurate in this section.

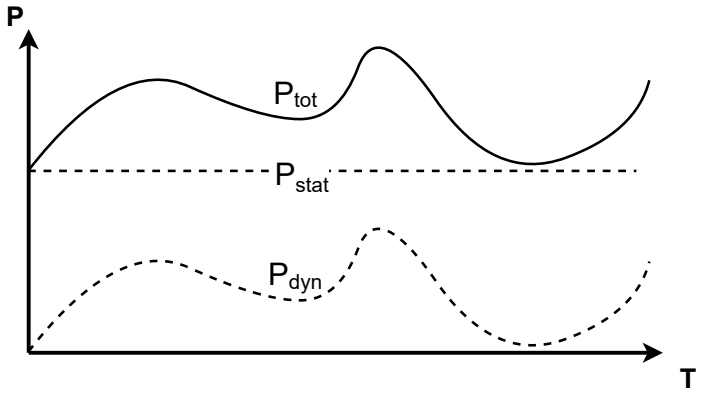


Figure 2.4 – Power usage of a device $P_{tot} = P_{stat} + P_{dyn}$

CPU power usage

The static power consumption of a node corresponds to the power used by the machine when idle. When a node processes data, the additional dynamic power usage depends on the number and utilization of active CPU cores and the frequency of the cores.

Gem5 [110] has an energy consumption model for CPUs considering instruction sets and microarchitecture implementation. Again, this type of simulation does not allow for large-scale simulation, despite providing fine-grained information.

SimGrid, IFogSim [63, 98], and YAFS [99] have models to study the energy consumed by bare metal servers and VMs. Properly calibrated, these models use linear regression to estimate the nodes' energy consumption depending on their average usage. SimGrid model was used to measure the energy consumption of cloud nodes in [111] for instance.

These models abstract microarchitecture details to estimate the energy consumption of CPUs when executing tasks.

Table 2.4 summarizes the power models available in different frameworks for task executions.

Network Interface Card power usage

The power consumption of a Network Interface Card (NIC) depends on its communication technology and usage.

Ns-3 has power models for wired communications in Ecofen [112]. Similarly, Wi-Fi and battery power models are also available and published in [113]. These models rely on ns-3's packet-level performance models that have been extensively validated. These models are considered accurate but limited by the scalability of packet-level simulation.

SimGrid has a model to simulate the energy consumption of wired network interfaces [86] based on its flow-based wired communication model. The results of this model have a relative error of 4% compared to ns-3. However, using a flow-level model reduces the runtime of scenarios executed in SimGrid by 120 times compared to simulations with ns-3. To the extent of our knowledge, there was no energy model for Wi-Fi using flow-level simulation before this thesis. A flow-based energy Wi-Fi model would be more scalable compared to packet-level solutions. Consequently, the flow-level simulation of the energy consumed by large-scale networks is currently limited to wired links.

The power models of IFogSim [63] do not differentiate between wired and wireless communication technologies. While this model can give broad estimations of the energy consumed by communications, the accuracy of communication times predictions and the lack of network protocol consideration reduces the accuracy of these models. Indeed, the energy consumed by a Wi-Fi network card changes depending on the number of antennas of the device, and the state of each antenna, which is not the case for wired links [114].

To study the energy consumption of fog networks at scale, using a flow-level framework with energy models for both wired and Wi-Fi communications is promising. However, Wi-Fi energy models are not offered by current flow-level tools.

2.4 End-to-end simulation of ICT infrastructures

In this section, we outline information about missing models for the end-to-end scalable simulation of network and processing infrastructures. In the previous section, we have seen

there already exist scalable and valid models for the simulation of wired communications and their energy consumption. However, available Wi-Fi models are either too fine-grained to study fog infrastructures or rely on strong hypotheses. Similarly, while microservice models exist in the literature, their calibration is very complex for large applications. In Section 2.4.1, we review Wi-Fi simulation requirements and introduce the material utilized in this thesis to model Wi-Fi communications. Section 2.4.2 focuses on the literature to model the energy consumption of Wi-Fi NICs. Finally, Section 2.4.3 reviews the design and calibration of microservice application models.

2.4.1 Wi-Fi medium access mechanisms

Wi-Fi channel access

In this thesis, we consider Wi-Fi in infrastructure mode, excluding ad-hoc networks. Thus, a Wi-Fi network is a set of Stations (STA) connected to an Access Point (AP) through a communication channel. The IEEE 802.11 standards define different parameters such as the frequencies that can be used by Wi-Fi devices, the bandwidth of communication channels, available coding schemes, and the number of spatial streams. These network parameters, known as the Modulation and Coding Scheme (MCS) configuration, have an impact on the communication throughput when devices use the network. Fine-grained simulation of Wi-Fi requires in-depth computations for each of these parameters to obtain accurate physical-layer transmission times. However, this accuracy comes at the cost of reduced scalability, limiting such models to the simulation of small-sized infrastructures as explained in Section 2.3.1. We decide to use a more coarse-grained model to simulate Wi-Fi communications and reach better scalability in terms of simulation runtime and platform size. In this context, an MCS configuration corresponds to the theoretical maximum throughput of the channel, defined at calibration time.

At the Medium Access Control (MAC) layer, IEEE 802.11 standards define the rules followed by Wi-Fi nodes (STAs and APs) to share a common communication channel. The Distributed Coordination Function (DCF) shares the access to the wireless channel between the STAs that try to send data concurrently. This function determines the time and the amount of data each station can send when active on the channel. It relies on different techniques to avoid collisions between several STAs that would otherwise start communicating concurrently. Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is the MAC layer protocol used to decide which node has access to the

channel. CSMA/CA states that every STA must sense an idle channel for a duration called Distributed Inter Frame Space (DIFS) before transmission. After sensing the channel as idle, an exponential backoff requires STAs to wait for an additional duration before sending their data. This duration is randomly chosen and multiplied by two after each channel access failure. Finally, the receiver emits an Acknowledgement (ACK) within the Short Inter Frame Space (SIFS) upon successful transmission. Figure 2.5 illustrates channel access between two STAs that a Wi-Fi performance model needs to reproduce.

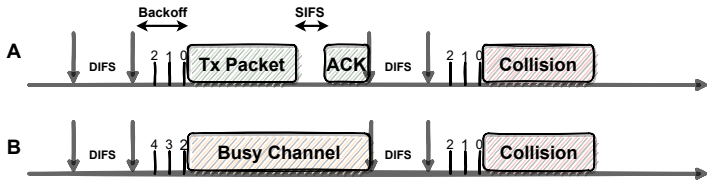


Figure 2.5 – MAC layer channel sharing example with 2 concurrent STAs trying to access the channel. Collisions happen when 2 STAs A and B start sending data simultaneously.

Flow-level Wi-Fi simulation model

A flow-level model was proposed for SimGrid by Loic Guegan in [53] to simulate the access to Wi-Fi channels with the DCF function. The calibration of this model consists in defining the application bandwidth for all STAs attached to an AP. Experiments using this flow-level model require less runtime and memory than the same experiments within ns-3 [53] but rely on strict hypotheses. This model does not consider collisions or mobility and was shown valid with a single MCS configuration. The authors also mention that validity is limited in scenarios with interferences, but *"the accuracy of our model is correlated with the number of stations that communicate through the wireless channel. Thus, grounding the interference model on the number of stations would be a good starting point toward accurate performance predictions"*.

Extending this model with an interference mechanism could enlarge the range of valid scenarios. The extended model could then be used to study more realistic fog scenarios, including dense Wi-Fi networks with nodes causing interferences.

2.4.2 Wi-Fi power consumption modeling

The duration of communications highly impacts the dynamic energy consumption of Wi-Fi NICs. Thus, Wi-Fi power models rely on performance models that predict

communication durations to evaluate the devices' energy consumption.

An example of the power usage of a Wi-Fi interface is given in Figure 2.6. We observe that interfaces switch between different states depending on the tasks executed by the NICs (sending, receiving data, channel sensing). These different power states originate from the activation of different parts of the NIC depending on its activity. As reviewed in the literature [113, 114], the states of a Wi-Fi network interface are:

— *IDLE*: The NIC does not perform any operation, consuming P_{IDLE};

— *Transmission (Tx)*: The NIC actively sends data, requiring to power the antennas of the device, consuming P_{Tx}. In contrast to CPU power models that experience varying power consumption depending on the number and frequency of active cores, Wi-Fi NICs transition between fixed states, without intermediate power levels;

— *Reception (Rx)*: The NIC actively receives data, which requires listening to the wireless channel, consuming P_{Rx};

— *Sleep*: The NIC deactivates some circuitry to reduce energy consumption. In this case, the power usage of the NIC reduces to P_{sleep}.

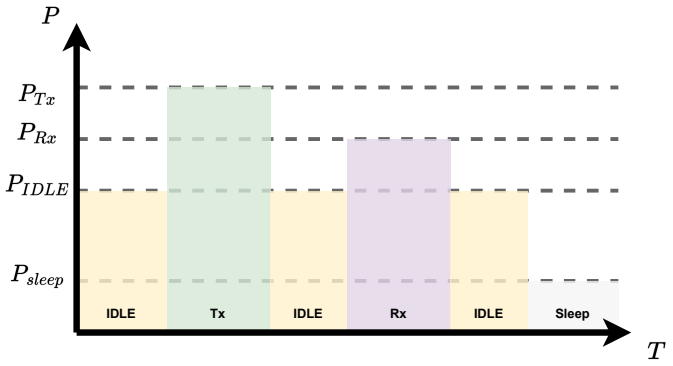


Figure 2.6 – A Wi-Fi NIC switches between different states depending on the task performed by the device

For the calibration of Wi-Fi power models, state-of-the-art contributions measure the power drawn by different NICs in each state.

The authors of [114] provide power values for an Intel Wi-Fi NIC supporting IEEE 802.11 a/b/g/n. They measure the power consumed by the NIC by placing a resistor on its power supply. Their experiment provides the NIC's power consumption with varying channel width, different spatial stream configurations (up to 3 antennas), and different transmit powers. Despite the accuracy of their measurements, this study is not representative of the Wi-Fi NICs used in other devices (smartphones, routers).

Another paper published in 2015 [115] adopted a similar approach to measure the power usage of a wide range of Wi-Fi interfaces. In this case, they measure the NICs' power usage for routers, a smartphone, and a Raspberry PI. For each device, they measure and model the power usage when idle, sending, and receiving data with different MCS configurations. Compared to the previous paper, the authors also measure the power consumed by the operating system to decode and process network packets. They observe that the implementation of the operating system's network stack can impact energy consumption and name this impact the cross-factor.

To instantiate ns-3's Wi-Fi power model, the authors of [113] reuse the measurements obtained in [114]. The model's authors also mention the energy consumed during *Clear Channel Assessment* (i.e. sensing if the channel is busy and if a station can send data). Based on previous works, their resulting model does not differentiate between the power usage in *IDLE* and *CCA* states. In this thesis, we consider the same default calibration values as ns-3's power model [113].

Modeling the duration and the energy consumed by Wi-Fi communications at scale permits the study of ICT infrastructures mixing between wired and wireless communications like fog infrastructures. In the following, we focus on understanding how resources are utilized by applications to simulate scenarios involving microservices. Specifically, we emphasize the instantiation of the models to ease the study of large applications.

2.4.3 Microservice applications simulation

Since the use of resources in ICT infrastructures depends on the applications, it is critical to simulate realistic applications to obtain credible results. We focus on microservice applications given their popularity in modern ICT infrastructures such as fogs [5].

Application model

Microservice simulation tools need to consider the application at three different levels: **a)** the execution of requests within each service, **b)** the interdependencies between services of the application, **c)** the placement and autoscaling algorithms to dynamically adapt the application to the load.

A request executed in a service uses resources (CPU, I/O, RAM). First, when receiving and forwarding requests, there is an overhead due to the reception and processing of network data [61]. This duration can be estimated using network models. Then, a

microservice instance can process some requests in parallel depending on the capacity of its host. The execution of the request necessitates CPU and I/O durations estimations. These durations can be estimated using processing and disk models, as available in SimGrid [116]. Then, the accuracy of intra-service request executions depends on the calibration of the model.

To model the interactions between services, we need to consider the requests' execution path. As explained in Section 2.2.1, microservice applications form DAGs of dependencies between tasks executions [60, 63, 107]. In this DAG, nodes correspond to executions of requests within a service, and edges to network messages between services.

Finally, microservice applications are monitored by tools such as Kubernetes [65] to automatically deploy new instances of services in case of high loads. In this context, considering autoscaling policies helps to produce realistic results. Different autoscaling policies are available, for example using system metrics to deploy new replicas when exceeding usage thresholds [117, 118]. Placement policies are also important because of the impact of services' location on application performance.

Model instantiation

Since microservice applications can be composed of numerous services with several request execution paths, the instantiation of microservice models is complex. When studying state-of-the-art microservice simulation frameworks [107], the calibration of the models is manual, requiring a lot of time and leading to potential errors.

In the domain of High-Performance Computing, researchers encountered comparable issues when simulating MPI applications. Some works base the calibration of their models on the data obtained within application traces to solve these issues. For instance, the authors of [103] use real-world MPI application traces to calibrate SimGrid's MPI model. A similar approach may ease the instantiation of microservice models.

Most microservice applications are instrumented to assist application maintainers in identifying performance bottlenecks. Distributed tracing standards like OpenTracing [119] and its successor OpenTelemetry [120] permit the generation of traces for each service's performance at runtime. These standards define the traces' format to simplify their processing. Jaeger [121], initially developed by Uber in 2015, and Zipkin [122] from Twitter and now managed by the OpenZipkin organization, are two popular open-source projects for trace processing and visualization. They reconstitute the path of requests among different services and measure single or average requests' execution durations in each service.

Thus, an alternative use for these tools is to benchmark an application and obtain service execution times and the DAG of function calls automatically for models' calibration. Since these monitoring solutions are widespread among microservice developers, traces are available without extensive programming effort.

To the extent of our knowledge, no state-of-the-art contributions for the simulation of microservices make use of application traces to instantiate the models.

2.5 Conclusion

Our review of existing literature on the study of end-to-end ICT infrastructures highlights important efforts made in this field to improve the efficiency of devices and perform accurate measurements.

In Section 2.1, we observed the important uncertainties between different studies regarding the evolutions of ICT infrastructures and their energy consumption. Different prediction methodologies lead to very different estimations of the future impact of ICT and show the need for reproducible and scalable methodologies to estimate this impact, especially during the use phase.

In Section 2.2, we focused on the evolution of ICT infrastructures and applications to improve application performance. We observed the complexity of modern ICT infrastructures with cloud and fog computing. We also reviewed the uncertain tradeoff between the gain of performance of novel fog approaches and the environmental impact of decentralized nano DCs. To better understand this tradeoff, we observed the necessity to evaluate the energy consumed by end-to-end infrastructures.

In Section 2.3, we focused on simulation to study the energy consumption of fog infrastructures. Compared to other methodologies, simulation models can be tuned to reach a given tradeoff between scalability and accuracy depending on the research questions under study. We observed that flow-level simulation models can provide sufficient accuracy in our context while preserving good scalability.

In Section 2.4, we reviewed the background towards the scalable simulation of end-to-end infrastructures. We focused on the simulation of Wi-Fi communications and their energy consumption, along with the execution of microservice applications.

These observations call for a methodology enabling accurate evaluations of the energy consumption of end-to-end fog infrastructures running applications, as we shall introduce in the next chapters. The position we defend in this work is that flow-level models can be

used to simulate end-to-end fog infrastructures at a large scale, while preserving accurate results.

Chapter 3

Validation of a Flow-Level Wi-Fi Model for Large Scale Network Simulation

Network communications at the edge of the network often rely on wireless technologies such as Wi-Fi. However, state-of-the-art simulation models for Wi-Fi either lack validation or scalability. In this chapter, we propose to study the tradeoff between the accuracy and the scalability of Wi-Fi models based on flow-level simulation. After describing a flow-based Wi-Fi model proposed in a previous work by Loic Guegan [53], we extend this model and evaluate its accuracy and scalability against ns-3.

This chapter is organized as follows. Section 3.1 introduces the flow-based framework used in the rest of this chapter and the design proposed in [53]. In Section 3.2, we extend the base model to provide improved estimations of network communication durations. In Section 3.3, we describe the implementation of this model in SimGrid and its calibration methodology. Section 3.4 compares the accuracy and scalability of the SimGrid implementation of the model against ns-3. Section 3.5 concludes this work.

3.1 Context and hypotheses

3.1.1 Flow model framework

Obtaining a scalable flow-based Wi-Fi model requires switching to a more abstract representation of communications than packet-level models. As described in Section 2.2.2, packet-level simulators consider every packet transmitted between nodes, each packet requiring computations to estimate the probability of its successful reception and its transmission time. Using flow-level simulation, the communication between two nodes of a network is represented as a single flow. In this chapter, we use the term *Wi-Fi link* to

describe the Wi-Fi cell, i.e., the aggregation of an AP, the communication channel, and the STAs attached to the AP. The bandwidth allocated to a flow depends on the properties of the link, and the other flows in the network. Flows have a constant throughput between network events such as the arrival or deletion of a flow. Upon network events, the bandwidth associated to each flow is updated to match the new state of the network. This approach is less compute-intensive than packet-level simulation since the number of packets is typically much higher than the number of flow-related events. However, the gains in scalability come at the price of less fine-grained results. Compared to a packet-level simulator that can provide the state of the network at any time, a flow approach only provides the average bandwidth between events.

In this context, designing a flow-level Wi-Fi model requires defining the properties of each Wi-Fi flow, and the rules to update the bandwidth of flows upon network events. This is done through the use of a solver that takes the state of the network as an input, modeled as an inequation system, and outputs the bandwidth allocated to each active flow. The inequation system is defined using variables and constraints similar to Equation 3.1.

$$\begin{cases} \sum\limits_{\substack{i \text{ using} \\ \text{link } 1}} a_{1,i} \times \rho_i \leq C_1 \\ \dots \\ \sum\limits_{\substack{i \text{ using} \\ \text{link } r}} a_{r,i} \times \rho_i \leq C_r \\ \dots \\ \sum\limits_{\substack{i \text{ using} \\ \text{link } m}} a_{m,i} \times \rho_i \leq C_m \end{cases} \tag{3.1}$$

Each link of the network is represented by one inequation r associated with a constraint C_r. The inequation is used to compute the throughput of all the flows passing through this link. The variable ρ_i represents the bandwidth allocated to the flow i. The constraint C_r ensures that the sum of the bandwidths of all the flows passing through link r will not exceed the link's capacity. C_r thus corresponds to the maximum throughput of the link. Coefficients $a_{r,i}$ can be used for each variable to model behaviors such as acknowledgments as their usage happens to reduce the link's available capacity.

To attribute the bandwidth ρ_i of each active flow i, we solve the following prob-

lem [123]:

$$\begin{aligned}&\text{MAXIMIZE } \min_i w_i \times \rho_i\\&\text{UNDER CONSTRAINTS}\\&\begin{cases}\forall r, \sum\limits_{\substack{i \text{ using}\\ \text{ressrouce } r}} a_{r,i} \times \rho_i \leq C_r\end{cases}\end{aligned}$$

Where the constraints are the system given in Equation 3.1, and the weight of flow i, w_i can be used to prioritize some flows and model phenomena such as RTT-Fairness (when the bandwidth of a node decreases inversely proportionally to its round trip time). Solving this problem can be done iteratively [123], starting from the most constrained link until all throughputs are determined. The load of the link associated to inequation r can be noted $\epsilon_r = C_r/(\sum_{i \text{ using } r} \frac{a_{r,i}}{w_i})$. Consequently, the most constrained inequation of the system is the one that minimizes ϵ_r. Solving that inequation is done by computing $\rho_i = \epsilon_r/w_i$ for each flow i traversing the link corresponding to this inequation. Once an inequation is fixed, constraints C_r through which passes flow i are updated such that $C'_r = C_r - a_{r,i} \times \rho_i$. Then, the same operation is iterated to fix all flows.

This kind of solver is used in the SimGrid simulation framework [95], and in other flow-level simulators such as Narses [89]. More details about the implementation of this solver in SimGrid called the Linear Max-Min solver (LMM), and how to apply it to model wired communications can be found in [123].

In the case of Wi-Fi, Loic Guegan [53] proposed to adapt this inequation system, initially proposed for the simulation of wired communications, to the constraints of Wi-Fi. In the following, we summarize and quote the equations for the Wi-Fi model of [53].

3.1.2 Flow-based Wi-Fi bandwidth sharing

The model introduced in this section is based on the description of the MAC layer IEEE 802.11 channel sharing given in Section 2.4.1. As explained in Section 2.4.1, a Wi-Fi model needs to reproduce the behavior of the DCF, i.e. managing the access of STAs to the wireless channel. In this section, the channel is supposed to be in ideal conditions with no interferences, no signal loss, and no hidden node. The two factors that play a major role in the way the channel is shared among stations are *a)* r_i, the datarate associated with station STA_i which depends on AP and STA configurations, *b)* d_i the amount of data to be sent by STA_i. For conciseness, in this chapter, we use the term *channel* to describe both the nodes (STA, AP) and the channel of a Wi-Fi cell. We now describe the flow-based model presented in [53].

The model's design is illustrated using the example of Figure 3.1 within a single Wi-Fi cell composed of one AP and three STAs. In this example, the three STAs want to send data to the AP concurrently over some time T. Each STA needs to send some packets: STA_1 and STA_2 have 8 packets to send, while STA_3 has 1 packet. Figure 3.1b shows one possible share of the channel that could be obtained using a packet-level model.

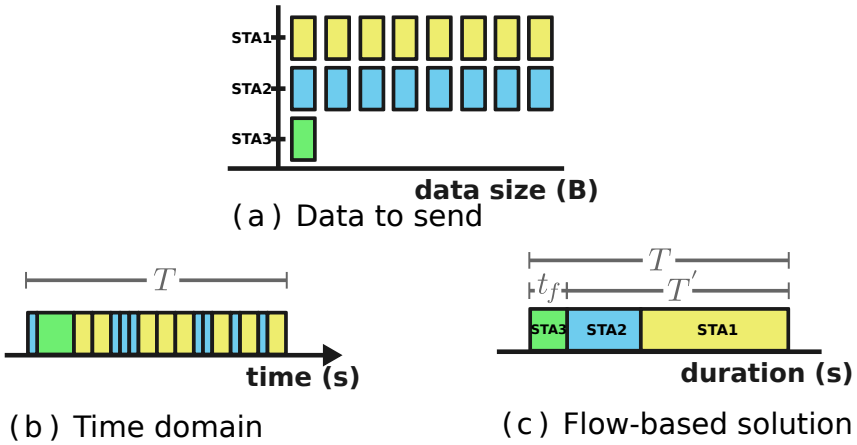


Figure 3.1 – Example of Wi-Fi communication over a period T with 3 STAs where $r_2 = 2r_1 = 4r_3$, taken from [53].

The goal is to compute the time each STA spends sending data over T, as shown in Figure 3.1c. The use of the Wi-Fi link for period T is expressed as $C = \frac{\sum_{i=1}^{n} d_i}{T}$. Because of the exponential backoff described in Section 2.4.1, all nodes have the same probability of accessing the channel and transmitting frames. This leads to a fair share of the channel regarding the amount of data sent by each station. Theoretically, all stations will transmit the same amount of data, $\forall i, j, d_i = d_j = d$, where d_k is the amount of data sent by STA_k during T. As a consequence, the throughput of all nodes during T is equal to the same value, $\forall i, j, \rho_i = \rho_j = \rho$. Therefore, the usage of link C in the max-min inequation system is a fair share among all active flows:

$$C = \sum_{i=0}^{n} \rho_i = \frac{n \times d}{T} = \frac{n \times d}{\sum_{i=0}^{n} \frac{d}{r_i}} = \frac{1}{\frac{1}{n} \sum_{i=0}^{n} \frac{1}{r_i}} \tag{3.2}$$

However, some of the flows might be fixed by other constraints of the system. Suppose the throughput of the flow starting from STA_1 (noted ρ_1) is fixed such that $\rho_1 \leq \frac{C}{n}$ because the receiver of the flow is slower than the maximum transmission rate of STA_1. Because STA_1 does not take all its fair share of airtime, the other flows will share with the remaining link capacity. C' corresponds to the remaining capacity of the channel once

the fixed flow has been removed. Let d_f be the amount of data sent by the fixed flow such that $d_f \neq d$. Since $t_f = \frac{d_f}{r_f}$ an $\rho_f = \frac{d_f}{T}$, we have:

$$T = T' + t_f = T' + \frac{d_f}{r_f} = T' + T \times \frac{\rho_f}{r_f}, \text{ hence } T = \frac{T'}{1 - \frac{\rho_f}{r_f}} \tag{3.3}$$

Let d' be the amount of data to be sent by the remaining stations. Since we have

$$C = \sum_{i=0}^{n} \rho_i = \rho_f + C' = \frac{d_f + (n-1)d'}{T},$$

the capacity of the remaining flows can be computed as:

$$C' = \sum_{i=0, i\neq f}^{n} \rho_i = \frac{(n-1)d'}{T} = \frac{(n-1)d'}{T'} \times \left[1 - \frac{\rho_f}{r_f}\right] = \frac{1}{\frac{1}{n-1}\sum_{i=0, i\neq f}^{n} \frac{1}{r_i}} \times \left[1 - \frac{\rho_f}{r_f}\right] \tag{3.4}$$

Let I be the set of all the flows not fixed in the system, and F the set of fixed flows, we can generalize Equation 3.4 to an arbitrary number of fixed flows:

$$C' = \frac{1}{\frac{1}{|I|}\sum_{i\in I} \frac{1}{r_i}} \times \left[1 - \sum_{f\in F} \frac{\rho_f}{r_f}\right] \tag{3.5}$$

Equation 3.5 defines how to update the inequation system once one or several flows have external constraints limiting their sending capacity. Solving this system is less computationally intensive than a packet-level approach where each network packet has to go through a set of complex models. Few parameters are necessary compared to packet-level models: **a)** the number of flows, **b)** the sizes of flows, **c)** the datarate of each station.

3.2 Model extension

The model presented in Section 3.1.2 can be used to simulate IEEE 802.11n communication durations in ideal conditions. In this section, we extend this model to permit considering non-ideal channel conditions.

3.2.1 Capacity reduction of concurrent flows

The equations from Section 3.1.2 enable sharing the bandwidth of a Wi-Fi channel between a set of flows in ideal conditions. We refine the base model by taking some

potential causes of performance degradation on the Wi-Fi channel into account.

One example is the case of collisions between nodes. The more flows are executed concurrently on a single channel, the higher is the probability of having two stations sending data at the same time, leading to collisions and retransmissions. This reduces the channel time for successful communications. The exponential backoff and mechanisms such as Request To Send/Clear To Send (RTS/CTS) try to minimize the probability of such events, but collisions still happen in real networks.

In packet-level simulators, successful transmissions and receptions are computed using protocol-specific propagation and probabilistic loss models [85, 124]. We propose a more minimalistic approach to capture this issue, adapted to a flow model.

We expand the bandwidth-sharing model to adjust the constraint associated with the maximum capacity of a Wi-Fi link depending on the number of active flows using the link. To compute a performance loss we can thus define a function f depending on the number of concurrent flows i using resource r, such that:

$$C_r = f(\sum_{i \text{ uses } r} 1) \tag{3.6}$$

This function f can be used to capture collision effects in heavily loaded Wi-Fi networks. In our simulations of single Wi-Fi cells using ns-3 in Section 3.3.2, we observe an overall stationary throughput in IEEE 802.11n cells up to a certain number of concurrent flows, after which the throughput linearly decreases. Once the threshold and the coefficients of the linear curve are obtained experimentally (either through real or simulated experiments using more fine-grained models), we can define f such that:

$$f(x) = \begin{cases} bw_0 & x < thresh \\ a * x + bw_0 & x >= thresh \end{cases} \tag{3.7}$$

Where bw_0 is the maximum throughput, a is the coefficient of the curve, and x is the number of concurrent flows. While this calibration process requires some experiments, once the function has been extracted, it can be reused for several simulations. In addition, other definitions of f could be used in the future to model other effects even if we only use f as defined in Equation 3.7 in this thesis.

3.2.2 Propagation model with SNR Levels

By default, the STAs on a Wi-Fi link are assumed equidistant to the AP. We can extend the base model to allow STAs to have different positions when starting an experiment. STA mobility during the experiments remains out of the scope of this work. The position of a STA can impact its datarate. Packet-level simulators compute a SNR for each data frame based on a loss model and use it to deduce a Packet Error Rate (PER) [85].

Again, our approach is more minimalistic than packet-level models. We do not have access to the same amount of information, preventing us from estimating SNR and PER values accurately. Instead, we allow the experimenter to define the list of datarates that we name *SNR levels* for each station in a cell. At runtime, it is possible to define which datarate to use for each STA separately depending on its position. As described in the equations of Section 3.1.2, the datarate limits the maximum bandwidth of a station if its SNR is lower than the theoretical maximum STA throughput.

3.3 Model implementation and calibration

3.3.1 Implementation

Loic Guegan's model [53] was implemented in 2021 in the SimGrid simulation framework. Our extension has been added to that base model [95]. As described in Section 2.2.2, SimGrid is a flow-level simulation framework for distributed systems proposing various models for Ethernet communication, CPU, and disk usage, among others. We chose SimGrid for the efficiency of the integrated inequation solver, and the possibility of combining the Wi-Fi model with other SimGrid models. SimGrid being open source software, this model is integrated within SimGrid's source tree and will be available in all future releases [125]. Currently, Wi-Fi cells must be at the end or start of a flow's path. Wi-Fi links in the middle of the communication path could be considered as future work.

Adding this model to SimGrid requires the creation of communication links specific to Wi-Fi. In the solver, the bandwidth of flows using Wi-Fi links are updated using the equations of Section 3.2. Wired and wireless links can thus be used jointly in a simulation.

3.3.2 Calibration

Since the bandwidth of Wi-Fi channels highly depends on the experimental scenario under study, the flow model must be carefully calibrated to obtain accurate performance estimations. The configuration of STAs, the density of nodes in the network, and external channel degradation have all an impact on the capacity of a Wi-Fi cell. Using our model, we rely on the calibration of Wi-Fi parameters before executing simulations to consider the impact of those effects. Calibration values can be obtained through real measurements or microbenchmarks on more fine-grained simulators such as ns-3. The rest of this section describes the different calibration steps.

Single station throughput:

The frames sent over an IEEE 802.11 channel do not only include applications' data frames. Control frames, beacons, and RTS/CTS messages can also be sent regularly on the Wi-Fi channel. This time spent managing the cell impacts the throughput available for application data compared to the theoretical maximum available throughput. The flow model does not account for the time spent sending such frames and thus needs to be calibrated to fit that maximum throughput. This value can be obtained by running microbenchmarks to obtain the maximum data throughput between a STA and an AP. The result can then be used to calibrate SimGrid.

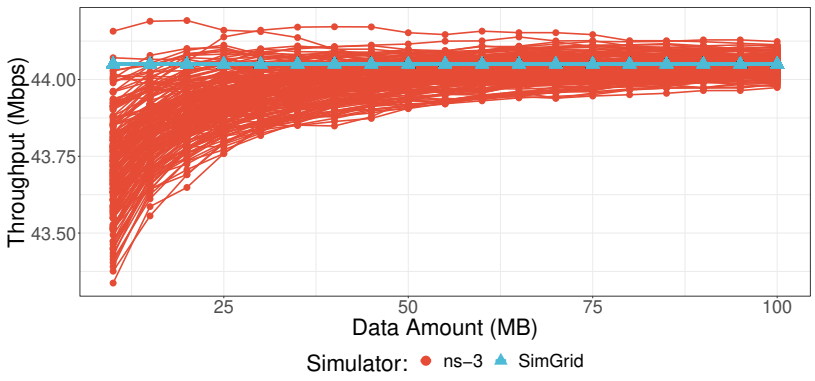


Figure 3.2 – Single station throughput given flow size with 100 seeds for ns-3 in red.

We chose to run an ns-3 simulation with one Wi-Fi cell made of an AP and one STA configured with an MCS of 3. A single flow is created from the STA towards the AP in this cell. The simulation is repeated several times with 100 different random seeds. Figure 3.2 shows the results of this experiment where red points are the throughput of

ns-3 for different data sizes. We observe that the maximum flow throughput converges around 44.1Mbps when increasing the size of flows.

By calibrating SimGrid's model with this datarate $r = 44.1Mbps$, we obtain the blue line of Figure 3.2. SimGrid estimations do not depend on random number generation, thus all experiments will give the same outputs in contrast to ns-3. This value remains within 1% of all observed values with ns-3 during this calibration experiment.

Overall bandwidth reduction upon high contention:

This step focuses on the throughput available for stations depending on the number of concurrent flows in the network.

Various amounts of concurrent flows are simulated in a cell, where a different number of STAs try to send data simultaneously. In this case, flows towards the AP (ascending) and from the AP to the STAs (descending) are mixed. Pairs of STAs are created and communicate from one node to the other, passing by the AP. We executed the same scenario with exclusively ascending or descending flows, leading to similar results. The results obtained with ns-3 are shown in Figure 3.3, where each point is a different simulation execution with as many STAs on the channel as the number of concurrent flows.

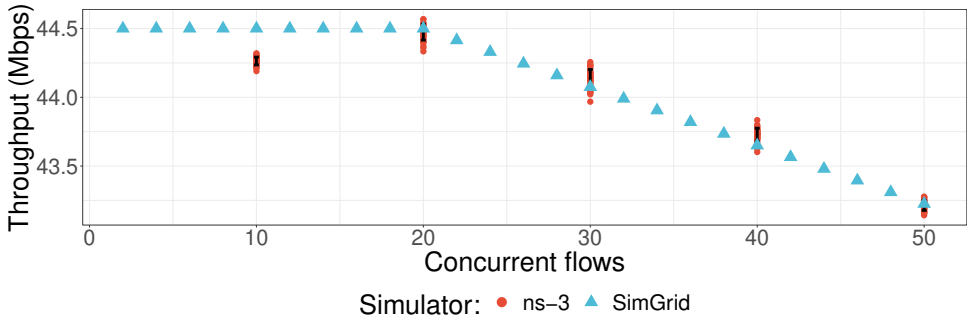


Figure 3.3 – Throughput degradation under concurrent flows.

This experiment highlights a new phenomenon that was not observable in the previous calibration step. The throughput available to communicate decreases with the number of concurrent flows. This happens regardless of the MCS configuration used for the experiments as shown in Section 3.4. This decrease can be explained by several factors: more concurrent flows require more management frames and RTS/CTS requests if activated (in this case, RTS/CTS is used for packets above 100 bytes), and it also increases the probability of having collisions when STAs start sending data at the same time. Even

without collisions, already communicating nodes slow down the others, which need to increase their backoffs. This issue is accounted for with the link capacity degradation mechanism described in Section 3.2.1.

Using the observations of ns-3, the function f is designed as a linear piecewise function, using a constant maximum throughput below 20 concurrent flows before linearly decreasing the maximum throughput once this threshold is exceeded. A linear regression has been done to estimate the throughput degradation related to the amount of concurrency. Once these values are obtained, we inject them into SimGrid to obtain the triangles visible in Figure 3.3. We observe that, from a few ns-3 simulations, we manage to calibrate our model to run under an arbitrary number of concurrent flows. Ns-3 results vary with different random seeds, while the flow model always gives the same results. The same approach can be used for other MCS configurations.

Distance to the access point:

The last benchmarks we used to calibrate our model observe the maximum distance between an AP and the STAs that communicate with it. It can be done by running an experiment within a Wi-Fi cell with one AP and a single STA. Several simulations are done, where the throughput obtained between the AP and the STA is observed depending on the distance between the two nodes. Once we know the maximum communication distance, we can set the datarate of STAs according to the values obtained experimentally, i.e. any STA located above the maximal distance will be assigned a datarate of 0bps. Rate adaptation algorithms [126] are out of the scope of this work, along with mobility during the experiment. In our validation, STAs have a fixed position, defined at the beginning of the experiment.

3.4 Validation

This section evaluates our flow-based model. As a baseline to compare the metrics provided by our model, we compare the outputs of SimGrid to the outputs of ns-3. This choice to compare a flow model to another simulator is motivated by the following reasons:

— real experiments using wireless devices are very hard to reproduce at scale;
— ns-3 is actively maintained by a large community;
— ns-3 models are publicly available and validated in peer-reviewed articles [85];
— ns-3 is extensively used for Wi-Fi simulations;

— the packet-level nature of ns-3 provides more fine-grained results than flow-level simulation. Closely related results between SimGrid and ns-3 would show the accuracy of our approach.

We compare different metrics such as the overall Wi-Fi cell throughput, single flow throughput, or performance metrics for several use cases. In the following, we first assess the validity of our model on simulations of limited scale, before validating the scalability of our approach on a realistic infrastructure.

Experimental Setup. Experiments use SimGrid v3.29 with the implementation of our model, whereas we use an unmodified ns-3 v3.33 with IEEE 802.11n models. Random number generation within ns-3 has been shown to lead to very different results depending on the random seed used to initialize the simulations. This is due to the probabilistic nature of some phenomena such as drawing random backups or signal propagation, among others. Experiments are executed on the Grid'5000 [71] testbed to sample executions using different seeds. Regarding STAs positions, we uniformly put the stations within a circle of radius of 15 meters around the AP when not explicitly written. This mitigates issues such as the hidden node problem [127] and simplifies the analysis of the results.

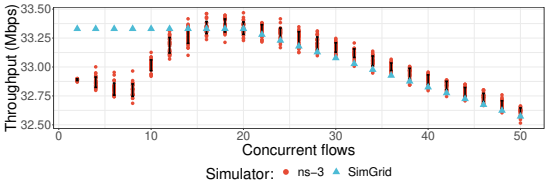
Source code and reproducibility. Experiment artifacts, including code, scripts, visualizations used for this chapter, and additional results are available at `https://github.com/klementc/wifi-reproducibility`

3.4.1 Small-scale validation through microbenchmarks

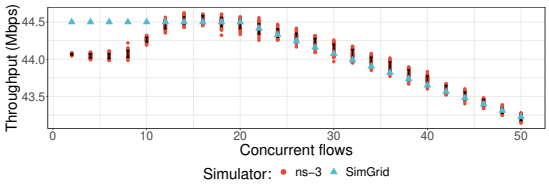
In the following, we study the capacity of our model to give accurate performance estimations on limited-size platforms.

Using the throughput reduction mechanism:

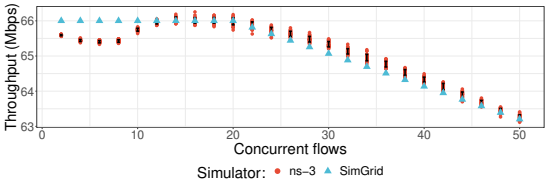
We observe the impact of the number of concurrent flows using different Wi-Fi MCS. Figure 3.4 shows the overall throughput of a cell using four MCS configurations (MCS 2 to 5). To obtain these results, we first execute ns-3 microbenchmarks to calibrate our model with the maximum available throughput and the coefficient of the linear decrease of function f. This calibration is made, as in Section 3.3.2, with a limited number of ns-3 experiments to avoid overfitting. The simulated communication flows start simultaneously, once all STAs are properly attached to the AP. We simulate the Wi-Fi cell with different numbers of concurrent flows and compare the outputs of the flow model to the



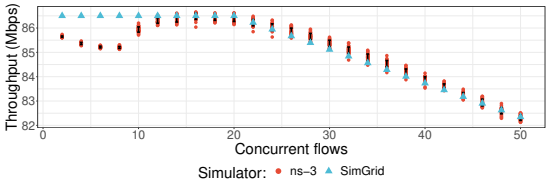
(a) MCS 2 with $x_0 = 4239574$, $a = -3210$



(b) MCS 3 with $x_0 = 5678270$, $a = -5424$



(c) MCS 4 with $x_0 = 8517520$, $a = -12023$



(d) MCS 5 with $x_0 = 1.120e{+}07$, $a = -1.792e{+}04$

Figure 3.4 – Comparison of overall throughputs between ns-3 (in red) and SimGrid (in blue) simulations, depending on the number of concurrent flows in a single Wi-Fi cell using 35 ns-3 seeds. Black bars show the standard deviation around the mean for ns-3 simulations.

outputs of ns-3.

Each plot in Figure 3.4 shows the throughput estimations obtained after calibration. Different MCS configurations lead to different maximum theoretical throughput values, visible in our plots where Figure 3.4a (MCS 2) is estimated to reach approximately 33.3 Mbps against 86.5 Mbps in Figure 3.4d (MCS 5). The maximum relative error of the overall throughput is 1.5% under MCS 2 with a small number of concurrent flows.

This experiment shows the ability of our model to simulate the share of a Wi-Fi channel between an arbitrary number of concurrent flows under various MCS configurations. After calibrating the model, it is possible to obtain closely related outputs compared to ns-3.

SNR levels microbenchmark:

This experiment simulates communications when STAs have different locations. Two STAs send data to the AP, where STA_1 is fixed, and the distance between STA_2 and the AP is increased between consecutive simulations. This scenario does not consider mobility since the STAs do not move during a simulation. The two stations are always within reach to avoid the hidden node issue. The flow-based model is calibrated following

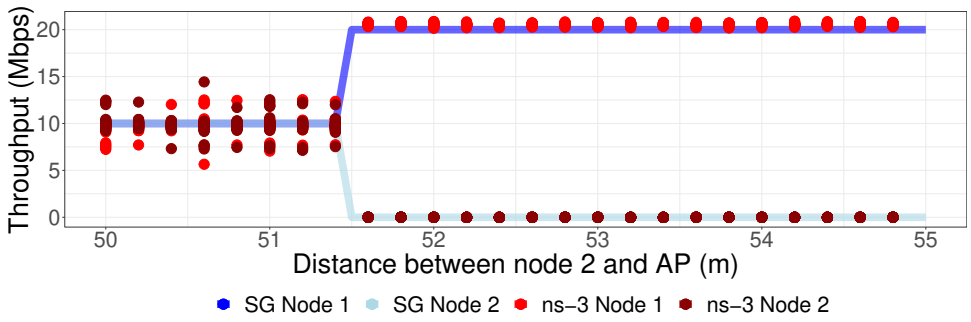


Figure 3.5 – Impact of STA distance on the throughput share between stations.

the methodology of Section 3.3.2. The MCS of the stations is set to 3, and channel bonding and RTS/CTS are deactivated.

Figure 3.5 shows the throughput of nodes 1 and 2, where the distance between the AP and STA_2 ranges from 50 m to 65 m. Ns-3 results are sampled with 30 different seeds since its signal propagation and packet-loss models depend on probabilistic computations. Below 51.5 meters, the channel is fairly shared between the two stations in both ns-3 and the flow model. Above this limit, node 2 is too far from the AP to attach and communicate properly, and cannot send any more data. The bandwidth that was previously dedicated to node 2 is given to node 1, which can use the full capacity of the link.

These results show that even without complex PER computations, simply defining the datarates of STAs as described in Section 3.3.2 allows simulating STAs with different locations. Both the flow model and ns-3 lead to the same bandwidth modification when node 2 is too far from the AP.

3.4.2 Use case: large scale infrastructure

This section evaluates the performance and accuracy of Wi-Fi communications in a more realistic use case. The infrastructure simulated in this section is a metropolitan public Wi-Fi network. Such networks are extensively used in wide areas such as university campuses [128], commercial centers [129], or to cover entire cities as in the case of Google Wi-Fi [130]. These networks are typically used to allow users to connect to an AP and access services over the Internet. They can be composed of hundred APs, used by clients to access the Internet. Figure 3.6 illustrates the infrastructure of the platform we simulate in this section. The network is decomposed into two layers: **a)** a set of Wi-Fi APs to which clients connect using their Wi-Fi devices, and **b)** a wired gateway between the APs

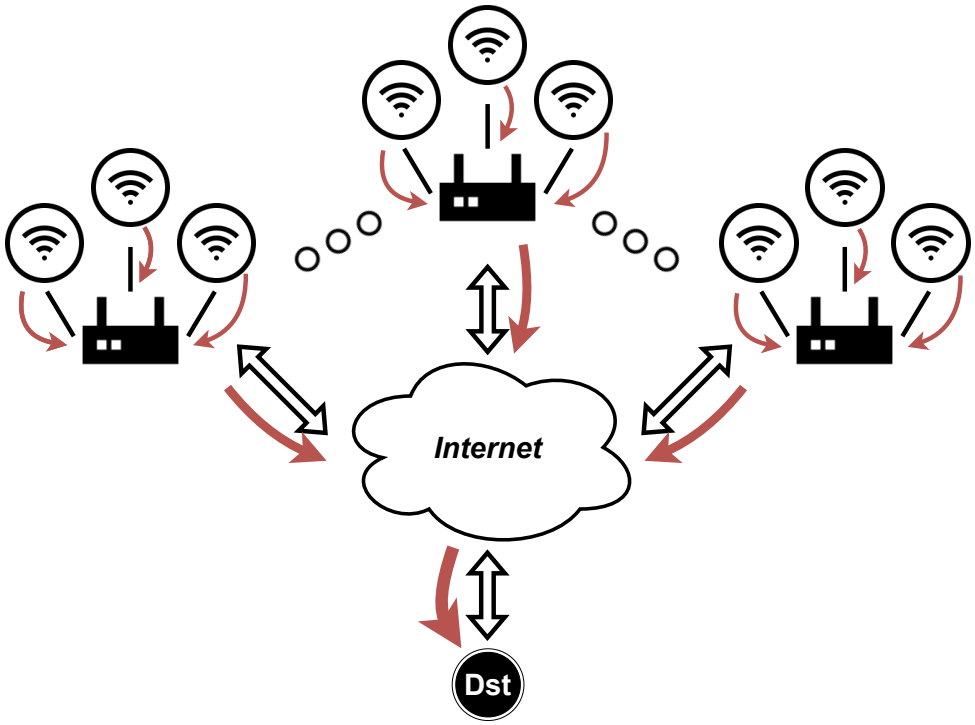


Figure 3.6 – Simulated architecture for the large-scale experiment

and the core network to access the Internet. Network flows are created between client STAs and the core network towards an Internet destination *Dst*.

We evaluate the accuracy of the flow-based model against ns-3, modifying the number of STAs per cell, the number of APs, the size of network flows, the number of messages sent by each STA, and the random seed of ns-3.

Overall throughput:

In this scenario, each Wi-Fi cell is composed of 20 nodes, sending a message whose size is randomly selected between 10 MB and 30 MB to the gateway. All flows start at the same simulated timestamp of 10 seconds. This ensures that all nodes have enough time to connect to their respective AP before the flows start. It also constitutes a more challenging situation for the model, as non-concurrent flows are easier to predict.

Figure 3.7 shows the duration between the beginning of communications and the time of the last received byte in both SimGrid and ns-3 with 3 Wi-Fi cells. We use 35 different seeds for ns-3, where we remove the runs where the flows are not finished by the end of the experiment (145 seconds), considered as outliers. This experiment shows that SimGrid's

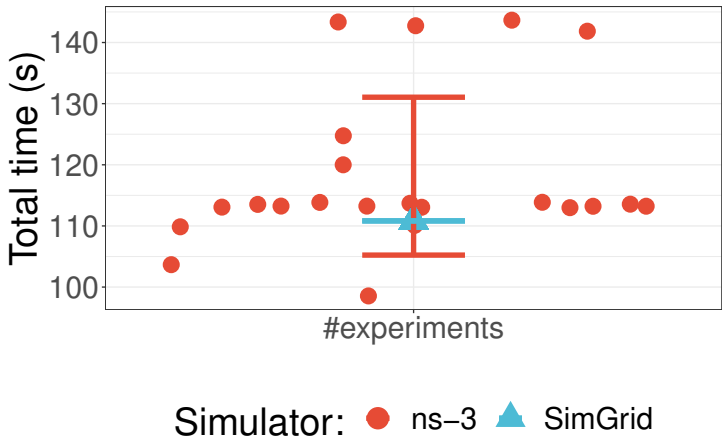


Figure 3.7 – Time to communicate the same amount of data between SimGrid in blue and ns-3 in red on the same network topology. Times above 145s with ns-3 are considered outliers and have been removed. Ns-3 points have different x-axis positions to observe the values for different seeds more easily.

estimation of the time to send data from concurrent stations in the network is within the error range of ns-3 outputs that differ by the random seed.

When considering the overall network usage, the outputs of SimGrid are coherent with the ones of ns-3 despite the more abstract model. Repeating this experiment with different flow sizes (between 10 and 30 MB per flow) and number of Wi-Fi cells (1 to 10 cells) gives similar outputs that are omitted for conciseness [1].

Single flows throughput:

This experiment compares the duration of each communication flow between the SimGrid implementation of the model and ns-3. Simulations are similar to the previous section, with ascending flows from the STAs towards the AP, with varying flow sizes. The flow-level model being less fine-grained than ns-3's, it does not compute probabilistic phenomena as ns-3 does. In Figure 3.8, flows do not overlap, a situation that would happen in a quiet Wi-Fi cell with low traffic. In the case of Figure 3.9, all the flows start at the same time, T=10s.

Figures 3.8 and 3.9 present chronological plots indicating the flow termination timestamps in both SimGrid and ns-3. The top points indicate the end times for each flow in SimGrid, while the bottom points represent the timestamps of the same flows in ns-3.

1. `https://github.com/klementc/wifi-reproducibility/blob/master/analysis/Figures_paper_reproducibility.ipynb` notebook with experiments' results, last accessed December 2023

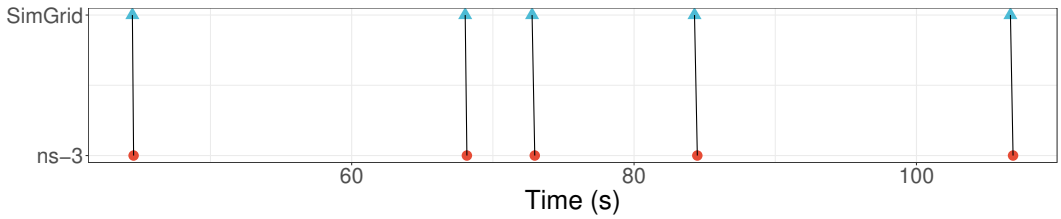
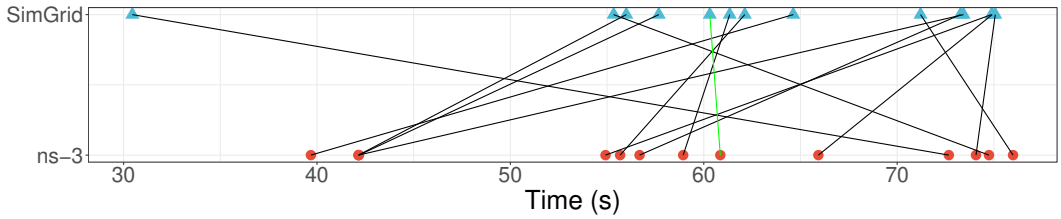
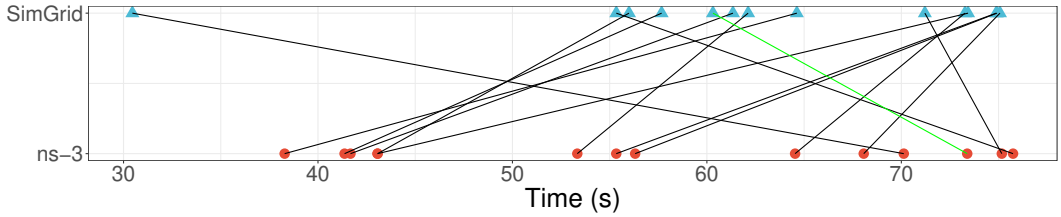


Figure 3.8 – Flows end timestamps without concurrency (no overlap between flows start and end) in a cell of 5 STAs



(a) seed=19



(b) seed=20

Figure 3.9 – Flows end timestamps with concurrency (flows start at T=10s) in a cell of 14 STAs

Each vertical line connects SimGrid and ns-3 for a given flow. When the outputs of SimGrid and ns-3 perfectly match for a given flow (as we intend to), the corresponding line is perfectly vertical. In the case of non-overlapping flows, the logarithmic error, as defined in [100], is on average equal to 0.051, meaning a 5% relative error between SimGrid and ns-3 flow durations. In the case of concurrent flows, however, the end timestamps do not match between ns-3 and SimGrid, leading to very important relative errors. In this case, the throughput sharing of ns-3 is very dependent on the random seed used for the simulation, as highlighted for seeds 19 and 20. For instance, the flow colored in green finishes just after 60 seconds in SimGrid. While ns-3 outputs a similar timestamp in Figure 3.9a, there are more than 10 seconds of delay in the case of Figure 3.9b (around 73 seconds). The error between the flow model and ns-3 is thus very different depending on the random seed of ns-3. Additionally, we can also observe that the end timestamps

of the last flows are similar in SimGrid and ns-3.

What is obtained using SimGrid is one of the possible shares of the channel between the concurrent flows (a theoretical fair share). Despite having very different results between ns-3 and SimGrid for single flow durations depending on the random seed, leading to an important logarithmic error, the flow-based model outputs a coherent time-share of frames in the network. As outlined previously, we estimate accurately the overall duration for all flows (shown in Figure 3.7) and give one possible share of the channel when observing flows separately.

Network throughput over time:

This experiment explores the evolution of the throughput in the same infrastructure under a variable load. In this case, 22 Wi-Fi cells and 126 active STAs are simulated. Each station sends a number of messages centered around 40 with a deviation of 3, of variable size 1.5 MB with a deviation of 1 MB over 1250 seconds of simulated time. Wi-Fi cells do not overlap to avoid inter-cell interference. Ns-3 experiments are replicated using 35 different seeds.

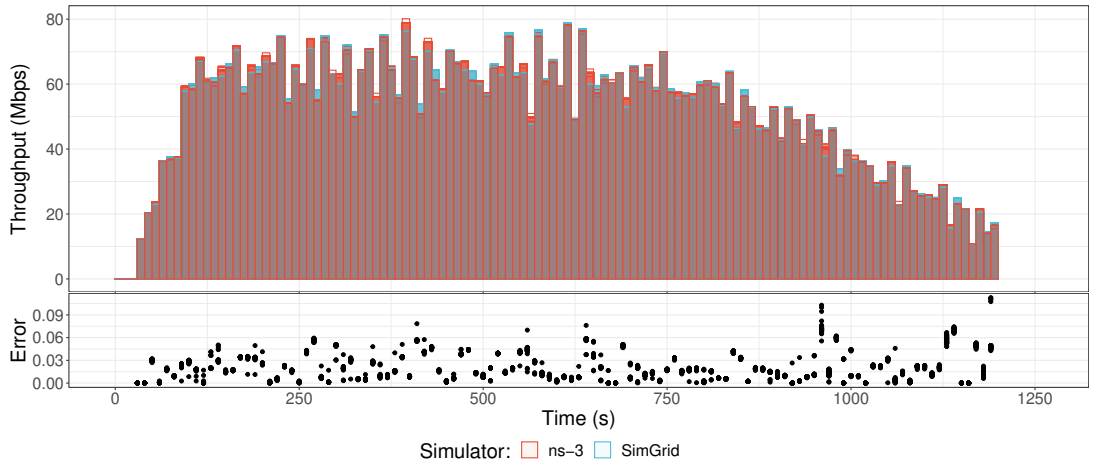


Figure 3.10 – On top, data received by the gateway node by 10s intervals with 22 cells and 126 stations. At the bottom, the relative error between ns-3 and SimGrid

Figure 3.10 shows the amount of data originating from the STAs and received by the gateway node by intervals of 10 seconds. We note th_{sg} (respectively th_{ns}) the throughput of SimGrid (respectively ns-3) during each interval. The amount of data going through the network changes over the simulated time because of the different packet sizes and time intervals between consecutive messages for each STA. The relative error regarding

the data throughput (computed as $\frac{|th_{sg}-th_{ns}|}{th_{ns}}$ for each interval) remains below 12.5% at most, and below 10% on average.

This experiment shows that our model can be used to study large-scale platforms under dynamic network loads with a reasonable relative error.

Performance comparison:

This experiment explores the performance improvement of our flow-based model compared to ns-3. To that extent, we simulate between 1 and 22 Wi-Fi cells. The number of STAs in each cell is centered around 7, with a deviation of 3, uniformly distributed around the AP. Flows are the same as above.

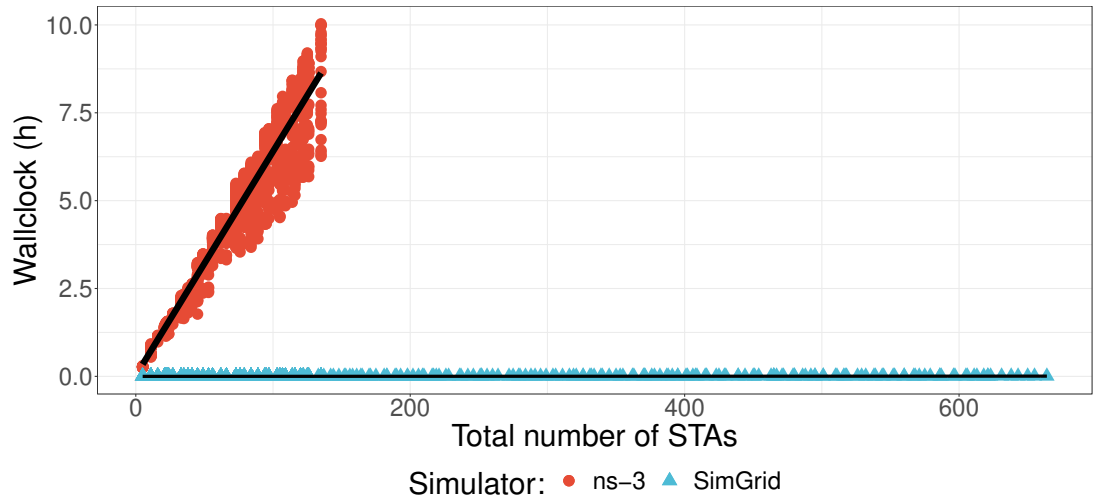


Figure 3.11 – Wallclock comparison of simulation durations between ns-3 and SimGrid. The black line is the linear regression of time values.

Figure 3.11 shows the time spent simulating our scenarios for infrastructures of different sizes. While the simulations using ns-3 require several hours for each seed, SimGrid-based simulations take at most a few seconds for the longest simulations. This difference is explained by the tradeoff between the model granularity of Wi-Fi communications depending on flow and packet-based models. While the packet-based approach can be of linear complexity with regard to the number of simulated packets, the complexity of the flow-based approach only depends on the number of flows. We executed other simulations in SimGrid, made of up to 650 STA across 100 cells. The longest simulations took approximately 25 seconds to complete. Similar experiments using ns-3 would require more than 44 hours according to a linear regression made with the data of Figure 3.11. For

large simulations, the peak memory usage was higher with SimGrid compared to ns-3 (approximately 300 MB against 150 MB for ns-3 to simulate 100 STAs) due to the large number of simultaneous flows to handle in SimGrid. But this memory usage takes place over a much shorter amount of time in SimGrid than ns-3.

Overall, our experiments show that using a flow-based model instead of the classical packet-level one leads to a runtime improvement of several orders of magnitude for simulating large Wi-Fi networks while providing estimations of similar accuracy.

3.5 Conclusion

In this chapter, we focused on the simulation of Wi-Fi communication times. Our contribution extends a Wi-Fi performance model implemented in SimGrid. We extended the model with a bandwidth reduction mechanism that can limit the maximum bandwidth of the link depending on the number of active flows. This can be used to model some interferences, and dense networks more accurately. Then, we validated our contribution against ns-3 simulations. Properly calibrated, this model can estimate accurately the duration of Wi-Fi communication under different MCS configurations and flow concurrency degrees. We also evaluated the scalability of our model in comparison to the Wi-Fi models of ns-3, showing significant speedup with a limited loss of accuracy.

Chapter 4

A Wi-Fi Energy Model for Scalable Simulation

In Chapter 3, we extended and validated a flow-level Wi-Fi performance model. This model provides closely related results compared to ns-3 while significantly improving simulation durations. Based on the communication time predictions of the performance model, we propose an energy model for Wi-Fi adapted to flow-level simulation.

In Section 4.1, we introduce the hypotheses and the context in which our Wi-Fi energy model can be used. In Section 4.2 we detail the model, its implementation in SimGrid, and its calibration. In Section 4.3 we compare the accuracy of the results of our model with the results of ns-3's energy models in microbenchmarks. We study a large-scale scenario and compare the salability of flow-level and packet-level energy simulation in Section 4.4. We discuss the limitations of our approach in Section 4.5. Section 4.6 concludes this chapter.

4.1 Hypotheses and conditions

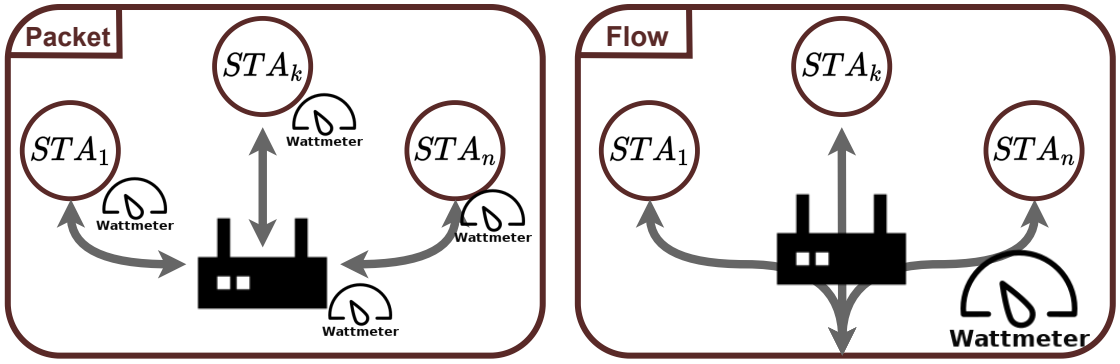
Estimating the energy consumption of Wi-Fi nodes depends on the evolution of the NICs' states during simulations. We have seen in Section 2.4.2 that communications modify the state of the NIC from *idle* to Rx or Tx, which is the main reason for power variation. In this chapter, we assume that the simulation of the durations of Wi-Fi communications is accurate.

Wi-Fi standards such as IEEE 802.11ax [131, 132] define different Power Saving Mechanisms, such as Target Wake Time, PS-POLL, or Operation Mode Indication. These mechanisms are out of the scope of this work and are not considered in this chapter.

The use of several antennas to add redundancy or to transmit and receive data in parallel, known as Multiple-Input Multiple-Output (MIMO), can be considered using the energy model presented in this chapter but has not been validated.

The model presented in this chapter is based on the Wi-Fi performance model presented in Chapter 3. This model does not consider control and management frames nor beacons which are used to manage Wi-Fi networks. These frames are responsible for advertising the network, performing probe requests, associating a STA to the AP, or synchronizing devices. Beacons are a type of Wi-Fi frame sent periodically from the AP toward STAs. The performance model does not inject them in the network load, thus they are not modifying the state of the simulated NICs contrarily to the packet-level simulation model of ns-3 [113]. We need to consider them in our model.

A flow-based energy model induces constraints in comparison to packet-based models. Our flow-based approach estimates the energy consumption of Wi-Fi NICs not at the level of Wi-Fi devices, but at the granularity of the communication medium (called *link* hereafter). As in the previous chapter, we use the term *Wi-Fi link* to describe the aggregation of an AP, the communication channel, and the STAs attached to the AP. Figure 4.1 shows the difference between simulating power consumption using a packet model and our approach. In Figure 4.1a, the power is estimated using network packets' events happening within each NIC, while Figure 4.1b illustrates our approach at the granularity of links, with network events at the beginning and termination of network flows.



(a) Packet-based simulation, power estimated at the granularity of network interfaces

(b) Flow-based measurement, power estimated at the granularity of Wi-Fi "links", aggregating the NICs of the AP and the STAs

Figure 4.1 – Flow vs. Packet power measurement. Packet-based measurements are done with NICs' granularity whereas flow-based measurements are done at the WLAN's level

Finally, the model introduced in this section is homogeneous. In contrast to heterogeneous models where the power states of each device can be different, a homogeneous model considers that all the devices attached to a Wi-Fi link consume the same amount of energy to perform the same actions. While implementing a heterogeneous model is

possible, homogeneous models provide good accuracy [87] and increase scalability.

4.2 Modeling the energy consumption of Wi-Fi NICs

4.2.1 Model overview

Different components need to be taken into account to compute the overall energy consumption of a Wi-Fi link over an experiment of duration T. The energy consumed by a Wi-Fi link in Joules over that period is expressed as the integral of the power of the link during that period.

$$E(T) = \int_0^T P(t)dt \tag{4.1}$$

The power of a Wi-Fi link corresponds to the sum of its static power consumption P_{stat} and its dynamic power consumption P_{dyn}. In our case, P_{stat} and P_{dyn} correspond to the energy consumption of all devices (AP and STAs) attached to a Wi-Fi link, expressed in Watts.

In a discrete event flow-level simulator, an event is triggered for every start or termination of a network flow. An event involving a Wi-Fi link may change the power usage of some devices, requiring an update of the overall power consumed on the link.

The energy consumed in between each event is computed using the following equation:

$$\begin{aligned} E_{tot}(link) = t \times (& \sum_{s \in STA(link)} (P_{stat}(s) + P_{dyn}(s)) \\ & + P_{stat}(AP) + P_{dyn}(AP)) \end{aligned} \tag{4.2}$$

where t is the duration since the previous event, and $STA(link)$ is the set of all stations on the Wi-Fi link $link$, and AP is the access point. To simulate a network with more than one Wi-Fi WLAN, the energy consumed by the whole network is equal to the sum of the energy consumed by each of the Wi-Fi links.

4.2.2 P_{stat}: Static power consumption

P_{stat} is the power used by network interfaces without active communications. For Wi-Fi links, computing this value requires knowledge of the power consumed by NICs in the *idle* state. Static power consumption on the link is then equal to the sum of the static

power consumption of each device, whether AP or STAs, as follows:

$$P_{stat}(link) = P_{idle}(AP) + \sum_{s \in STA(link)} P_{idle}(s) \tag{4.3}$$

In the validation of this work, only homogeneous network devices (in terms of electric consumption) are considered, so the idle power consumption on a Wi-Fi link is the same for all NICs. Thus, n STAs attached to the link (and an AP), with a homogeneous idle consumption on the link P_{idle}, we get :

$$P_{stat}(link) = (n+1) \times P_{idle} \tag{4.4}$$

This equation gives the power consumed by a Wi-Fi link without any active communication. P_{stat} can be multiplied by the simulated time to obtain the energy consumed by the link. Switching from a homogeneous to a heterogenous model only requires using Equation 4.3 instead of Equation 4.4, and considering the energy consumed by each NIC individually. However, it requires additional computation at simulation time.

4.2.3 P_{dyn}: Dynamic power consumption

The dynamic power consumption P_{dyn} is the power used by the Wi-Fi devices to receive or send data actively on the communication channel. Since Wi-Fi networks use broadcasting, when one device sends data on the channel, all STAs in the range of that device receive the data at the NIC level, even if only the destination acknowledges the frame and processes it at the application level.

The computation of P_{dyn} for a Wi-Fi link is based on the power consumption of NICs in the Tx and Rx states. When a device starts a communication, it switches to the transmission state, consuming power P_{Tx}, and the other nodes in the range of the STA on the same channel (including the AP) listen for this information, consuming P_{Rx}. This way, in a network composed of n STAs and one AP, we obtain :

$$P_{dyn}(link) = U(link) \times (P_{Tx}(link) + n \times P_{Rx}(link)) \tag{4.5}$$

where U is the usage function of the link: If there is at least one active communication on the Wi-Fi link, U is equal to 1. If there is no active communication the value of U is equal to 0, consequently the dynamic power consumption is also equal to 0. Again, since

the proposed model is homogeneous, the values of P_{Tx} and P_{Rx} are the same for all the interfaces within a Wi-Fi LAN.

4.2.4 Power consumption of control frames

Since the performance model used to build this power model only considers the applicative data transferred on a Wi-Fi link, we need to consider control frames separately. This factor is important in networks with few communications where control frames can become the main source of dynamic energy consumption.

We extend our model to account for their energy consumption using an additional factor. We only consider beacons because once STAs are connected to their AP, they represent the majority of control frames in the network. Since beacon frames are sent periodically, usually at approximately 10Hz for IEEE 802.11 (every 102.4 ms), the energy cost of sending and receiving beacons is constant over time and depends on the number of machines on the link. To obtain the overall power consumption, we add the cost of beacons to the static and dynamic power consumption of nodes. We note C the factor used to model the energy consumption of beacons. C corresponds to a ratio of the time spent each second by the nodes of a WLAN exchanging beacons. By introducing C into Equations 4.4, 4.5, we obtain:

$$P_{tot}(link) = P_{stat}(link) + P_{dyn}(link) + C \times (P_{Tx} + n \times P_{Rx}) \tag{4.6}$$

The value of C can be computed using the lowest rate of the frequency band employed in the Wi-Fi network – since it is the rate employed to transmit beacons (especially for backward compatibility reasons) – and using the beacon size (depending on the Wi-Fi version). It can also be estimated experimentally from the amount of time spent in the communication of beacons. Section 4.3 shows how to do it using the ns-3 simulator for a single transmission rate, but this could be done for other network configurations.

Finally, applying Equation 4.6 to Equation 4.2 provides an estimation of the overall energy consumed by a Wi-Fi network.

4.2.5 Implementation

We implemented this model in SimGrid. We rely on the Wi-Fi performance model presented in Chapter 3 to accurately compute the duration of Wi-Fi flows once correctly

calibrated.

SimGrid has a plugin system allowing easy integration of our energy model by hooking onto the network events of Wi-Fi links. Our model relies on the estimation of the power consumption of a Wi-Fi link between each arrival or termination of a Wi-Fi flow. The plugin hooks onto events linked to flow creation and destruction and updates the power consumption of the link using the previously described equations.

Since the model is based on the Wi-Fi performance model of Chapter 3, this implementation inherits from the limitations of the underlying Wi-Fi performance model.

4.3 Validation

4.3.1 Experimental setup and methodology

To validate this energy model, we compare the output of our implementation in SimGrid to the predictions of ns-3 [113]. We chose to compare the two simulators for the same reasons as given in Section 3.4 of the previous chapter.

The ns-3 and SimGrid models are calibrated with the default energy values of ns-3 given in Table 4.1. These values originated from [114].

Table 4.1 – Power values used to calibrate the model. Obtained from ns-3's *WifiRadioEnergyModel* [113, 114]

NIC State	Power (W)
IDLE	0.82
Rx	0.94
Tx	1.14
Sleep	0.10

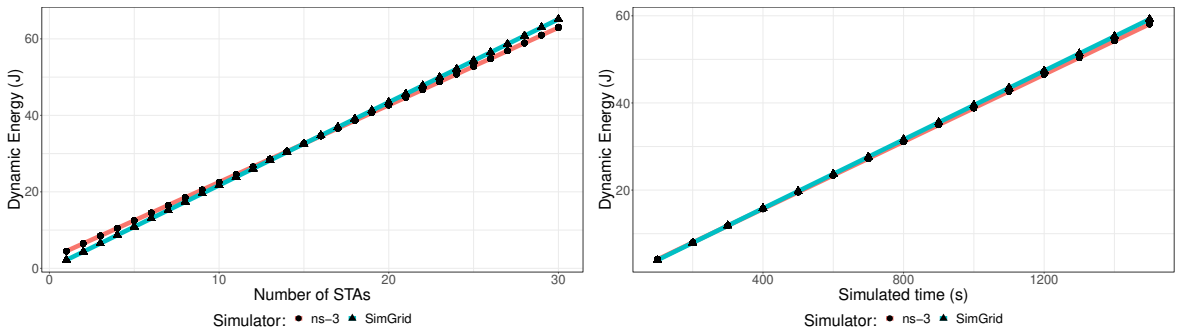
The metrics compared between the two simulators are: **a)** the overall energy consumption of Wi-Fi interfaces, **b)** the dynamic energy consumption only since it is the most challenging value to estimate, **c)** the memory footprint and the execution time between ns-3 and SimGrid for different experiments to evaluate the scalability of our approach.

All experiments have been executed using the Grid'5000 [71] experimental testbed, allowing the simulation of a large number of scenarios while using several random seeds in ns-3, which impact the underlying error models and communication times.

Source code and reproducibility: The code of the model, the code of the experiments, the scripts used to generate all figures, the logs of the results used in this chapter, information on how to reproduce our results, and additional results are available at the following address: `https://github.com/klementc/wifi_energy_experiments`.

4.3.2 Evaluating the cost of beacons

The first set of simulations calibrates the cost of control frames in the network, as described in Section 4.2.4. We compute experimentally the value of the factor C to obtain accurate predictions of the energy consumed in the network without active flows by using ns-3 simulations. We run several experiments where we measure the dynamic energy consumption of ns-3 devices in a single Wi-Fi cell. Measuring only the dynamic energy consumption means setting the *idle* power consumption to 0W in the configuration of ns-3 while leaving the other values identical to Table 4.1. Simulations are performed with between 1 and 30 STAs connected to the AP, varying the simulated time of the experiment between 100s and 1500s, with 100 different ns-3 random seeds. No flows are created between the STAs and the AP, meaning that only control frames are consuming energy on the Wi-Fi channel. Using the results, we estimate the ratio of time spent communicating beacons on the channel per second.



(a) Dynamic energy consumption with different numbers of STAs and a fixed simulation time (1100 seconds)

(b) Dynamic energy consumption with different simulation times and 20 STAs attached to the AP

Figure 4.2 – Simulating the energy cost of beacon frames in a single WLAN

The results obtained are shown in Figure 4.2. Figure 4.2a shows the dynamic energy consumption of the network depending on the number of devices for 1100 seconds of simulated time, while Figure 4.2b shows the dynamic energy consumption for different simulated times in a cell with 20 STAs. In both cases, we first execute experiments in

ns-3. We can observe that the energy consumption values obtained vary linearly with the number of STAs in the LAN and the simulated time, matching the description of periodic beacons given in Section 4.2.4. Using linear regression, we compute a value $C = 0.0021$ for our model, leading to the outputs observed for SimGrid. This value of C gives a mean relative error (computed as $\frac{|val_{sg} - val_{ns}|}{val_{ns}}$) of 7.0% over all the simulated experiments, showing that we obtain closely related results with both models.

Consequently, simulations use the value $C = 0.0021$ in our experiments. To match other transmission configurations, C can be modified in SimGrid at calibration time.

These results also allow us to observe that control frames have a relatively small impact on the overall energy consumption of Wi-Fi nodes. While this factor is negligible for heavily used Wi-Fi networks where large flows consume much energy, it may become important on networks with long inactive channel periods.

4.3.3 Single WLAN energy prediction

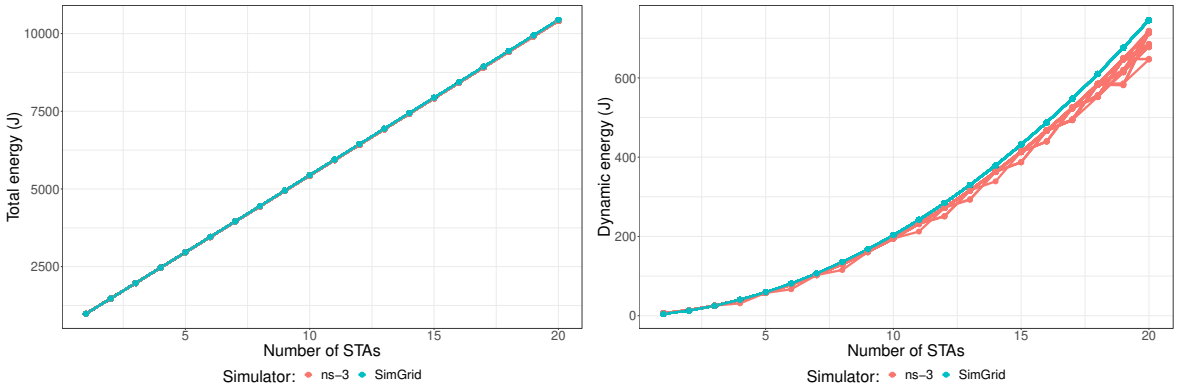
Once the model is calibrated for control frames, it is possible to run a set of microbenchmarks to evaluate the accuracy of our model in the presence of network flows in a single Wi-Fi cell. We perform several experiments using two different types of flows: **a)** flows between STAs and the AP, **b)** flows between pairs of STAs (going through the AP).

The explored parameters are: the size of network flows (between 1 MB and 30 MB) and the number of STAs in the simulated Wi-Fi network (between 1 and 20 for the AP/STA flows, and between 2 and 20 for the flows between pairs of STAs). To avoid comparing our results to an outlier output of ns-3, each scenario is executed with 100 different ns-3 random seeds.

Figure 4.3 shows results for the simulation of flows between the STAs towards the AP with 600s of simulated time, where each flow has a size of 10MB. Figure 4.3a shows the overall energy consumption ($E_{tot} = E_{dyn} + E_{stat}$), while Figure 4.3b shows only the dynamic energy consumption for the same experiments, summing both the cost of control frames and the cost of application flows in the network.

We observe that the overall energy consumption in Figure 4.3a is a quasi-linear function depending on the number of connected STAs. This observation is consistent with the fact that static energy consumption accounts for most of the overall energy consumption in the network. In Figure 4.3b, we observe that more flows lead to more dynamic energy consumption, along with additional beacon receptions in both simulators.

The results of both figures show that the values computed by our model match with the



(a) Total energy consumption comparison between SimGrid and ns-3 for different numbers of STAs

(b) Dynamic energy consumption comparison between SimGrid and ns-3

Figure 4.3 – Energy consumption comparison between SimGrid and ns-3 using different numbers of STAs in the downstream flows scenario

predictions of ns-3. The average relative error over all our experiments when computing the total energy consumption is equal to 0.15%, while the same error is equal to 5.22% regarding the dynamic energy consumption only. The error is bigger when measuring the dynamic energy consumption only since it is highly dependent on the simulated duration of network flows, which is complex to estimate (and slightly varies between different ns-3 seeds).

We also execute the flow pairs scenario, where flows start from one STA, and go towards another STA in the same WLAN, passing through the AP. Figure 4.4 gives an overview of the average dynamic power usage of the Wi-Fi nodes between ns-3 and SimGrid for this scenario. Results are shown only for a flow size of 20 MB, while the results with other flow sizes are available in the online notebook [1]. We observe that the average power in the cell depends on the size of the Wi-Fi cell, i.e. the number of STAs attached to the AP. The flow-based model provides very similar results compared to the outputs of ns-3, showing the accuracy of our model in this scenario.

Overall, the microbenchmarks at the scale of a single Wi-Fi cell show that the flow-level model matches the energy predictions of ns-3. This is the case when comparing the overall energy consumption but also the dynamic energy consumption individually.

1. `https://github.com/klementc/wifi_energy_experiments/blob/master/analysis/Figures_microbenchmarks.ipynb`, last accessed December 2023

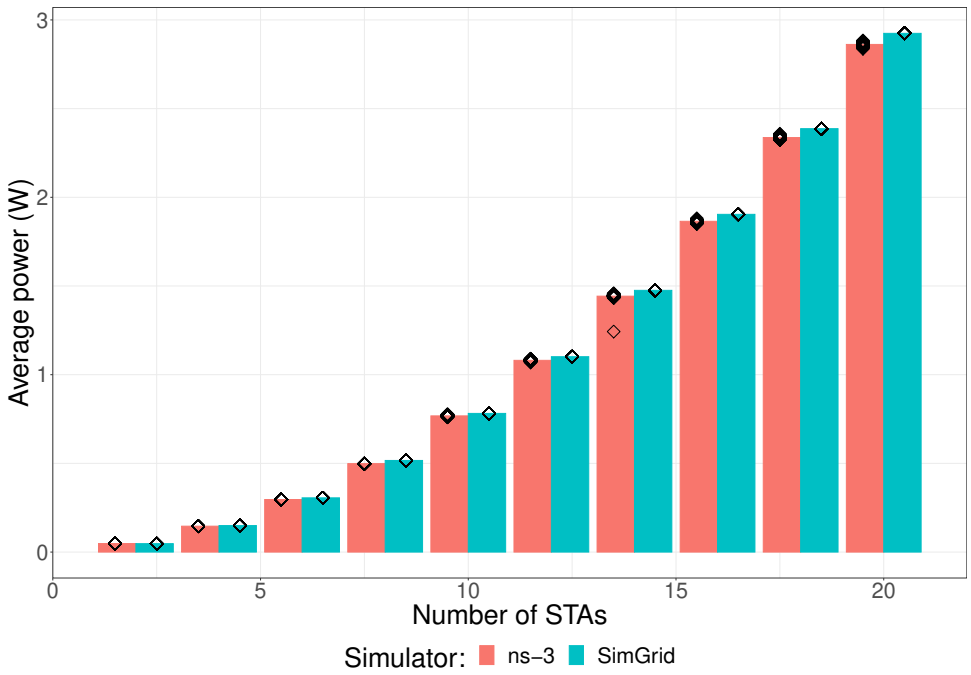


Figure 4.4 – Average dynamic power usage of STAs (*IDLE* power set to 0 W) in the flow pairs scenario for different ns-3 random seeds represented as points

4.3.4 Simulation of mixed Wi-Fi/Wired communications

Network simulators such as ns-3 and SimGrid allow the simulation of heterogenous networks (i.e. networks mixing different communication technologies in the same experiment). This feature is important to study realistic infrastructures, for example, mixing Ethernet links and Wi-Fi cells. This set of experiments simulates a small-scale heterogeneous infrastructure composed of two Wi-Fi networks joined by an Ethernet link.

Each of the STAs in the first Wi-Fi network is the source of a network flow towards another STA in the second Wi-Fi network. Each network flow passes through the wired link, leading to additional simulation constraints. We use the standard *PointToPoint* model of ns-3 to simulate this link and the default model for wired communications from SimGrid where the throughput of the link is set to 10Gbps. We run several simulations varying the size of flows between 3 MB and 20 MB, the number of STAs in each cell between 1 and 10, a simulated time between 700 and 1100 seconds, and 40 different ns-3 seeds.

Figure 4.5 shows the results obtained with a simulation time of 900 seconds, between 3 and 6 STAs in each cell, and flow sizes of 6, 9, and 12 MB respectively. In each

case, we provide a violin plot of the dynamic energy consumption estimated in ns-3 using different seeds against SimGrid's energy predictions (SimGrid predictions do not depend on random computations, leading to a single point). The violin and the black dots allow observing the dispersion of ns-3 results. Depending on the random seed, the Wi-Fi channel is shared differently in ns-3, leading to different communication times, hence different dynamic energy consumption values. Even if most values are relatively close, there are outliers such as in the 12 STAs setup where we can observe a difference of $\sim 17\%$ between different executions of ns-3.

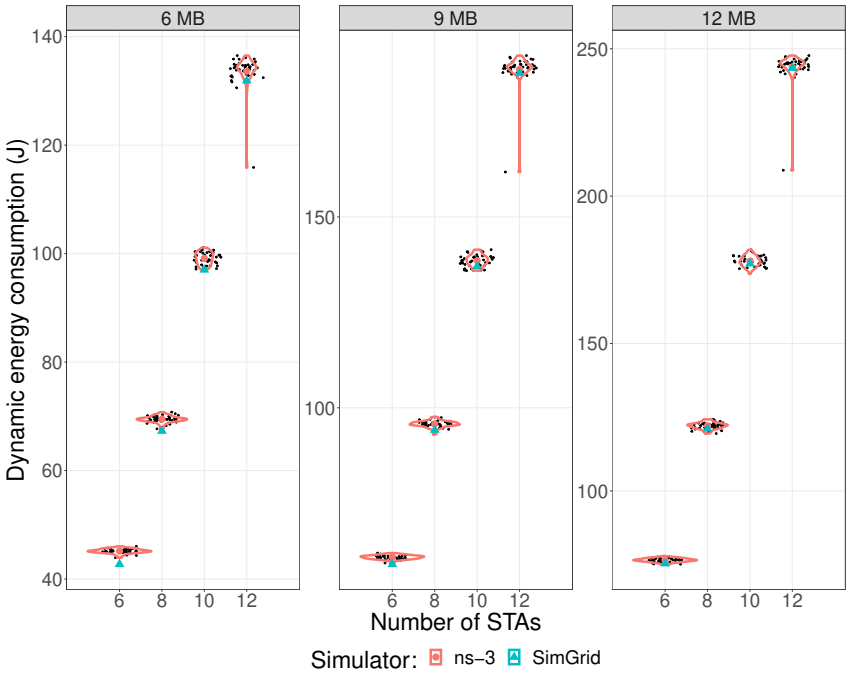


Figure 4.5 – Dynamic energy consumption in the heterogeneous infrastructure. Each facet corresponds to a different flow size, where colored points designate the mean dynamic energy consumption for every number of STAs. Black dots show the dispersion of ns-3 predictions depending on the random seed, and the red violin the probability density of the data at different values.

Overall, our results remain close to the average of ns-3, with a mean relative error of 4.4% for the dynamic energy consumption. In this setup, the error is more important for small flow sizes. As in the previous experiments, looking at the overall energy consumption of the network further reduces this error given the important proportion of energy consumed in the *idle* state, which is easier to predict.

This set of experiments validates our model at a small scale in the case of a heteroge-

neous network.

4.4 Large scale simulations

The experiments in this section explore the ability of a flow-based model to efficiently simulate the energy consumption of large-scale networks. We reuse the same simulated network infrastructure based on a public Wi-Fi network as in Chapter 3. A description of this platform is given in Section 3.4.2 and Figure 3.6. Since communication times are supposed to be valid in this infrastructure, we can focus on energy metrics in this section.

We perform several simulations in both ns-3 and SimGrid using different parameters: the number of available APs for STAs to connect to (between 1 and 50), the number of STAs connected to each AP (17 on average with a standard deviation of 2), the size of the network flows, the timestamp at which network flows are created, and 30 ns-3 seeds. In every experiment, each STA in the network sends 25 messages on average towards the destination node *Dst* connected to the APs via wired network links. Each simulation runs for 1,100 seconds of simulated time.

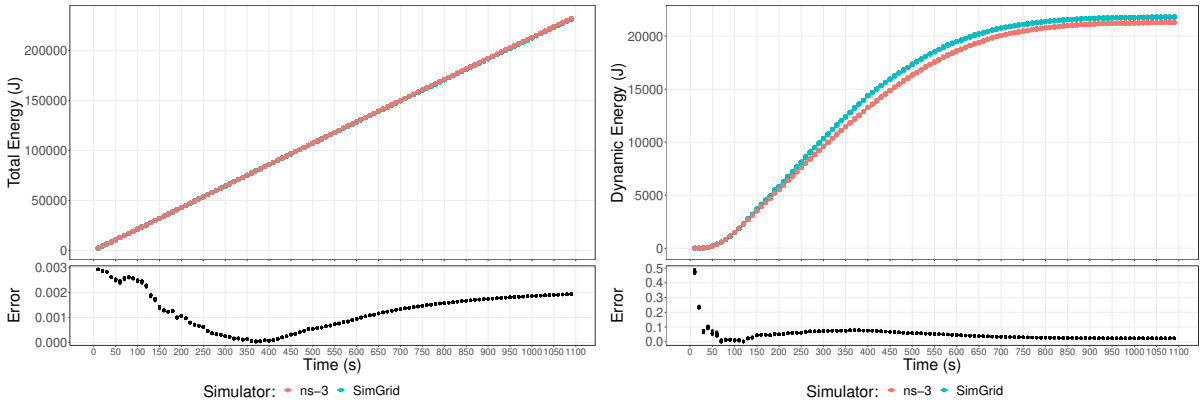
4.4.1 Energy predictions

Figure 4.6 shows the energy consumption of the overall infrastructure over time for one execution of the experiment with 15 Wi-Fi cells and a simulated time of 1100 seconds. Figure 4.6a plots the overall energy consumption while Figure 4.6b only measures the dynamic energy. In each case, we measure by steps of 10 seconds the energy consumed by the nodes and compute the relative error between ns-3 and SimGrid, visible as black points. Similar figures using other parameters are available in our online notebook [2].

We observe that, even at a larger scale, ns-3 and SimGrid compute similar energy consumption values. The relative error does not exceed 0.3% for the overall energy consumption, while it is almost always below 10% for the dynamic energy consumption. The error in dynamic energy consumption is more important during the first few time slots since only a small amount of data is going through the link, while our model is calibrated for large data throughput.

During the phase when most messages are sent (i.e. between 200 and 550 seconds), a rapid increase in the dynamic energy consumption can be observed, but this increase

2. `https://github.com/klementc/wifi_energy_experiments/blob/master/analysis/Figures_large_scale.ipynb`, last accessed December 2023



(a) Total energy consumption

(b) Dynamic energy consumption

Figure 4.6 – Evolution over time of the overall and dynamic energy consumption values between ns-3 and SimGrid for the large-scale experiment

seems to have a limited impact on the overall energy consumption of the network that remains quasi-linear.

Even at a much larger scale than the previous microbenchmarks, a flow-level model provides accurate energy prediction values compared to ns-3 within 10% of relative error.

4.4.2 Performance comparison

We now compare the memory footprint and simulation times between execution of the same scenarios (i.e. 1,100 seconds of simulated time) in ns-3 and SimGrid. Results for the memory usage are in Figure 4.7 while Figure 4.8 shows the time required to simulate each scenario with the two energy models. In both figures, we can observe that using a flow-based model increases the performance by several orders of magnitude compared to the packet-based model of ns-3.

Regarding the memory footprint, Figure 4.7 shows that simulating 55 Wi-Fi cells required up to 80 GB of memory with ns-3, while 1.2 GB was needed at maximum to study the same scenarios using SimGrid. Regarding runtime, simulations take up to 8 hours to complete using ns-3, while the runtime of SimGrid simulations does not exceed a few seconds.

Comparing the performance gains between a packet-based energy model and our approach, we can see the opportunity of our solution to perform large-scale studies with better performance than the packet model of ns-3, while keeping comparable accuracy.

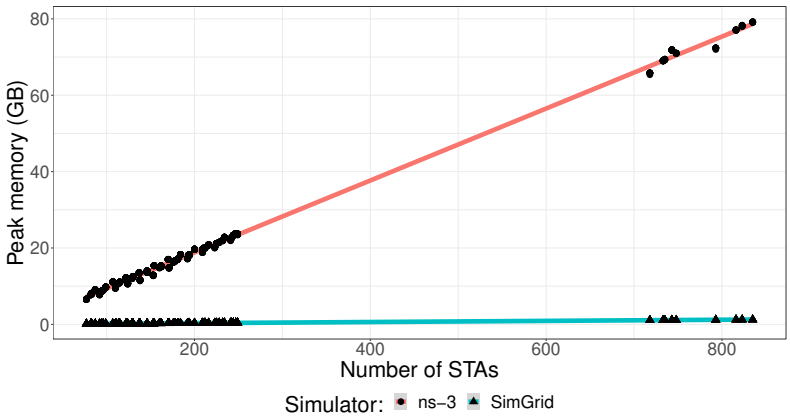


Figure 4.7 – Memory footprint comparison of SimGrid and ns-3 simulations (1,100 seconds of simulated time)

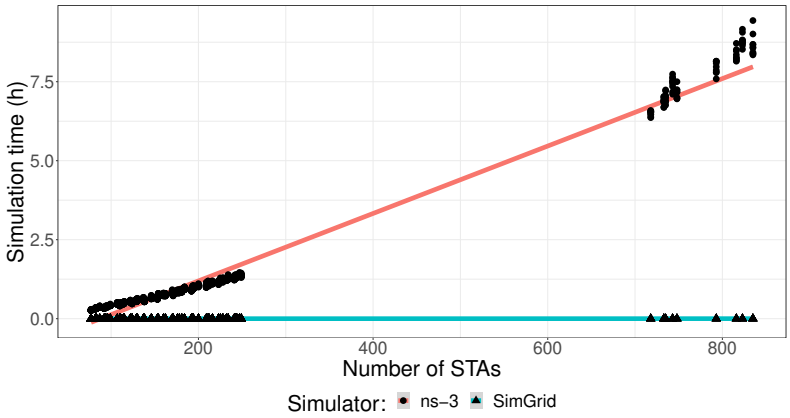


Figure 4.8 – Simulation time comparison of SimGrid against ns-3 (1,100 seconds of simulated time)

4.5 Threats to validity

Section 4.3 and 4.4 show the validity of the flow-level energy model in different use cases. However, the model can be limited under certain conditions. Below is a list of issues not directly taken into account by our model.

The Wi-Fi performance model only models the successful application throughput on the link while bad conditions may result in an important amount of errors and retransmissions. One solution to account for data loss due to retransmissions in energy predictions is to multiply the energy values of Table 4.1 by the average number of retransmissions in the channel, but this remains to be done and carefully validated.

Our validation experiments do not account for hidden nodes, which can lead to col-

lisions when two STAs are not in range and start sending data simultaneously. The collisions occurring on STAs in range with both emitting nodes are difficult to detect for the emitters. This leads to a reduced application throughput. A model where STAs subject to hidden node issues consume more energy due to retransmissions could improve the accuracy of predictions in this context, but remains to be validated.

Our model does not reproduce the variability observed in the results of ns-3 with different random seeds. This variability depends on ns-3's computations for backoffs, errors, and retransmissions. The energy model should give accurate predictions without modifications if the underlying performance model from Chapter 3 considered these phenomena.

Regarding the energy consumption of control frames, we only considered beacons and ignored other frames such as association requests that are very sparse in the studied use cases. In a network with many association frames, their energy consumption could be estimated using the same approach, and by recalibrating the model to account for them.

Since the hardware of APs is often different than the hardware of STAs, another possibility is to have separated static power consumption values between STAs and APs. It would be possible to model this difference by having different power calibration values or using a multiplicative factor: one for STAs and the other for the AP.

4.6 Conclusion

This chapter introduces a flow-level model to simulate the energy consumption of Wi-Fi NICs. This linear model computes the energy consumed by Wi-Fi interfaces when sending or receiving data, as well as the energy consumed by control frames in the network.

The experimental results on both small and large-scale infrastructures show that the implementation of the model in SimGrid provides energy predictions within 10% of ns-3 predictions in a large set of scenarios while requiring much fewer resources than ns-3. While ns-3 performs more fine-grained simulations, the flow-level approach enables large-scale experiments without extensive runtime and memory usage.

The model presented in this chapter and the Wi-Fi performance model introduced in Chapter 3 can be used to perform accurate simulations of communication times and energy consumption in IEEE 802.11n. The use of these models along with other models for wired communications would allow studying the impact of different application structures and deployment configurations on the network's energy consumption.

Chapter 5

Automated performance prediction of microservice applications using simulation

Chapters 3 and 4 introduced scalable simulation models for the performance and energy consumption of Wi-Fi communications. The combination of these models with already developed energy and performance models enables the simulation of realistic infrastructure with heterogeneous devices and communication technologies.

In this chapter, we focus on the applications executed by these simulated infrastructures. We propose a model to simulate the execution of microservice applications. Since these applications are complex and require several interactions between different actors, we also propose a methodology to transpose real applications into their simulated equivalent.

5.1 Overview

As explained in Section 2.2.1, the microservice approach is very popular and used by some of the largest internet actors, such as Twitter, Netflix, Uber, and WeChat [55].

Despite their advantages, microservices require complex interactions to fulfill requests. Large applications can be composed of hundreds of services and serve huge workloads. With this complexity, optimizing the deployment settings of microservices is challenging. Given an application and a constrained infrastructure budget, one must evaluate and answer the following questions:

Q1. How will the execution times of a microservice react to a variable load?

Q2. How would a CPU upgrade improve the maximum sustainable load?

Q3. Will distributing microservices on more than one node increase performances?

Q4. How to optimize the location of services in a computing cluster to obtain the best performances?

To answer these questions, we try to use simulation techniques adapted to microservice performance studies. Our analysis of previous contributions about microservice simulation in Section 2.4.3 has shown a costly process to transpose real applications into their simulated twins. There is a need for a methodology to ease the simulation of real applications.

Our contributions are the following:

— In Section 5.2, we propose a model for microservice-based applications. This model is simple enough to be calibrated without extensive work while being complete enough to simulate real applications.

— In Section 5.3, we propose a methodology to instantiate our model and accurately study the performances of real applications.

— In Section 5.4, we validate the accuracy of our approach using a set of microbenchmarks as well as real use cases based on existing microservice benchmarks.

Finally, Section 5.5 concludes this work.

5.2 A simple microservice model

Modern applications are composed of many services interacting together to fulfill requests. To understand the performance of a complete microservice application, one must first understand the behavior of single, isolated microservices. A microservice offers a well-defined interface, constantly listening for incoming requests. When a request is received, it triggers internal functions computing a result that is then returned to the initiator of the request.

We introduce a model representing microservice request executions. Our design goal for this model is to be as simple as possible to enable fast calibrations while being accurate enough to represent real microservices. Figure 5.1 describes the design of our internal microservice model as a 3-stage pipeline. The time spent by a request in each stage depends on both the state of the service when the request is received (queuing, resource overload, etc.), and intrinsic service properties such as its degree of parallelism. The total execution time of request r in service s is given by:

$$D_{exec}(r, s) = d_{queue}(r) + d_{CPU}(r) + d_{IO}(r) + d_{comm}(r)$$

where $d_{queue}(r)$ is the queuing time of request r before starting its execution, $d_{CPU}(r)$ is the CPU time to execute request r, $d_{IO}(r)$ the amount of time waiting for I/O operations, and $d_{comm}(r)$ the time spent communicating with external services. We now detail these factors.

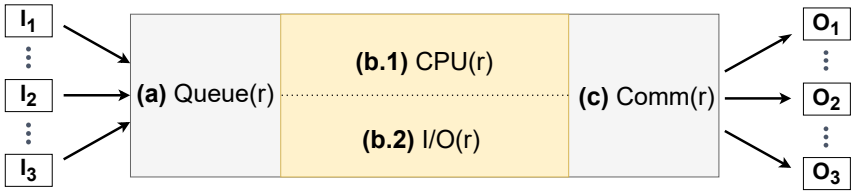


Figure 5.1 – Intra-service execution pipeline.

(a) Queuing time: On Figure 5.1, when a request is sent to a service, it can experience variable queuing times depending on the service's state. Most services limit their maximum amount of concurrent requests to avoid resource overloads and performance degradation due to the system's context switches. In our model, a service comprises a waiting queue where incoming requests are stored until their execution starts. The time spent by a request in the reception queue is dependent on both the input load and the scale of the deployed application. Through vertical scaling, the service is deployed on more efficient resources, leading to reduced execution times during step **(b.1)**, and an increased maximum load capacity. Horizontal scaling does not improve the execution time of single requests but allows for more requests to be executed in parallel through the use of several service instances during step **(b)**.

(b.1) CPU usage: A request starts executing once an execution slot is available in an instance of the service. The duration of a request execution $d_{CPU}(r)$, depends on its CPU usage r and the number of requests executed concurrently. The machine executing the request has a limited amount of CPU resources (in flops) and shares them between all active requests on the host. The cost associated with a request execution further depends on its type. A single microservice can offer more than one function through its interface, each of them leading to different execution times. Thus, the CPU usage of a service corresponds to: the provisioned capacity of the executing node, a mapping of request type to CPU costs, and the maximum amount of concurrent executions.

(b.2) I/O idle time: A service execution does not only require CPU processing but also I/O operations. In some cases, I/O can be overlapped with CPU executions. In other cases, I/O can result in periods where the CPU remains idle. We define the time spent in I/O by using an *active ratio* that represents the time spent doing I/O compared to pure

CPU execution. Currently, we model I/O as a simple delay, thus we do not consider disk contention.

(c) Service output: Most microservices request data from other services. Using our model, a request can be forwarded to other services once it has finished CPU execution **(b.1)** and waiting for I/O to finish **(b.2)**. The type of the request defines the list of services to contact. If the output services called during this step are running on different computing nodes than the current service, a network communication is initiated to forward the request. This enables the observation of performance bottlenecks due to network limitations. Microservices linked through inter-service communications form a DAG for each request type, as described in Section 2.2.1. In Figure 5.1, we separate the execution of a request from outer communications. It is a simplification compared to real microservices that often interleave executions and communications. Our approach does not represent the execution of a request at such a fine granularity but conserves the overall execution time as well as communication dependencies.

Table 5.1 gives a summary of the parameters to instantiate this model.

Table 5.1 – Summary of inter-service and intra-service properties used to calibrate our microservice model.

Granularity	Parameters
Request	- Type of the request
Intra-service properties	- CPU costs - I/O ratios - Parallelization degree
Inter-service properties	- Output services (a DAG for each type of request) - Network requests sizes

This microservice model is implemented on top of the SimGrid simulation framework and is available online [1]. It allows to define a microservice application using a simple interface with the parameters from Table 5.1. In addition to the service model, we provide interfaces to define autoscaling policies that adapt the number of replicas of each service depending on the input load. The proposed implementation includes a CPU autoscaler based on usage thresholds inspired by the literature [118]. Users can use this autoscaler and define the CPU usage threshold to automatically create or delete service instances at

1. `https://github.com/klementc/microservices_simgrid_reproducibility` code of the used model, scripts and benchmarks for the contribution's validation, last accessed December 2023.

frequent time intervals.

The model can be manually instantiated to study real applications in order to evaluate their performance in various configurations. Even if possible, the manual instantiation of the models for all services of large applications with the correct values remains a challenging task. Both service interactions and individual service behaviors need to be taken into account for possibly hundreds of services. In the following, we propose a methodology to automate the transformation of real applications into instantiated simulation models.

5.3 Modeling real microservice applications

Previous works on microservice simulation such as μqsim [107] recommend modifying the code of applications to log additional data later used as calibration values. However, manual code modification remains burdensome and error-prone for large applications. It is also not evolutive since it has to be made again when the structure of the code is modified. Our goal is to allow developers of microservice applications using state-of-the-art service monitoring techniques to automatically obtain the description of the structure of their application and the calibration values for our microservice model. This methodology does not require code modifications provided that the targeted application uses one of the standard service monitoring solutions, as detailed hereafter.

To instantiate our microservice model, we need to gather values for the parameters given in Table 5.1. The *request types* to study depend on the application and the goal of the experiment.

Intra-service properties need to be observed at both application- and system-levels. Indeed, overall service execution times can be observed easily from the application, but the ratio of active and *idle* CPU times requires lower-level information.

Inter-service information can be obtained by observing the network interactions between services running in separate containers.

From our experience, *distributed tracing* as discussed in Section 2.4.3 can provide most calibration values at a low cost. Whereas the typical use case of distributed tracing is to help identify the services inducing performance issues, we leverage it to calibrate our model. Traces entail the path followed by requests and the amount of time spent in each service. Each service execution is called a *span*, and the set of all spans linked to the same request forms the DAG of the overall execution. The collected data provide the inter-service DAGs and intra-service execution times.

Figures 5.2 and 5.3 show an execution trace automatically extracted from the DeathStarBench's social network microservice benchmark [60]. In Figure 5.2, a partial execution trace can be observed, directly taken from the web interface of Jaeger [121], whereas Figure 5.3 is a DAG representation of the same request. Figure 5.4 summarizes our approach exploiting the information contained in a single execution trace per type of studied request. Based on this example, we now show how to calibrate our model by following the steps of Figure 5.4.

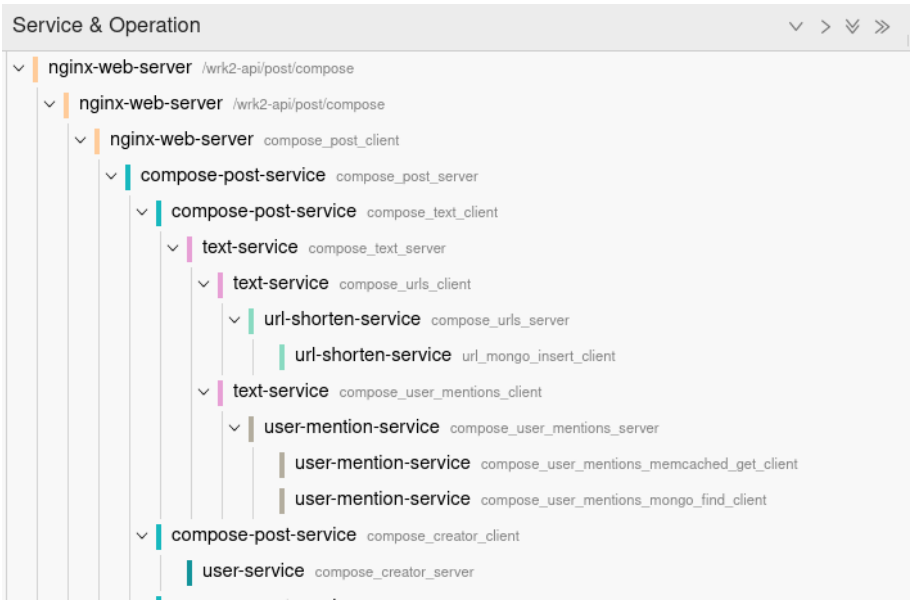


Figure 5.2 – Part of a trace from the execution of DeathStarBench's [60] social network COMPOSE request extracted from Jaeger [121]. Services implied in the execution of the request as well as the relation between services can be observed.

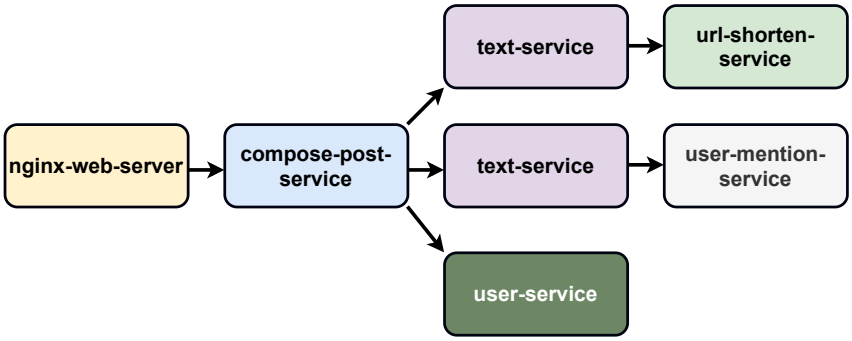


Figure 5.3 – Graph representation of the request sample from Figure 5.2.

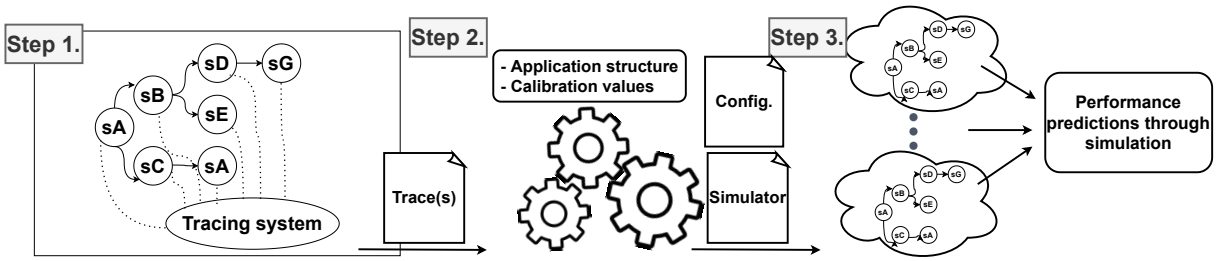


Figure 5.4 – Transformation of an execution trace in an executable simulator.

Step 1: Obtain calibration traces. The application modeler gathers execution traces of the requests to simulate. These traces are created upon request executions, with a tracing system such as *OpenTracing* [119], *OpenTelemetry* [120], or *Kieker* [133]. The user runs the instrumented application on a machine representative of the targeted platform. Finally, the traces containing the information to use in the following steps are obtained using the tracing system.

Particular care must be taken in the selection of the traces to ensure that they are representative of the behavior of the application. Using average execution times helps avoiding outliers. In the case of Figure 5.2, the application runs a medium workload for several minutes to select a trace corresponding to the application's behavior in steady-state: no overloaded services, the caches are warm, databases are populated, etc.

Step 2: Process the traces to obtain calibration values. The second step is to obtain the values required for the instantiation of our microservice model from the traces. They provide enough information to calibrate our model even if low-level metrics may be missing. In our example of Figure 5.3, a Jaeger trace contains the execution DAG of services execution. We can use it to define inter-service properties. In this example, the size of network requests is not available, but additional monitoring could for example allow us to send request sizes to *Prometheus* [134]. Intra-service properties are also partially included in the traces. We obtain an estimation of real request execution durations for each service. Still, our example trace does not provide an I/O ratio, but complementary solutions such as Docker monitoring could be used.

Step 3: Building and configuring the simulator. The values obtained during the previous steps are used to instantiate a microservice model for each service of the target application. We can then observe the duration of end-to-end request executions through simulation. Simulations can be configured to explore the performance of the application under various deployment settings. Because obtaining calibration values is relatively easy

with our approach, it is possible to adjust them rapidly after the modification of a service's implementation.

To profit from the ability of distributed tracing to provide the intra- and inter-service properties of an application, we provide a script [2] to automatically produce the code of a simulator with calibrated microservices given Jaeger traces. This allows for a semi-automatic performance simulation of compatible instrumented applications.

5.4 Experimental validation

To evaluate our contribution, we rely on both a set of microbenchmarks and published microservice benchmark use cases. We compare the simulation outputs using our approach to real application executions.

Experimental setup: All experiments have been run on the Grid'5000 testbed [71]. We run our experiments on the *paravance* cluster composed of nodes with 2 × *Intel Xeon E5-2630 v3* with 8 cores/CPU, 128 GiB of memory, and 2 × 10 Gbps network interfaces. They run Debian 10 under kernel *4.19.0-16-amd64*. Services run within *Docker containers* and multi-node deployments are done with *Docker-swarm* [135].

Source code and reproducibility: The source code of our contribution is available online at `https://github.com/klementc/microservices_simgrid_reproducibility` along with the scripts used to obtain the results. We provide notebooks with the code used to generate the figures and additional experimental results.

5.4.1 Microbenchmarks

Before predicting the performance of large-scale microservice applications, we need to ensure that our model allows for accurate execution time predictions at the scale of single services. This first experiment aims to show the ability of our microservice model to reproduce request execution times accurately under a dynamic load.

We launch a microservice application that executes a fixed amount of CPU work for each request. The microservice fetches incoming requests through a RabbitMQ queue, and it is possible to chain multiple services to obtain multi-step executions. The results detailed hereafter consist in a single service sending its results directly to a sink. We run

2. `https://github.com/klementc/microservices_simgrid_reproducibility/tree/main/script`, last accessed December 2023.

microbenchmarks using 2 chained services leading to similar results not shown here [3].

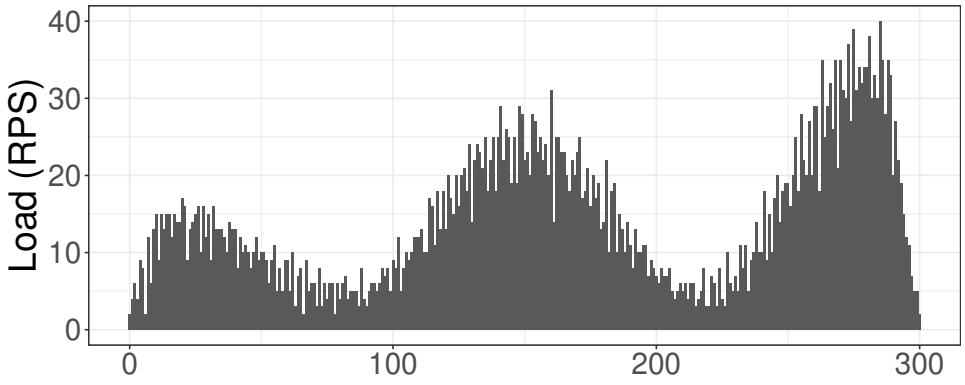


Figure 5.5 – Synthetic load generated for microbenchmark experiments: Requests Per Second (RPS) during 5 minutes with 3 load spikes.

To simulate this application using our model, we proceed in two steps. First, we obtain the execution time of a request given its CPU cost by sending calibration requests. These calibration values represent the execution duration when a service is not overloaded. Through linear regression, we estimate the duration of request executions for any CPU cost assigned to the service. We then calibrate the service model to use these values before running simulations and comparing the results to executions on a real platform.

We generate a synthetic load using LIMBO, an HTTP load model and generator [136]. It consists in requests spanning over 5 minutes with between 1 and 40 requests per second and three activity spikes for a total amount of around 4,500 requests as shown in Figure 5.5. We launch this experiment 5 times for each configuration. Configurations vary by the quantity of work to be executed and the maximum amount of parallel executions.

Figure 5.6 shows the results obtained during the microbenchmark execution with one service, a CPU cost of 25ms per request, and two concurrency degrees: five and ten. The concurrency degree refers to the maximum number of parallel requests for a given service. We restrict the application to execute using only a single CPU core. For each request executed during the experiment, Figure 5.6 shows the estimated execution time of requests obtained using our model in SimGrid and the execution times of a real deployment.

We make two observations. First, both SimGrid predictions and real-world results have higher execution times during request arrival spikes, which happen at 20s, 150s, and

3. `https://github.com/klementc/microservices_simgrid_reproducibility/blob/main/comparison/Comparison_analysis_scenario2.ipynb`, last accessed December 2023.

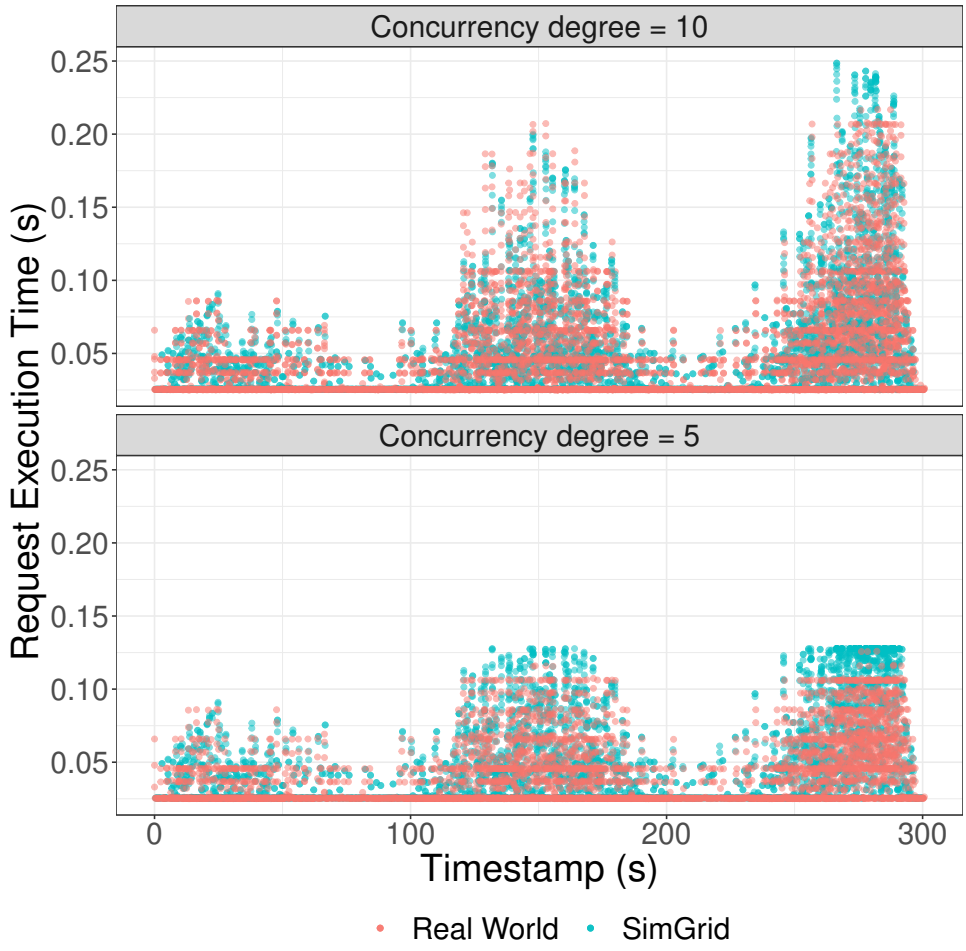


Figure 5.6 – Comparison between our model's prediction and a real execution with two different concurrency degrees under a synthetic load.

250s. This is caused by the processing of several requests in parallel and queuing. This shows the ability of our service model to reproduce the processing times of requests under a dynamic load.

Second, we observe that the execution time per request changes with the concurrency degree of the application. With a concurrency degree of 5, the maximum request execution time (125ms) is approximately two times lower than the maximum request execution time with 10 parallel requests. A smaller concurrency degree decreases single request execution time at the cost of increased queuing times. Executing an important amount of parallel requests on a single CPU core will also lead to overheads due to operating system thread switching. The execution model of SimGrid does not take into account context switching costs, thus it might underestimate execution times when the number of parallel requests

is much higher than the number of CPU cores to execute them. Yet, in the results of this chapter, this effect does not impact our observations.

These results show that our microservice model implementation in SimGrid can accurately predict the performance of simple CPU-intensive microservices under variable loads, and thus it answers the first question:

Q1. How will the execution times of a microservice react to a variable load?

5.4.2 Use-case 1: TeaStore login requests

We now observe the performance when modifying the resources (i.e. number of cores) dedicated to the execution of an application within a single computing node. This question is of importance for real deployments to estimate the cost/gain ratio of different hardware options. Our goal is to obtain the same results between real observations and our simulations when changing the number of resources to be used. To evaluate the versatility of our approach, we only rely on one calibration experiment detailed hereafter to instantiate our service models.

We run TeaStore [137], a microservice application benchmark used in microservice performance evaluation literature [138]. We focus on the *LOGIN* request of this application. This request involves 4 different services running in separate Docker containers. We study the maximum sustainable load (in *Requests Per Second, RPS*) of the application under different resource configurations and compare real results to simulated predictions.

TeaStore is natively instrumented with Kieker [133]. We use the average request execution duration of each service to calibrate our service models within SimGrid by doing a real execution under a low load profile. From this execution, we extract a trace with Kieker, providing us the calibration values. As recommended by TeaStore's documentation, the load is generated by LIMBO [136]. We benchmark the application under 3 different configurations. For the real experiments, the application is deployed on a machine with either 4, 8, or 16 cores dedicated to the execution of the services. We execute each configuration 20 times. A summary of the results is shown in Figure 5.7.

The goal of this experiment is to detect the breaking point after which the application is overloaded. This breaking point is detected by our model, for example around 320 requests per second in the configuration with 4 cores, as shown in Figure 5.7. Regarding the accuracy of our model, we observe the *mean relative error (MRE)* between SimGrid predictions and real-world values. Over the different workloads shown in Figure 5.7, we observe an average MRE of 11.8%, 4.9%, and 3.6% with maximum values of 21%, 17%,

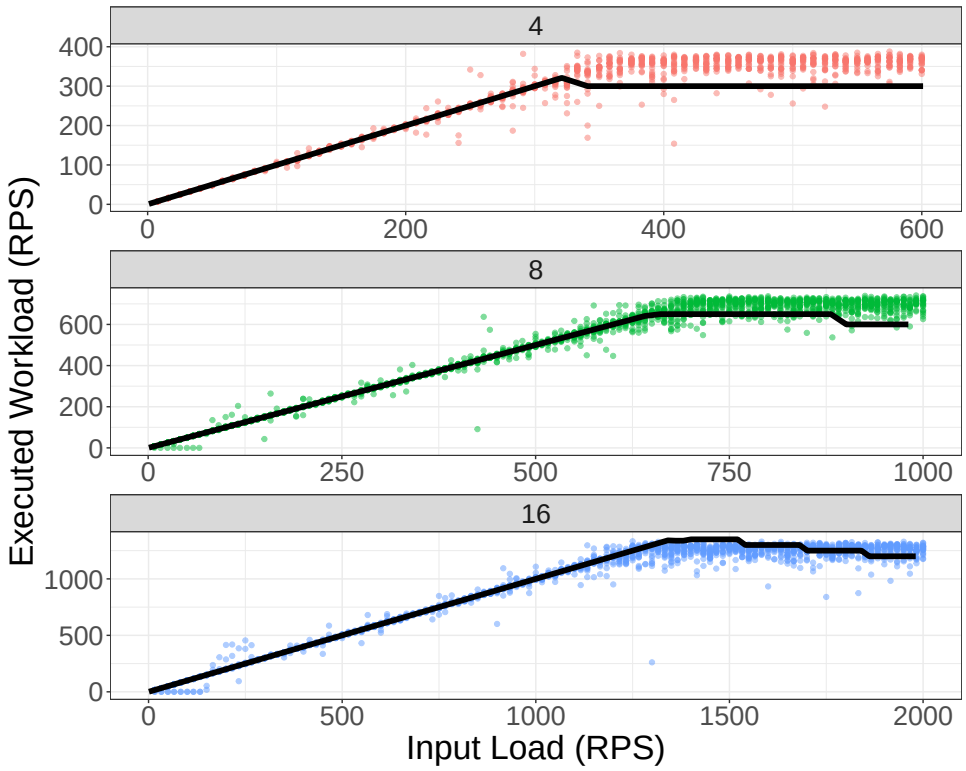


Figure 5.7 – Comparison of teastore's LOGIN performance between SimGrid predictions (black line) and real-world executions (colored dots) for 4, 8, and 16 cores dedicated to the execution of the application.

and 14.9% in the 4, 8 and 16 cores configurations respectively. While the maximum error observed is non-negligible, especially in the 4 cores configuration, the predictions of our model allow observing trends, and comparing the advantages of one configuration over another. This experiment allowed us to answer the second question:
Q2. How would a CPU upgrade improve the maximum sustainable load?

5.4.3 Use-case 2: DeathStarBench's social network

The next step is to evaluate the performance of an application running on more than one physical server, one of the main assets of the microservice architecture. Yet, finding the best partitioning of services is a complex task.

We chose to study one of the most realistic published microservice benchmarks to our knowledge, the social network from DeathStarBench [60]. We reproduce the most complex request of this application, the *COMPOSE* request that submits a publication to

the social network. This request includes more than 30 spans across 12 different services. We deploy the social network using *Docker-swarm* and vary the location of each service and the number of replicas. We compare the maximum sustainable request throughput obtained with our simulator against real executions with 10 real runs per configuration.

Table 5.2 – Service configurations for the social-network application.

	Node 1	**Node 2**
Config 1	nginx_web_server, compost_post_service, unique_id, media_service, user_service, text_service, user_mention, home_timeline, social_graph, user_timeline, post_storage_service, url_shorten	
Config 2.a	nginx_web_server, compose_post, unique_id, user_service, text_service, user_mention, home_timeline, social_graph, url_shorten, user_timeline	media_service, post_storage_service
Config 2.b	nginx_web_server, compost_post, home_timeline, social_graph, user_timeline, post_storage_service	unique_id, media_service, user_service, text_service, user_mention, url_shorten

Table 5.2 shows the server allocations for the microservices required to execute the *COMPOSE* request in each of the three studied configurations. With configuration 1, all services are using the resources of a single node. It should be the least efficient configuration due to fewer resources available for each service. Configurations 2.a and 2.b divide the 12 services into two randomly selected groups, each running on different nodes. For all configurations, other services of the application, not involved in executing the *COMPOSE* request, are running on a separate node not considered here. The SimGrid calibrated simulator is generated using the Jaeger trace partially shown in Figure 5.2 and the code generation script described in Section 5.3.

We observe one limitation of our approach during this experiment as our model does not capture the communication overhead due to *Apache Thrift* under high load. We choose to reduce this overhead (that is known to the authors of [60]) by executing two

instances of each service, as would be done in a real deployment to improve the application throughput. Such fine-grain overheads could be considered in future work.

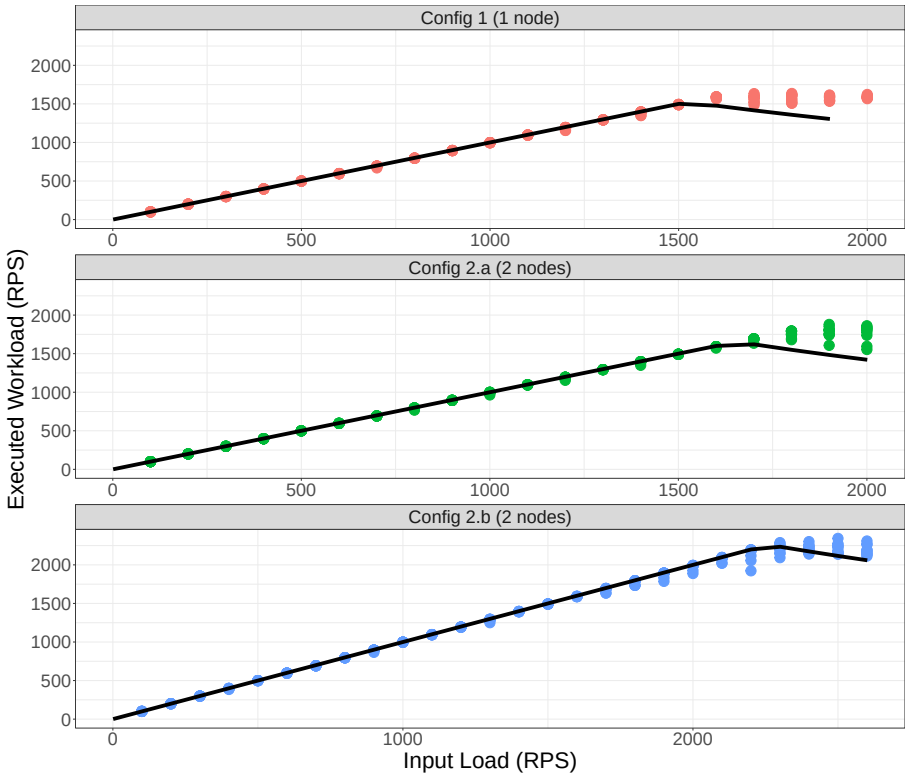


Figure 5.8 – Comparison of social network COMPOSE request between SimGrid predictions (black line) and real-world observations (colored dots) with 10 cores per node and either 1 (config. 1) or 2 (config. 2.a and 2.b) nodes.

Figure 5.8 shows the result obtained with 10 cores per node to execute the application. In this figure, we observe the maximum request throughput estimated by SimGrid and obtained through real-world executions. We observe that SimGrid accurately detects the breaking point where the application becomes saturated, at around 1,500 RPS with *1×10 cores* for configuration 1, and 1,750 RPS and 2,200 RPS under configurations 2.a and 2.b with *2×10 cores*. A non-proportional maximum throughput between the configurations can be observed. Indeed, configuration 2.a presents an unbalanced grouping of services among the nodes which leads to the overloading of one node while the other is still able to process requests. A very naive service model would predict an improvement factor of two between configurations 1 and 2.a, and would predict identical performance for configurations 2.a and 2.b. Since our model considers the processing costs of each service, it is able to show that configuration 2.b performs better than configuration 2.a, as in the

real executions.

Regarding the accuracy of our predictions, configurations 1, 2.a, and 2.b have an average MRE of respectively 3.5%, 3.7%, and 1.4%, whereas the maximum measured errors are 21%, 20%, and 6.6%. The maximum error values can be explained, as with the *TeaStore* experiment, by a different behavior of our model compared to the real application once the breaking point is reached. Before that point, the error remains very low between the simulated predictions and real measurements, while the breaking point is observable in Figure 5.8, at 1500, 1700, and 2250 requests per second in scenarios 1, 2.b and 2.b.

These experimental results show that our approach, based on a microservice model and code tracing tools, provides accurate estimations of the performance of microservice applications with different configurations, thus answering both questions:
Q3. Will distributing microservices on more than one node increase performance?
Q4. How to optimize the location of services in a cluster to obtain the best performance?

5.5 Conclusion

Microservice applications trade monolithic complexity for intricate interactions between simple services, hindering the system performance evaluation.

In this chapter, we proposed a microservice simulation model based on a reduced number of calibration values to describe microservice applications. Our contribution is more precise than large grain models while being easier to instantiate than precise models. We proposed a methodology leveraging distributed tracing systems to instantiate the simulation models of real applications using standard instrumentation solutions. Our model has been implemented on top of SimGrid, and we applied our methodology to applications instrumented with Jaeger and Kieker.

Our contributions were evaluated on microservice benchmarks, demonstrating their ability to answer the operational questions **Q.1-4** introduced in Section 5.1 on such applications. This could be used in various what-if analyses such as the exploration of performance trade-offs under scarce resources that are common in fog infrastructures. More interestingly, it could even be used to dimension a Fog infrastructure given an application and a workload to serve, an intractable problem with other solutions.

In the next chapter, we combine this microservice simulation approach and the Wi-Fi performance and energy models of Chapters 3 and 4 to study the performance and energy consumption of an end-to-end fog infrastructure executing a microservice application.

Chapter 6

Studying the end-to-end performance, energy consumption, and carbon footprint of fog applications

---

After examining state-of-the-art simulation models for the study of the energy consumption of ICT infrastructure in Chapter 2, we put existing works in 2 different categories: models intended to study a single part of ICT infrastructures, and models allowing end-to-end analysis of ICT infrastructures and their applications. We observed two issues with state-of-the-art single component models: **a)** the detailed results of fine-grained models prevent their use for large-scale simulations, and **b)** the focus on the simulation of a single part of ICT infrastructures requires the combination of several models in different frameworks. The end-to-end simulation tools studied in Section 2.3.1 are often limited in terms of validity or do not permit studying heterogeneous infrastructures accurately.

In this chapter, we propose to combine the validated simulation models introduced in this thesis with other models implemented in the SimGrid simulation framework. Contrarily to some other solutions, these models have been validated, use flow-level simulation, and are all implemented in a single simulation framework. This chapter illustrates one possible purpose of the models introduced in Chapters 3, 4, and 5. Other possible uses of these models include the comparison of different autoscaling policies for microservice applications in the fog or the study of the impact of different energy production methods on the GHG emissions of a datacenter.

We use these models to simulate a microservice application running in a fog infrastructure from the end-user devices up to the fog computing nodes. The rest of this chapter is organized as follows. Section 6.1 describes an end-to-end model for fog infrastructure and applications. Section 6.2 proposes an overview of the metrics available using our approach

through a use-case inspired by the literature. Section 6.3 concludes this work.

6.1 End-to-end modeling of a fog infrastructure and its applications

We propose a generic description of fog infrastructures coupled with validated models allowing the simulation of devices, network interfaces, and their energy consumption. The models are all implemented in SimGrid and can be used simultaneously to study various scenarios. We describe fog hardware in Section 6.1.1. In Section 6.1.2, we describe fog microservice applications, before focusing on energy parameters in Section 6.1.3.

6.1.1 Infrastructure model

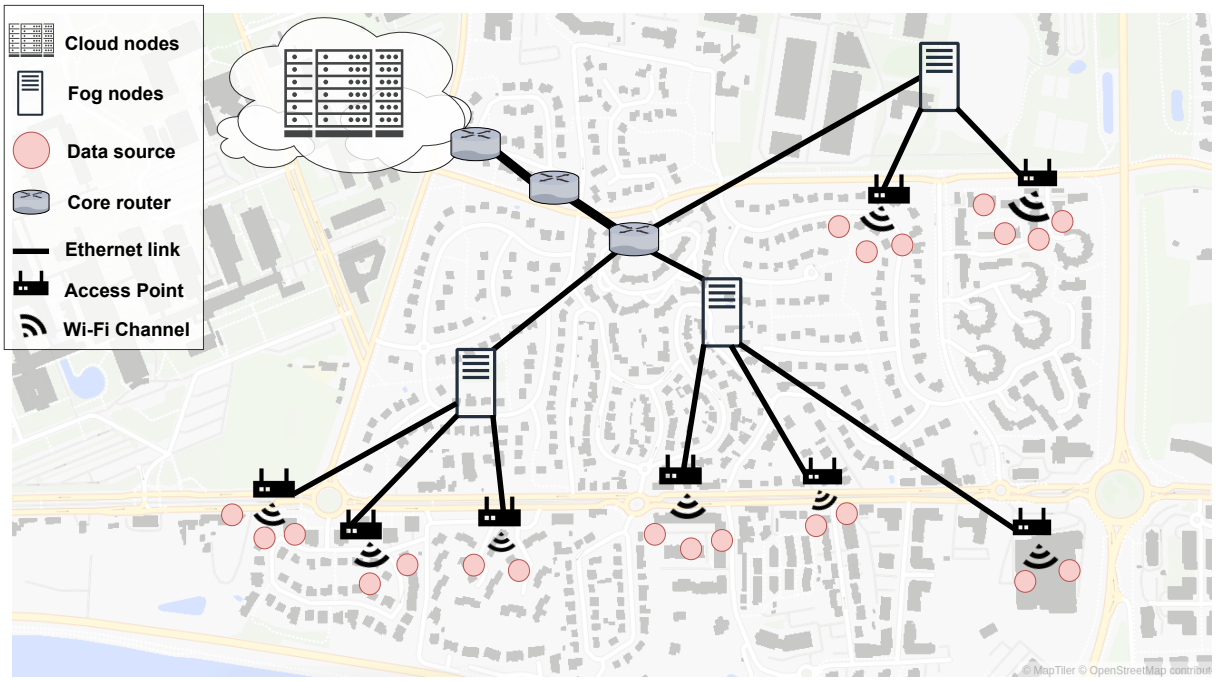


Figure 6.1 – Overview of the different actors in an end-to-end fog infrastructure

Figure 6.1 presents an overview of the components common to all fog infrastructures. We divide it into three parts: **1)** end-users, **2)** fog micro-datacenters, **3)** cloud datacenter. This infrastructure corresponds to a graph $G(N, L)$. In this graph, vertices $n \in N$ are

computing nodes or routers and edges $l \in L$ are network links (wired or wireless) between pairs of nodes.

Computing nodes

All machines in the fog infrastructure are capable of executing tasks. The maximum number of concurrent task executions on a node corresponds to its number of CPU cores N_{core}. Each core has a processing capacity C_{core} expressed in flops. Thus, nodes can execute at maximum $N_{core} * C_{core}$ flops. Table 6.1 provides realistic calibration values for a server from the experimental testbed Grid'5000 [71] and a Raspberry Pi 4 model B. To simulate task executions, we use the execution model of SimGrid. As explained in Section 2.3.2, this model, properly calibrated, provides valid results for different types of applications such as MPI [103], cloud VMs [104], and microservices as shown in Chapter 5.

Since the energy consumption of nodes depends on their CPU usage, each node has a power profile P_{prof} detailed in Section 6.1.3.

Table 6.1 – Nodes CPU capacity

	Cloud Node	Fog Node
Device	Grid'5000 Taurus [71]	Raspberry Pi 4B [139]
N_{core}	32	4
C_{core}	4 GFLOPS/core	1 GFLOPS/core

Network links

A network link enables two neighbor nodes to communicate. Different communication technologies can be used in the fog for different parts of the network. In the following, we consider Ethernet and Wi-Fi links.

As explained in Section 2.3.1, separate communication technologies need separate models to obtain accurate results. We leverage already validated flow-based models to simulate Ethernet links [88] and the Wi-Fi model of Chapter 3 implemented in SimGrid. Compared to other network models based on packet-level simulation such as ns-3, these models enable more scalable simulations while providing sufficiently accurate results for our use. Validation of the models estimated 5% of relative error on average for wired

communications in [88] and 10% on average for Wi-Fi compared to ns-3. The models' calibration requires providing the bandwidth of the network links and their latencies.

The power used by NICs also partly depends on their usage. We attach to network interfaces a power profile P_{net} used to measure power usage, detailed in Section 6.1.3. Table 6.2 provides bandwidth calibration values for different parts of the network taken from the literature.

Table 6.2 – Bandwidth of network interfaces

Interface	Bandwidth	source
Intra-Cloud	10Gbps	[140]
Core/Edge router	48x1Gbps (avg_U=25%)	[141]
Intra-Fog	1Gbps	/
WLAN	44.23Mbps	Chapter 3

Overall structure

As shown in Figure 6.1, end-user devices are connected to access points using Wi-Fi. Access Points are then connected to the fog using Ethernet links. A core network enables nodes of the fog layer to communicate with the cloud datacenter. The core network comprises a set of core routers. The core network links' bandwidth is usually much bigger than that of the aggregation network between the end users and the fog.

6.1.2 Microservice application model

As explained in Chapters 2 and 5, microservice applications are decomposed into light interconnected services. Services can run on different computing facilities. This allows, for example, to run the interface on low-latency but limited fog nodes while placing compute-intensive services further away in the cloud [142].

We use the microservice model proposed in Chapter 5 where each service is modeled as a three-step pipeline: **1)** storing received requests in a queue; **2)** executing requests using the host node's resources; **3)** forwarding the result to the output services. Service requests are characterized by their CPU cost C_{req} (in flops) and their network size $size_{req}$ (in bits). We note the ratio between the received request's size and the size of the result sent to the output services $ratio_{I/O}$. We can also leverage the CPU-based autoscaling

policy to dynamically adapt the number of service replicas. Using this model, users can generate a simulator corresponding to a real application given application-level traces. This model has been validated against real-world microservice benchmarks in Section 5.4.

6.1.3 Energy and gas emission models

Based on the infrastructure and application models, we estimate the energy consumption of an application. It corresponds to the sum of the energy of the computing nodes, the network interfaces, and the rest of the infrastructure (cooling, lighting). As explained in Section 2.3.3, the power usage of a device is the sum of **a)** the *static power consumption* P_{stat} (the minimum power to operate the device) and **b)** the *dynamic power consumption* P_{dyn} (depending on the activity of the device).

Power consumption of nodes

The static power consumption of a computing device (either cloud, fog, or end-user) corresponds to the power used by the machine when *idle*, noted P_{idle}^{node}. When devices process data, the additional dynamic power usage depends on the number of active CPU cores, their frequency, and their utilization. The device power at maximum use is P_{max}^{node}. In between, its power consumption is extrapolated with linear regression as in [103]. For core network routers, we compute their energy consumption as a ratio between the data sent by the executing applications over the total data that could be sent over the router. This approach has been used in previous works [111, 143]. In the rest of this chapter, we use the model described in Section 2.3.2 implemented in SimGrid [103].

Network links

The power consumption of a network interface depends on its communication technology. In the case of an Ethernet interface, we note P_{idle}^{eth} the power of an Ethernet interface when *idle*, and P_{max}^{eth} at maximum use. The dynamic power usage of the Ethernet interface is proportional to the utilization of the device $u \in [0, 1]$.

In Section 2.4.2, we explained that Wi-Fi NICs switch between states depending on whether asleep, idle, receiving, or sending data. Each state has a different power usage: P_{idle} when *idle*, P_{Rx} when receiving data, P_{Tx} when sending data, and P_{sleep} if the interface is in sleep mode. In this chapter, we estimate the energy of network interfaces using the

wired power model of [86] and the Wi-Fi power model from Chapter 4 calibrated with values from the literature summarized in Table 6.3.

Table 6.3 – Calibration values for the power usage of nodes and their network interfaces

Component	Parameter	Power	source
CPUs cloud	$[P^{cloud}_{idle}, P^{cloud}_{max}]$	[94.75, 178.88]W	[103]
CPUs fog	$[P^{fog}_{idle}, P^{fog}_{max}]$	[2.28, 6.82]W	[139]
CPUs end-user	$[P^{EU}_{idle}, P^{EU}_{max}]$	[2.28, 6.82]W	/
Core router NIC	$[P^{eth}_{idle}, P^{eth}_{max}]_{corerouter}$	[0, 21.168]W	[111, 141]
Edge router NIC	$[P^{eth}_{idle}, P^{eth}_{max}]_{edgerouter}$	{0, 0.441}W	[111]
Wi-Fi AP NIC	$\{P_{idle}, P_{Tx}, P_{Rx}, P_{sleep}\}_{WiFi}$	{0.82, 1.14, 0.94, 0.1}W	[113]
Routers	$P^{corerouter}_{static}$	555 W	[111, 141]
	$P^{edgerouter}_{static}$	150 W	[111, 141]
	P^{AP}_{static}	11 W	[144]
Infrastructure	PUE_{fog}	1.7	[21]
	PUE_{cloud}	1.1	[45]

Power Usage Effectiveness of the infrastructure

To estimate the additional power consumption of cloud and fog datacenters for actions such as cooling or lighting, we use the PUE. As explained in Chapter 2, the value of the PUE depends on the location and scale of datacenters [21]. In this chapter, we consider a PUE of 1.1 for cloud datacenters [45] and 1.7 for fog micro-datacenters [21], as summarized in Table 6.3. We multiply the PUE by the power consumption of the datacenter's nodes to obtain total power consumption.

GHG Emissions

Energy consumption values allow estimating GHG gas emissions. Table 6.4 provides CO_2 equivalent emissions per kWh in different countries. This value depends on the countries' energy sources. We multiply the energy consumed during an experiment by this value to estimate GHG emissions.

Table 6.4 – CO_2e emissions rates due to electricity production in different countries (expressed in $gCO2e/kWh$).

Country	gCO2e/kWh	Source
France	56	[12]
Spain	141	[12]
Great-Britain	184	[12]
USA	388	[11]

6.2 Use-case

Based on our model of fog infrastructures, we show the ability of our approach to evaluate the performance, energy consumption, and GHG emissions of an end-to-end fog infrastructure executing a microservice application.

6.2.1 Setup and methodology

The application under study is inspired by the use-cases of IFogSim [98]. It processes data from video cameras to detect movements using four services. Figure 6.2 shows the DAG of this application, and Table 6.5 the request processing cost for each service. Cameras send requests to the *Motion Detect* service to detect an object's movement in images. *Object Detect* identifies the object and sends results to *Object Tracker* and *User Interface*. *Object Tracker* is in charge of reorienting the camera to follow the movement, while *User Interface* displays results to the user. We execute more or less compute-intensive scenarios using different values of the computing ratio ρ: $\rho \in \{0.1, 0.5, 1\}$. This ratio is multiplied by the default cost of a task when executing a request. The size of network requests also changes to simulate a more or less data-intensive application: $size_{req} \in \{80kb, 1Mb\}$. We simulate scenarios with both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communications to observe the impact of the network protocol on energy and performance. We compare two application placement policies. **a)** cloud only: deployment of services in the cloud datacenter, **b)** fog only: deployment of services using fog nodes, each micro-datacenter having a copy of the application.

The application runs in the infrastructure depicted in Figure 6.1. This platform includes one cloud datacenter of 64 servers and 14 fog micro-datacenters. A micro-datacenter has 64 Raspberry Pi model 4B. Four 802.11n access points are connected

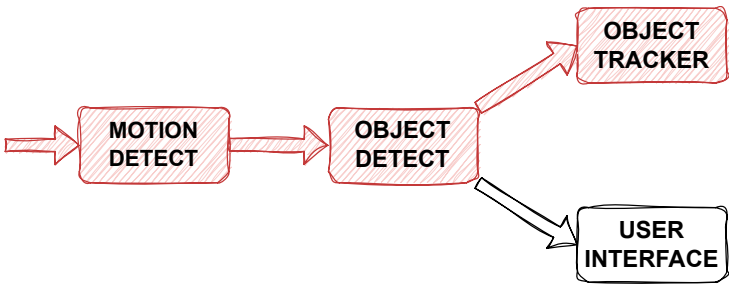


Figure 6.2 – DAG of the application, inspired from the surveillance camera use-case of [98]. The critical path is in red (used to measure end-to-end request latency).

Table 6.5 – CPU cost of executing a request in the application's services. $\rho \in \{0.1, 0.5, 1\}$ allows modifying processing intensity.

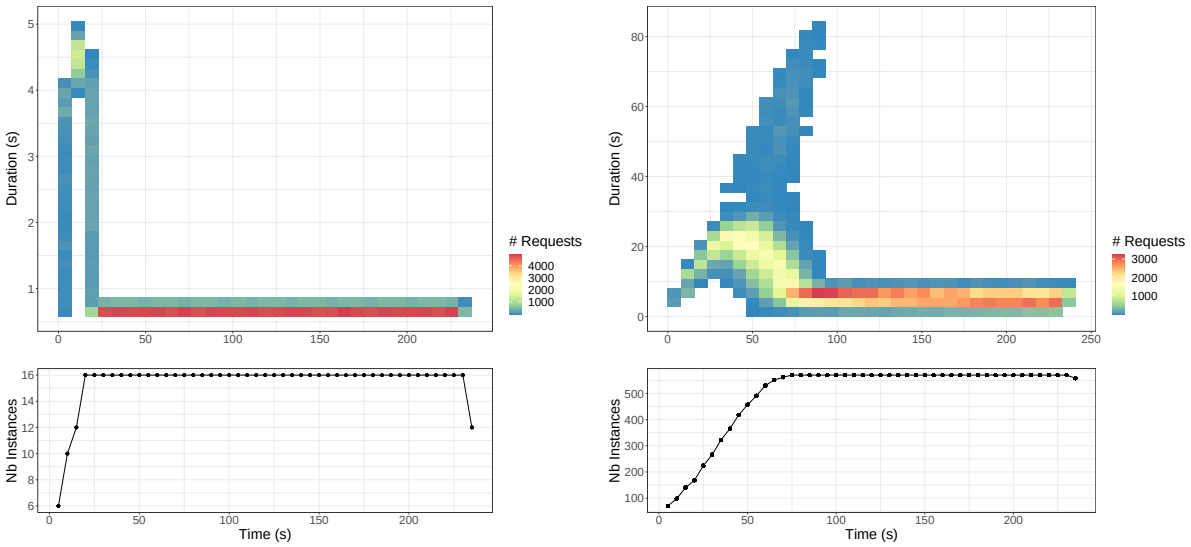
Service	C_{req} (MFLOPS)
MOTION DETECT	$\rho * 500$
OBJECT DETECT	$\rho * 250$
OBJECT TRACKER	$\rho * 500$
USER INTERFACE	$\rho * 250$

to each fog. Four cameras are connected to each access point, for a total of 224 cameras.

The cameras send between 1 and 5 requests per second. Each simulation runs for 230 seconds before stopping the cameras. We execute simulations with different distances between the fog and the cloud datacenters by modifying the number of intermediate core routers: 2, 5, and 7. Each core router adds 5ms of latency. The latency between micro-datacenter nodes varies between 2 and 5 ms.

At the beginning of the simulation, we deploy only one instance of each service. Every 5 seconds, a reactive autoscaler based on CPU thresholds can deploy additional service instances. If the average CPU usage of the nodes hosting the service exceeds 70%, the autoscaler creates a replica. Nodes that do not host any service are powered off at the beginning of the scenario and turned on by the autoscaler when needed.

All simulations use the models described in Section 6.1, implemented in SimGrid. The code, scripts to reproduce results, and notebooks used to generate the data of this chapter are online: `https://github.com/klementc/end-to-end-fog-reproducibility`. Additional results not covered in this chapter can be visualized at `https://klementc.github.io/end-to-end-fog-reproducibility/energy_analysis/visualize/`.



(a) End-to-end execution time of requests under cloud-only deployment policy.

(b) End-to-end execution time of requests under fog-only deployment policy.

Figure 6.3 – Simulating the execution time of requests under different deployment policies

6.2.2 End-to-end latency of requests

In this experiment, we study a processing-intensive scenario with high CPU usage and relatively small network requests. Network communications use TCP, and the size of each request is 80 kb. There are seven hops to reach the cloud datacenter from the fog, a latency of 2ms between fog nodes, and 0.5ms between cloud nodes. The CPU execution ratio is $\rho = 1$, and $ratio_{I/O} = 1$. In Figure 6.3, we show the evolution of end-to-end request execution times and the total number of service instances during the experiment. Figure 6.3a shows results for the cloud deployment and Figure 6.3b for the fog deployment.

At the beginning of the simulation, the end-to-end latency rapidly increases in both cases because of the time taken to scale the application to the workload. The cloud deployment can manage the load with 16 service instances after 20 seconds. Since fog nodes are less powerful, more than 550 service instances are used over all fog datacenters. Scaling takes longer in the fog because the autoscaler adds one service replica per trigger.

Once the number of replicas is stable, we observe more variations between the end-to-end latency of fog requests compared to cloud ones. A large number of service replicas increases the number of possible execution paths.

In this experiment, execution times dominate communication times. Lower ρ values

can invert this trend, decreasing execution times and leading to faster fog end-to-end duration.

6.2.3 Impacts of network configuration on performance

Table 6.6 compares output metrics between different network configurations. In all scenarios, $\rho = 0.1$, and $size_{req} = 1Mb$. This means the application is network intensive.

Com_{first} is the average communication time between end-users and the first service. Com_{other} is the average service-to-service communication time. When the distance between the end-user and the cloud increases from 2 to 7 core hops, we observe a significant increase of Com_{first}: 198% under TCP and 134% using UDP. It does not affect Com_{other} since the latency between cloud nodes does not change. In the fog, modifying the latency between fog nodes from 2 to 5 ms does not impact significantly the latency between end-users and the first fog service but increases the latency between services (200% with TCP, 128% with UDP). Regarding execution time, the average execution time of requests D_{exec} is higher for the fog application since fog nodes have fewer CPU resources.

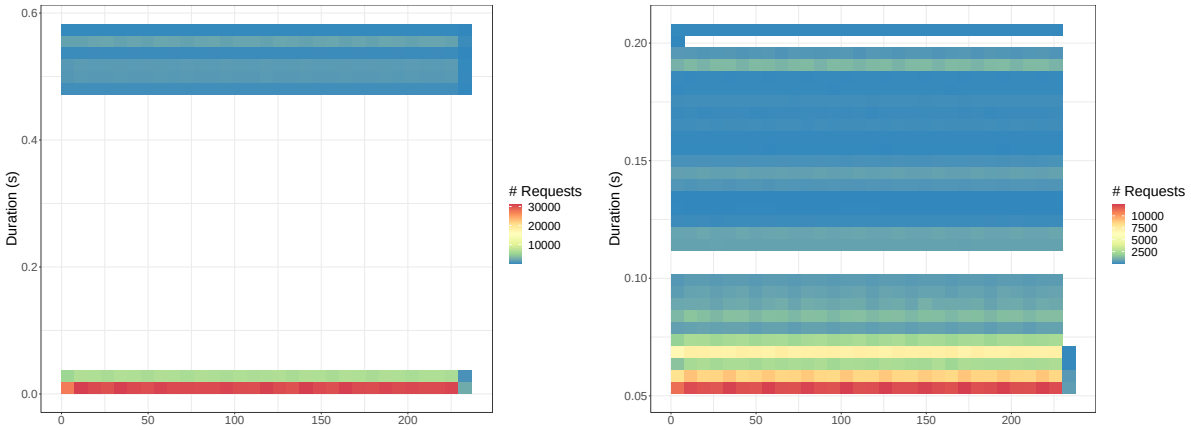
Table 6.6 – Impact of network configuration on application metrics in the network-intensive scenario. Energy results do not consider power usage effectiveness.

		Cloud (#Hops)			Fog (lat)	
		2	5	7	2ms	5ms
Com_{first}	TCP	331ms	526ms	657ms	162ms	197ms
	UDP	78ms	94ms	105ms	71ms	73ms
Com_{other}	TCP	16ms	16ms	16ms	74ms	148ms
	UDP	4ms	4ms	5ms	21ms	27ms
D_{exec}	TCP	9ms	9ms	9ms	148ms	149ms
	UDP	9ms	9ms	9ms	149ms	149ms
E_{netdev}	TCP	66.0kJ	66.7kJ	72.8kJ	70.2kJ	70.8kJ
	UDP	65.7kJ	66.3kJ	66.7kJ	67.0kJ	68.0kJ
$E_{routers}$	TCP	3.7kJ	9.2kJ	12.8kJ	0kJ	0kJ
	UDP	3.7kJ	9.2kJ	12.8kJ	0kJ	0kJ
E_{hosts}	TCP	425.1kJ	425.0kJ	425.1kJ	368.5kJ	368.5kJ
	UDP	424.9kJ	425.0kJ	425.0kJ	368.5kJ	368.5kJ

The network configuration has little impact on energy consumption. The energy used by network interfaces E_{netdev} and core router devices $E_{routers}$ (not considering PUE to get

only the power usage of devices) do not significantly change between scenarios. $E_{routers}$ in the cloud increases when adding more network hops due to additional core routers, while no core routers are necessary in the fog. Since the workload does not change, E_{hosts} does not significantly vary with modified network parameters.

Figure 6.4 shows the communication durations of individual requests in the cloud and fog placement policies. These durations depend on the bandwidth and latency of the traversed network links, and on the size of the network flows. In Figure 6.4a, there are 5 core network hops and 0.5ms of latency between cloud nodes in the same datacenter. In Figure 6.4b, there are 2ms of latency between fog nodes. With the cloud policy, we observe two groups of values: the first hop to reach the datacenter, and the communications between cloud nodes. In the fog, the highest communication durations are about 3 times lower than to reach the cloud. However, the minimum communication latency is much higher than in the cloud. This observation illustrates the need to consider the placement of services according to the number of network hops between services, and the importance of co-located services.



(a) Communication times of requests under cloud-only deployment policy.

(b) Communication times of requests under fog-only deployment policy.

Figure 6.4 – Simulated communication times

We executed additional experiments to show the importance of co-locating fog services to avoid too many network hops. When placing *Motion Detect* and *Object Detect* on the same fog nodes, the end-to-end latency of data-intensive applications reduces significantly. In the cloud, the impact is lower since the latency between nodes is low.

6.2.4 Overall energy consumption

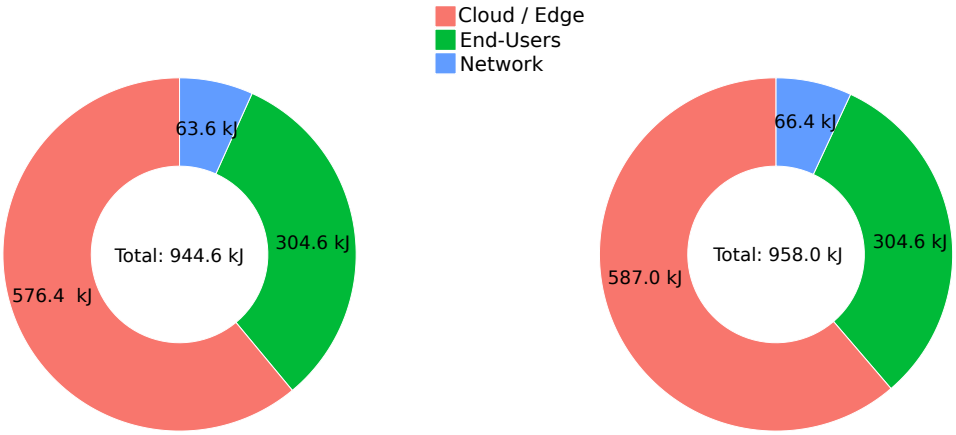
Figure 6.5 shows the energy consumption of different parts of the infrastructure. We reuse the scenario of Section 6.2.2 which is CPU-intensive. The energy consumption for the cloud placement is in Figure 6.5a, and Figure 6.5b for the fog placement.

We compute the energy of the end-users as the sum of the energy consumption of the camera nodes and the Wi-Fi APs. Network energy is the sum of the energy of the Wi-Fi interfaces, the links between datacenters nodes, and the application's use of core routers.

In Figure 6.5, we observe that network communications consume less than 10% of the overall energy. Compared to end-user devices and cloud servers, the energy consumed by the network does not have a high impact. Fog nodes consume less than cloud nodes individually, but their limited processing capacity requires more replicas for each service.

Since the PUE of a fog micro-datacenter is higher than a cloud, we observe higher energy consumption in the fog in this scenario. More energy-efficient fogs could lead to significant energy savings at runtime. In this chapter, we only consider runtime energy usage. Studying the entire life of devices (building, using, recycling) with a life-cycle assessment would be interesting since fogs require many small nodes.

In conclusion, efficient energy management in the fog is crucial to overcome cloud infrastructure performances. We observe that fog datacenters, usually having higher PUE values than high-end cloud datacenters [21], do not always permit energy gains.

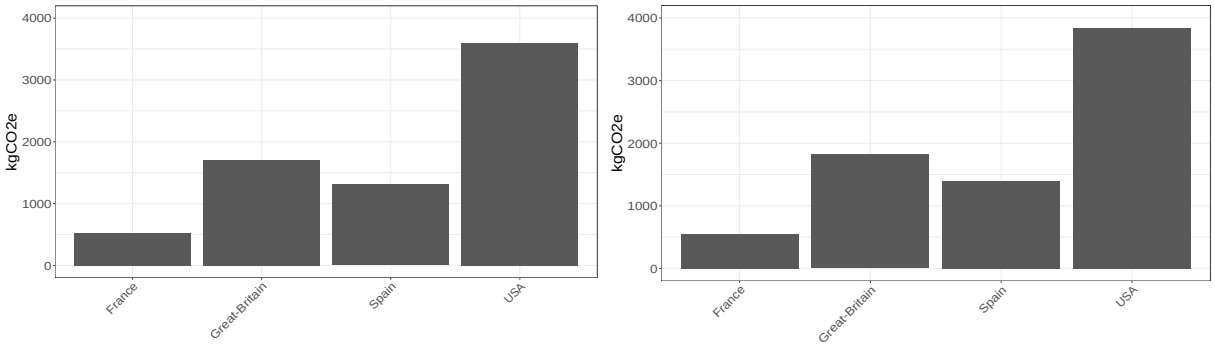


(a) Energy of a cloud deployment

(b) Energy of a fog deployment

Figure 6.5 – Estimated energy consumption in different parts of the infrastructure composed of 224 cameras, 14 fog nDC equipped with 64 Raspberry PIs, and 1 cloud datacenter with 64 servers. The simulated duration is 230 seconds.

6.2.5 GHG emissions



(a) CO_2 emissions under cloud deployment

(b) CO_2 emissions under fog deployment

Figure 6.6 – GHG emissions for the same energy consumption in different countries

Figure 6.6 shows estimated GHG emissions to run the application for one year with 224 active cameras. This is the same scenario as Section 6.2.2, except the execution ratio we set to $\rho = 0.5$. Since we do not expect the number and activity of cameras to change with time, we use the energy consumption for the duration of the simulation, and extrapolate the energy used during a year. Then we use the GHG emission rates in different countries from Table 6.4, to compute the emissions of the fog and cloud infrastructures.

We observe a high variation of emissions to produce the same amount of energy in different countries. Running a fog application in the USA, where the end-users and processing nodes are in the same neighborhood leads to more than 3.5 tCO_2e. In terms of order of magnitude, the $3.5tCO_2e$ utilized by the 224 cameras align with the average cabon emissions of using a diesel car in 2022 (130g/km [145]) for a journey of 26,900 km. If we switch from fog to cloud placement in a French datacenter, the amount of GHG drops to approximately 500 $kgCO_2e$. We also observe that the difference between emissions of a fog and a cloud application in the same country is very small since they have similar energy usage.

While local execution in fogs sometimes improves performance, the micro-datacenter's location can lead to increased GHG emissions. Traveling long distances to reach clusters using clean energy production methods can be more efficient from a greenhouse gas emissions point-of-view. Application placement policies should consider GHG emissions altogether with energy usage.

6.3 Conclusion

Despite its potential, fog computing raises questions about how to effectively leverage its benefits. Application developers need to adapt their applications to take advantage of this new architecture.

In this chapter, we proposed an end-to-end description of fog infrastructures and their applications. We leveraged existing and validated models to simulate each part of the infrastructure with accurate results. We studied a microservice application deployed under different settings using the SimGrid simulator and its models. Simulation outputs provide metrics for the application's performance but also the energy consumption and GHG emissions of the underlying infrastructure in the use phase.

Our use case shows that our approach permits to study the impact of application placement, workload, and network configuration on performance and energy consumption. We observe the importance of studying the tradeoffs between application performance and energy efficiency. Considering and improving the PUE of fogs could allow consequent energy gains. Finally, the energy savings of fog computing can lead to negative impacts in terms of GHG emissions depending on the location of fog clusters.

In the future, this approach could allow the study of different aspects of resource placement algorithms for fog applications. It also allows getting application knowledge before real deployments and playing with *what-if* scenarios.

Chapter 7

Conclusion

Concerns regarding the impact of ICT infrastructures and applications on the world's GHG emissions require accurate and scalable tools to estimate this impact and help build sustainable solutions. Cloud and fog computing paradigms rely on the collaboration of multiple actors to process and transmit data, with actors located on different machines. In this context, considering the interactions between networks, computing nodes, and end-user devices, and estimating the cost of platforms with many nodes is necessary for credible results. Consequently, the estimations' scalability, accuracy, and reproducibility are crucial. This thesis centers on using simulation to study plausible ICT scenarios in a reproducible manner. We observed the limitations of state-of-the-art contributions regarding scalability and accuracy, coupled with the fragmentation of models scattered across separate simulation frameworks. Then, we explored the opportunity of flow-level simulation to estimate the impact of end-to-end ICT infrastructures at scale.

7.1 Conclusion

The first objective of this thesis was to design performance and energy models to simulate large-scale, end-to-end ICT infrastructures. In this context, models for heterogeneous network communication technologies (wired and wireless) are necessary. While validated scalable flow-level models already exist for the simulation of Ethernet communications, available wireless models are either too fine-grained for the study of large-scale infrastructures or lack model validation. We proposed to focus on the use of Wi-Fi given the popularity and widespread usage of this technology. We extended a basic flow-level model for Wi-Fi communications with an interference mechanism to enable the simulation of additional scenarios. Then, we have shown the validity of a correctly calibrated model considering the configuration of the channel and the conditions of the studied scenarios by comparing our flow-level Wi-Fi model to the ns-3 network simulator. Simulation outputs had similar accuracy when measuring communication durations. At the same time, the

scalability of the flow-level approach reduces simulation times for 650 STAs from several hours to seconds.

Additionally, we proposed an energy model adapted to the study of the energy consumed by Wi-Fi NICs. This model has been built on top of the Wi-Fi flow-level performance model and inherits its scalability properties. Validation against the ns-3 Wi-Fi energy model shows that the results of the proposed model accurately estimate the energy consumed by Wi-Fi devices within 10% of relative error. The proposed Wi-Fi performance and energy consumption models are available in the SimGrid software distribution and can be extended and used by everyone to conduct reproducible experiments.

The second objective of this thesis was to simplify the process of simulating real-world applications deployed on distributed infrastructures. Large-scale and distributed applications such as microservices can have thousands of interdependent services located on separate computing nodes. The manual calibration of application models in this context is time-consuming and error-prone. We proposed a microservice execution model accounting for the dependencies between services, the processing costs of requests, and the ratio between I/O and CPU usage. To facilitate real-world application transposition into simulators, the instantiation of this model can be semi-automatic based on application-level traces from popular microservice monitoring solutions. We validated our model and methodology by comparing simulations to real-world executions of a popular microservice benchmark deployed under different configurations. Results show that our approach can estimate the application's performance with different service placement strategies.

The third objective of this thesis was to combine the previous models to simulate a realistic fog infrastructure running applications. We leveraged existing and validated models along with our contributions to simulate each part of the infrastructure with accurate results. We studied a microservice application deployed under different settings using the SimGrid simulation framework. Simulation outputs provide metrics for the application and network performance, the energy consumption, and the GHG emissions of the underlying infrastructure during the use phase. The analysis of the results shows that application placement, workload, and network configuration can significantly impact performance and energy consumption. We observe that the tradeoff between application performance and energy efficiency needs to be further studied while improving the PUE of fog nano datacenters could allow substantial energy gains.

The contributions of this thesis extended the use of flow-level models to consider wireless network communications and to enable the study of heterogeneous infrastructures.

We considered the tradeoff between accuracy and scalability to simulate networks at scale while ensuring credible outputs.

7.2 Future directions

Our contributions help to simulate large-scale, heterogeneous, end-to-end ICT infrastructures. However, they are based on hypotheses limiting the study of some research questions. This section explores possible extensions of our work to address some limitations of our contributions.

7.2.1 More heterogeneous networks and machines

In Chapters 3 and 4, we design and validate models for the simulation of Wi-Fi. These models are only validated using the IEEE 802.11n standard and still rely on some hypotheses. A future direction is to extend these models to consider more modern features such as energy-saving mechanisms that we did not consider [132]. Also, additional work is needed to study scenarios where the nodes' SNRs vary with time, where loss rates are important, and with mobile devices. Using rate adaptation algorithms can help in these situations. Additionally, our contributions would benefit from an improved calibration procedure for the Wi-Fi performance model. A possible improvement of the calibration is to provide default values for typical Wi-Fi configurations depending on the MCS. Another possibility is to automate the calibration depending on the scenario under study: number and configuration of AP and STAs, interferences during the experiment.

Some components of an end-to-end ICT infrastructure have not been considered in Chapter 6, such as redundant network devices in the core network. Future work could consider the impact of those devices.

Regarding the validity of the contributions, the proposed models have been evaluated against ns-3. Additional validation against other state-of-the-art simulators (e.g. OMNET++ [93], WLAN Toolbox [84]) could be considered. Comparing the predictions of the proposed models to real-world experiments is also possible, despite the difficulty of analyzing and reproducing the behavior of real-world Wi-Fi networks.

While Wi-Fi is very popular at the edge of networks, other communication technologies and types of machines could be considered. For instance, IoT sensors are very different than the end-user nodes studied in Chapter 6. Such sensors can depend on energy sources

like intermittent renewable energy, and operate differently than the studied use case. Similarly, communication technologies such as LoRa and 802.15.4 are popular in the IoT community and consume energy differently than Wi-Fi network cards. Considering these more heterogeneous devices and protocols are promising future works.

7.2.2 In-depth study of the tradeoffs during use-phase

Chapter 6 shows the ability of our contributions to simulate an end-to-end infrastructure by combining a set of flow-level models. While we observe with this study some interesting tradeoffs between the advantages and drawbacks of cloud and fog computing, our scenario remains simplistic.

The service placement strategies used in this use case consider fog-only and cloud-only placements. Several works in the literature [146] propose more complex algorithms to optimize performance and energy consumption. Considering these allocation strategies and exploring the advantages and drawbacks of different approaches would be of interest.

Similarly, the autoscaling strategy used throughout Chapter 6 relies on CPU usage thresholds. An extension of our contributions would be to implement and evaluate more complex autoscaling strategies from the literature [117, 118]. A comparison of these policies considering energy instead of only performance metrics would be interesting.

Some methods are proposed in the literature to save energy by optimizing different components: the utilization of devices, the structure of the application, among others. Our models can help to analyze the tradeoffs between the gains of such optimizations in end-to-end infrastructures, and the impact they have on application performance.

7.2.3 Full life-cycle analysis

Our contributions focus on the energy consumption of ICT resources during the use phase. Future works could complete this thesis to consider the other phases.

During the use phase, we considered the energy consumed by datacenter infrastructures using the PUE and the emissions of the devices with publicly available data. Future works could consider the use of alternative energy production and storage technologies by modeling renewable energies or batteries. The energy distribution network could also be considered to account for the advances of the smart grid.

Another future work is the consideration of the other life-cycle steps of ICT devices. The other life-cycle phases represent a big part of the impact of devices [1]. The con-

tributions presented in this thesis do not enable the evaluation of the cost of devices' manufacturing and disposal, but their output could complete the results obtained using LCA. They can also be used to study scientific questions regarding the energy demand during the use phase for prospective scenarios.

BIBLIOGRAPHY

[1] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S Blair, and Adrian Friday, « The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations », *in*: *Patterns* 2.*9* (2021).

[2] UNFCCC, *Paris climate change conference*, 2015, URL: https://unfccc.int/sites/default/files/english_paris_agreement.pdf.

[3] ADEME and ARCEP, *Evaluation de l'impact environnemental du numérique en France et analyse prospective*, 2022, URL: https://www.arcep.fr/uploads/tx_gspublication/etude-numerique-environnement-ademe-arcep-note-synthese_janv2022.pdf.

[4] Hannah Ritchie, Max Roser, and Pablo Rosado, « CO2 and Greenhouse Gas Emissions », *in*: *Our World in Data* (2020), URL: https://ourworldindata.org/co2-and-greenhouse-gas-emissions.

[5] U Cisco, « Cisco annual internet report (2018–2023) white paper », *in*: *Cisco: San Jose, CA, USA* 10.*1* (2020), pp. 1–35.

[6] Anders SG Andrae and Tomas Edler, « On global electricity usage of communication technology: trends to 2030 », *in*: *Challenges* 6.*1* (2015), pp. 117–157.

[7] Lotfi Belkhir and Ahmed Elmeligi, « Assessing ICT global emissions footprint: Trends to 2040 & recommendations », *in*: *Journal of cleaner production* 177 (2018), pp. 448–463.

[8] Eric Williams, « Energy intensity of computer manufacturing: hybrid assessment combining process and economic input- output methods », *in*: *Environmental science & technology* 38.*22* (2004), pp. 6166–6174.

[9] Thibault Pirson and David Bol, « Assessing the embodied carbon footprint of IoT edge devices with a bottom-up life-cycle approach », *in*: *Journal of Cleaner Production* 322 (2021), p. 128966.

[10] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean, « The carbon footprint of machine learning training will plateau, then shrink », *in*: *Computer* 55.*7* (2022), pp. 18–28.

[11] *How much carbon dioxide is produced per kilowatthour of U.S. electricity generation?*, `https://www.eia.gov/tools/faqs/faq.php?id=74&t=11`, last accessed September 2023.

[12] *Bilan électrique 2022*, `https://analysesetdonnees.rte-france.com/bilan-electrique-synthese`, last accessed September 2023.

[13] Sugandha Rathi, Renuka Nagpal, Deepti Mehrotra, and Gautam Srivastava, « A metric focused performance assessment of fog computing environments: A critical review », *in*: *Computers and Electrical Engineering* 103 (2022), p. 108350.

[14] Steve Sorrell, « Jevons' Paradox revisited: The evidence for backfire from improved energy efficiency », *in*: *Energy policy* 37.*4* (2009), pp. 1456–1469.

[15] Kelly Widdicks, Federica Lucivero, Gabrielle Samuel, Lucas Somavilla Croxatto, Marcia Tavares Smith, Carolyn Ten Holter, Mike Berners-Lee, Gordon S Blair, Marina Jirotka, Bran Knowles, et al., « Systems thinking and efficiency under emissions constraints: Addressing rebound effects in digital innovation and policy », *in*: *Patterns* 4.*2* (2023).

[16] GeSI, *ICT Solutions for 21st Century Challenges*, 2015, URL: `http://smarter2030.gesi.org/downloads.php`.

[17] IIMT Union, « IMT traffic estimates for the years 2020 to 2030 », *in*: *Report ITU* 2370 (2015).

[18] *Internet of Things (IoT) and non-IoT active device connections worldwide from 2010 to 2025*, `https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/`, last accessed September 2023.

[19] Philip Sparks, « The route to a trillion devices », *in*: *White Paper, ARM* (2017).

[20] M Shirer and J Rydning, « IDC's global datasphere forecast shows continued steady growth in the creation and consumption of data », *in*: *International Data Corporation (IDC)* (2020).

[21] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner, *United States data center energy usage report*, 2016.

[22] Nurzaman Ahmed, Debashis De, and Iftekhar Hussain, « Internet of Things (IoT) for smart precision agriculture and farming in rural areas », *in*: *IEEE Internet of Things Journal* 5.*6* (2018), pp. 4890–4899.

[23] Jean-François Soupizet, « La smart city: mythe et réalité », *in*: *Futuribles 1* (2020), pp. 49–65.

[24] *Territoire intelligent*, `https://www.angersloiremetropole.fr/un-territoire-en-mouvement/territoire-intelligent/index.html`, last accessed September 2023.

[25] Daniel Zhang, Saurabh Mishra, Erik Brynjolfsson, John Etchemendy, Deep Ganguli, Barbara Grosz, Terah Lyons, James Manyika, Juan Carlos Niebles, Michael Sellitto, et al., « The AI index 2021 annual report », *in*: *arXiv preprint* (2021).

[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., « Language models are few-shot learners », *in*: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[27] *Stable Diffusion GitHub repository*, `https://github.com/CompVis/stable-diffusion`, last accessed September 2023.

[28] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool, « Ai benchmark: Running deep neural networks on android smartphones », *in*: *European Conference on Computer Vision (ECCV) Workshops*, 2018.

[29] *Blockchain Energy Consumption*, `https://www.iea-4e.org/edna/news/edna-publishes-policy-brief-on-blockchain-energy-consumption/`, last accessed May 2023.

[30] Vlad Coroama, *Blockchain energy consumption. An exploratory study*, Bern, Switzerland, Sept. 2021, URL: `https://www.aramis.admin.ch/Default?DocumentID=68053`.

[31] Alex De Vries, « Cryptocurrencies on the road to sustainability: Ethereum paving the way for Bitcoin », *in*: *Patterns* 4.*1* (2023).

[32] Georg David Ritterbusch and Malte Rolf Teichmann, « Defining the metaverse: A systematic literature review », *in*: *IEEE Access* (2023).

[33] Philippe Graff, Xavier Marchal, Thibault Cholez, Stéphane Tuffin, Bertrand Mathieu, and Olivier Festor, « An analysis of cloud gaming platforms behavior under different network constraints », *in*: *IEEE International Conference on Network and Service Management (CNSM)*, 2021, pp. 551–557.

[34] Yuhua Lin and Haiying Shen, « Cloud fog: Towards high quality of experience in cloud gaming », *in*: *2015 44th International Conference on Parallel Processing*, IEEE, 2015, pp. 500–509.

[35] L ITU-T, « Greenhouse gas emissions trajectories for the information and communication technology sector compatible with the UNFCCC Paris Agreement », *in*: *Document ITU-T* (2020).

[36] *Science Based Targets Initiative Annual Progress Report, 2021*, report, `https://sciencebasedtargets.org/resources/files/SBTiProgressReport2021.pdf`, last accessed September 2023.

[37] Miriam Börjesson Rivera, Cecilia Håkansson, Åsa Svenfelt, and Göran Finnveden, « Including second order effects in environmental assessments of ICT », *in*: *Environmental Modelling & Software* 56 (2014), pp. 105–115.

[38] UK parliament, *Energy consumption of ICT*, 2022, URL: `https://researchbriefings.files.parliament.uk/documents/POST-PN-0677/POST-PN-0677.pdf`.

[39] Laurence Williams, Benjamin K Sovacool, and Timothy J Foxon, « The energy use implications of 5G: Reviewing whole network operational energy, embodied energy, and indirect effects », *in*: *Renewable and Sustainable Energy Reviews* 157 (2022).

[40] Lisa P Nathan, Predrag V Klasnja, and Batya Friedman, « Value scenarios: a technique for envisioning systemic effects of new technologies », *in*: *CHI'07 extended abstracts on Human factors in computing systems*, 2007, pp. 2585–2590.

[41] Kang Kai, Wang Cong, and Luo Tao, « Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues », *in*: *the journal of China Universities of Posts and Telecommunications* 23.*2* (2016), pp. 56–96.

[42] Yusuke Kishita, Yohei Yamaguchi, Yasushi Umeda, Yoshiyuki Shimoda, Minako Hara, Atsushi Sakurai, Hiroki Oka, and Yuriko Tanaka, « Describing long-term electricity demand scenarios in the telecommunications industry: a case study of Japan », *in*: *Sustainability* 8.*1* (2016), p. 52.

[43] Yi Wei and M Brian Blake, « Service-oriented computing and cloud computing: Challenges and opportunities », *in*: *IEEE Internet Computing* 14.*6* (2010), pp. 72–75.

[44] *Uptime Institute Global Data Center Survey 2023*, `https://uptimeinstitute.com/resources/research-and-reports/uptime-institute-global-data-center-survey-results-2023`, last accessed July 2023.

[45] *Google Data Center PUE performance*, `https://www.google.com/about/datacenters/efficiency/`, last accessed May 2023.

[46] *Latency - the sine qua non of AR and VR*, `https://web.archive.org/web/20200213143525/http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/`, last accessed September 2023.

[47] Batyr Charyyev, Engin Arslan, and Mehmet Hadi Gunes, « Latency comparison of cloud datacenters and edge servers », *in*: *IEEE Global Communications Conference (GLOBECOM)*, 2020, pp. 1–6.

[48] Mung Chiang, Sangtae Ha, Fulvio Risso, Tao Zhang, and I Chih-Lin, « Clarifying fog computing and networking: 10 questions and answers », *in*: *IEEE Communications Magazine* 55.*4* (2017), pp. 18–20.

[49] Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju, « (How Much) Can Edge Computing Change Network Latency? », *in*: *IFIP Networking Conference*, IEEE, 2021, pp. 1–9.

[50] Sven Smolka and Zoltán Ádám Mann, « Evaluation of fog application placement algorithms: a survey », *in*: *Computing* 104.*6* (2022), pp. 1397–1423.

[51] *What is edge computing?*, `https://www.akamai.com/glossary/what-is-edge-computing`, last accessed September 2023.

[52] Davaadorj Battulga, Mozhdeh Farhadi, Mulugeta Ayalew Tamiru, Li Wu, and Guillaume Pierre, « LivingFog: Leveraging fog computing and LoRaWAN technologies for smart marina management (experience paper) », *in*: *IEEE Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2022, pp. 9–16.

[53] Loic Guegan, « Scalable end-to-end models for the time and energy performance of Fog infrastructures », PhD thesis, École normale supérieure de Rennes, 2021.

[54] Johannes Thönes, « Microservices », *in*: *IEEE software* 32.*1* (2015), pp. 116–116.

[55] Aurojit Panda, Mooly Sagiv, and Scott Shenker, « Verification in the age of microservices », *in*: *Workshop on Hot Topics in Operating Systems*, 2017.

[56] *What is Cloud Native?*, `https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/definition`, last accessed May 2023.

[57] Konrad Gos and Wojciech Zabierowski, « The comparison of microservice and monolithic architecture », *in*: *IEEE International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 2020, pp. 150–153.

[58] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek, « Monolithic vs. microservice architecture: A performance and scalability evaluation », *in*: *IEEE Access* 10 (2022), pp. 20357–20374.

[59] Abdul Razzaq and Shahbaz AK Ghayyur, « A systematic mapping study: The new age of software architecture from monolithic to microservice architecture—awareness and challenges », *in*: *Computer Applications in Engineering Education* 31.*2* (2023), pp. 421–451.

[60] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, et al., « An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems », *in*: *Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 3–18.

[61] Sachin Sharma, Navdeep Uniyal, Besmir Tola, and Yuming Jiang, « On monolithic and microservice deployment of network functions », *in*: *IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 387–395.

[62] Subhadeep Sarkar and Sudip Misra, « Theoretical modelling of fog computing: a green computing paradigm to support IoT applications », *in*: *Iet Networks* 5.*2* (2016), pp. 23–29.

[63] Redowan Mahmud, Samodha Pallewatta, Mohammad Goudarzi, and Rajkumar Buyya, « iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments », *in*: *Journal of Systems and Software* 190 (2022).

[64] Chih-Kai Huang and Guillaume Pierre, « Acala: Aggregate Monitoring for Geo-Distributed Cluster Federations », *in*: *ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 156–164.

[65] *Kubernetes*, `https://kubernetes.io`, last accessed September 2023.

[66] Adelinde M Uhrmacher, « Seven pitfalls in modeling and simulation research », *in*: *Winter Simulation Conference (WSC)*, IEEE, 2012.

[67] Steve Blattnig, Lawrence Green, James Luckring, Joseph Morrison, Ram Tripathi, and Thomas Zang, « Towards a credibility assessment of models and simulations », *in*: *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2008, p. 2156.

[68] Paul R Muessig, Dennis R Laack, and JL Wrobleski, « An integrated approach to evaluating simulation credibility », *in*: *Summer Computer Simulation Conference*, Society for Computer Simulation International; 1998, 2000, pp. 449–457.

[69] Steven N Goodman, Daniele Fanelli, and John PA Ioannidis, « What does research reproducibility mean? », *in*: *Science translational medicine* (2016).

[70] Mathilde Jay, Vladimir Ostapenco, Laurent Lefèvre, Denis Trystram, Anne-Cécile Orgerie, and Benjamin Fichel, « An experimental comparison of software-based power meters: focus on CPU and GPU », *in*: *CCGrid IEEE/ACM international symposium on cluster, cloud and internet computing*, 2023, pp. 1–13.

[71] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec, « Adding Virtualization Capabilities to the Grid'5000 Testbed », *in*: *Cloud Computing and Services Science*, ed. by Ivan I.

Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, vol. 367, Communications in Computer and Information Science, Springer International Publishing, 2013, pp. 3–20.

[72] Mulugeta Ayalew Tamiru, Guillaume Pierre, Johan Tordsson, and Erik Elmroth, « mck8s: An orchestration platform for geo-distributed multi-cluster environments », *in*: *2021 International Conference on Computer Communications and Networks (ICCCN)*, IEEE, 2021.

[73] Joe Mambretti, Jim Chen, and Fei Yeh, « Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn) », *in*: *IEEE International Conference on Cloud Computing Research and Innovation (ICCCRI)*, 2015, pp. 73–79.

[74] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, et al., « FIT IoT-LAB: A large scale open experimental IoT testbed », *in*: *IEEE World Forum on Internet of Things (WF-IoT)*, 2015, pp. 459–464.

[75] Georgios Z Papadopoulos, Antoine Gallais, Guillaume Schreiner, and Thomas Noel, « Importance of repeatable setups for reproducible experimental results in IoT », *in*: *ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, 2016, pp. 51–59.

[76] Fabrice Bellard, « QEMU, a fast and portable dynamic translator. », *in*: *USENIX annual technical conference, FREENIX Track*, vol. 41, California, USA, 2005, p. 46.

[77] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan, « Crystalnet: Faithfully emulating large production networks », *in*: *26th Symposium on Operating Systems Principles*, 2017, pp. 599–613.

[78] Luc Sarzyniec, Tomasz Buchert, Emmanuel Jeanvoine, and Lucas Nussbaum, « Design and evaluation of a virtual experimental environment for distributed systems », *in*: *IEEE Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2013, pp. 172–179.

[79] Paulo Gouveia, João Neves, Carlos Segarra, Luca Liechti, Shady Issa, Valerio Schiavoni, and Miguel Matos, « Kollaps: decentralized and dynamic topology emulation », *in*: *Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[80] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker, « Control plane compression », *in*: *Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 476–489.

[81] Léo Cosseron, Martin Quinson, Louis Rilling, and Matthieu Simonin, *TANSIVTx: Time-Accurate Network Simulation Interconnecting VMs with Hardware Virtualization Towards Stealth Analysis*.

[82] *Direct Code Execution*, `https://www.nsnam.org/about/projects/direct-code-execution/`, last accessed July 2023.

[83] Hossein-Ali Safavi-Naeini, Farah Nadeem, and Sumit Roy, « Investigation and improvements to the OFDM Wi-Fi physical layer abstraction in ns-3 », *in*: *Workshop on ns-3*, 2016, pp. 65–70.

[84] *WLAN Toolbox*, `https://fr.mathworks.com/help/wlan/index.html`, last accessed July 2023.

[85] Rohan Patidar, Sumit Roy, Thomas R Henderson, and Amrutha Chandramohan, « Link-to-system mapping for ns-3 Wi-Fi OFDM error models », *in*: *Workshop on ns-3*, 2017, pp. 31–38.

[86] Loic Guegan, Betsegaw Lemma Amersho, Anne-Cécile Orgerie, and Martin Quinson, « A Large-Scale Wired Network Energy Model for Flow-Level Simulations », *in*: *Advanced Information Networking and Applications: International Conference on Advanced Information Networking and Applications (AINA-2019) 33*, Springer, 2020, pp. 1047–1058.

[87] Loic Guegan, Betsegaw Lemma Amersho, Anne-Cécile Orgerie, and Martin Quinson, « A Large-Scale Wired Network Energy Model for Flow-Level Simulations », *in*: *International Conference on Advanced Information Networking and Applications*, Springer, 2019, pp. 1047–1058.

[88] Pedro Velho and Arnaud Legrand, « Accuracy Study and Improvement of Network Simulation in the SimGrid Framework », *in*: *International Conference on Simulation Tools and Techniques (ICST)*, 2010.

[89] Thomas J Giuli and Mary Baker, « Narses: A scalable flow-based network simulator », *in*: *arXiv preprint cs/0211024* (2002).

[90] Gilbert Anggono and Tim Moors, « FLEO: A flow-level network simulator for traffic engineering analysis », *in*: *IEEE International Telecommunication Networks and Applications Conference (ITNAC)*, 2015, pp. 131–136.

[91] Sian Jin, Sumit Roy, Weihua Jiang, and Thomas R Henderson, « Efficient abstractions for implementing TGn channel and OFDM-MIMO links in ns-3 », *in*: *Workshop on ns-3*, 2020, pp. 33–40.

[92] T. Henderson, M. Lacage, G. Riley, C. Dowell, and J. Kopena, « Network simulations with the ns-3 simulator », *in*: *SIGCOMM demonstration* 14 (2008).

[93] Andras Varga, « OMNeT++ », *in*: *Modeling and tools for network simulation* (2010), pp. 35–59.

[94] Sergio Barrachina-Munoz, Francesc Wilhelmi, Ioannis Selinis, and Boris Bellalta, « Komondor: A wireless network simulator for next-generation high-density WLANs », *in*: *2019 Wireless Days (WD)*, IEEE, 2019, pp. 1–8.

[95] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter, « Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms », *in*: *Journal of Parallel and Distributed Computing* 74.*10* (June 2014), pp. 2899–2917, URL: `http://hal.inria.fr/hal-01017319`.

[96] Kayo Fujiwara and Henri Casanova, « Speed and accuracy of network simulation in the simgrid framework », *in*: *International ICST Workshop on Network Simulation Tools*, 2010.

[97] Manoel C Silva Filho, Raysa L Oliveira, Claudio C Monteiro, Pedro RM Inácio, and Mário M Freire, « CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness », *in*: *IFIP/IEEE symposium on integrated network and service management (IM)*, 2017, pp. 400–406.

[98] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya, « iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments », *in*: *Software: Practice and Experience* 47.*9* (2017), pp. 1275–1296.

[99] Isaac Lera, Carlos Guerrero, and Carlos Juiz, « YAFS: A simulator for IoT scenarios in fog computing », *in*: *IEEE Access* 7 (2019), pp. 91745–91758.

[100] P. Velho, L. Schnorr, H. Casanova, and A. Legrand, « On the validity of flow-level TCP network models for grid and cloud simulations », *in*: *ACM Transactions on Modeling and Computer Simulation* 23.*4* (2013), pp. 1–26.

[101] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al., « The gem5 simulator », *in*: *ACM SIGARCH computer architecture news* 39.*2* (2011), pp. 1–7.

[102] *The gem5 architecture simulator: A tutorial for HPCA'23*, `https://www.gem5.org/assets/files/hpca2023-tutorial/gem5-tutorial-hpca-2023.pdf`, last accessed July 2023.

[103] Franz Christian Heinrich, Tom Cornebize, Augustin Degomme, Arnaud Legrand, Alexandra Carpen-Amarie, Sascha Hunold, Anne-Cécile Orgerie, and Martin Quinson, « Predicting the energy-consumption of mpi applications at scale using only a single node », *in*: *IEEE international conference on cluster computing (CLUSTER)*, 2017, pp. 92–102.

[104] Takahiro Hirofuchi, Adrien Lebre, and Laurent Pouilloux, « Simgrid vm: Virtual machine support for a simulation framework of distributed systems », *in*: *IEEE Transactions on Cloud Computing* 6.*1* (2015), pp. 221–234.

[105] Dilshad Hassan Sallo and Gabor Kecskemeti, « Enriching computing simulators by generating realistic serverless traces », *in*: *Journal of Cloud Computing* 12.*1* (2023), p. 36.

[106] Gabor Kecskemeti, « DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds », *in*: *Simulation Modelling Practice and Theory* 58 (2015), pp. 188–218.

[107] Yanqi Zhang, Yu Gan, and Christina Delimitrou, « μqsim: Enabling accurate and scalable simulation for interactive microservices », *in*: *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 212–222.

[108] David Meisner, Junjie Wu, and Thomas F Wenisch, « BigHouse: A simulation infrastructure for data center systems », *in*: *IEEE International Symposium on Performance Analysis of Systems & Software*, IEEE, 2012, pp. 35–45.

[109] Khaled Alwasel, Devki Nandan Jha, Fawzy Habeeb, Umit Demirbaga, Omer Rana, Thar Baker, Scharam Dustdar, Massimo Villari, Philip James, Ellis Solaiman, et al., « IoTSim-Osmosis: A framework for modeling and simulating IoT applications over an edge-cloud continuum », *in*: *Journal of Systems Architecture* 116 (2021), p. 101956.

[110] Basireddy Karunakar Reddy, Matthew J Walker, Domenico Balsamo, Stephan Diestelhorst, Bashir M Al-Hashimi, and Geoff V Merrett, « Empirical CPU power modelling and estimation in the gem5 simulator », *in*: *IEEE International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[111] Loic Guegan and Anne-Cécile Orgerie, « Estimating the end-to-end energy consumption of low-bandwidth IoT applications for Wi-Fi devices », *in*: *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019, pp. 287–294.

[112] Anne-Cecile Orgerie, Laurent Lefevre, Isabelle Guerin-Lassous, and Dino M Lopez Pacheco, « Ecofen: An end-to-end energy cost model and simulator for evaluating power consumption in large-scale networks », *in*: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2011, pp. 1–6.

[113] He Wu, Sidharth Nabar, and Radha Poovendran, « An energy framework for the network simulator 3 (ns-3) », *in*: *International Conference on Simulation Tools and Techniques (ICST)*, 2012.

[114] Daniel Halperin, Ben Greenstein, Anmol Sheth, and David Wetherall, « Demystifying 802.11 n power consumption », *in*: *International conference on Power aware computing and systems*, 2010.

[115] Pablo Serrano, Andres Garcia-Saavedra, Giuseppe Bianchi, Albert Banchs, and Arturo Azcorra, « Per-frame energy consumption in 802.11 devices and its implication on modeling and design », *in*: *IEEE/ACM Transactions on Networking (ToN)* 23.*4* (2015), pp. 1243–1256.

[116] Adrien Lebre, Arnaud Legrand, Frédéric Suter, and Pierre Veyre, « Adding Storage Simulation Capacities to the SimGrid Toolkit: Concepts, Models, and API », *in*: *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2015, pp. 251–260.

[117] André Bauer, Veronika Lesch, Laurens Versluis, Alexey Ilyushkin, Nikolas Herbst, and Samuel Kounev, « Chamulteon: Coordinated auto-scaling of micro-services », *in*: *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 2015–2025.

[118] HamidReza Arkian, Guillaume Pierre, Johan Tordsson, and Erik Elmroth, « Model-based stream processing auto-scaling in geo-distributed environments », *in*: *IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–10.

[119] *OpenTracing*, `https://opentracing.io`, last accessed September 2023.

[120] *OpenTelemetry*, `https://opentelemetry.io`, last accessed September 2023.

[121] *Jaeger*, `https://www.jaegertracing.io`, last accessed September 2023.

[122] *Zipkin*, `https://zipkin.io`, last accessed September 2023.

[123] A. Legrand, « Scheduling for large scale distributed computing systems: approaches and performance evaluation issues », HDR dissertation, Univ. Grenoble, 2015.

[124] Mirko Stoffers and George Riley, « Comparing the ns-3 propagation models », *in*: *IEEE international symposium on modeling, analysis and simulation of computer and telecommunication systems*, 2012, pp. 61–67.

[125] *SimGrid*, `https://simgrid.org/`, last accessed September 2023.

[126] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti, « IEEE 802.11 rate adaptation: a practical approach », *in*: *ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MASCOTS)*, 2004, pp. 126–134.

[127] Ping Chung Ng, Soung Chang Liew, Ka Chi Sha, and Wai Ting To, « Experimental study of hidden node problem in IEEE 802.11 wireless networks », *in*: *SIGCOMM Poster*, 2005.

[128] Mukulika Maity, Bhaskaran Raman, and Mythili Vutukuru, « TCP download performance in dense Wi-Fi scenarios », *in*: *International Conference on Communication Systems and Networks (COMSNETS)*, 2015, pp. 1–8.

[129] V. Brik, S. Rayanchu, S. Saha, S. Sen, V. Shrivastava, and S. Banerjee, « A measurement study of a commercial-grade urban Wi-Fi mesh », *in*: *ACM SIGCOMM Conf. on Internet Measurement*, 2008, pp. 111–124.

[130] Mikhail Afanasyev, Tsuwei Chen, Geoffrey M Voelker, and Alex C Snoeren, « Usage patterns in an urban Wi-Fi network », *in*: *IEEE/ACM Transactions on Networking* 18.*5* (2010), pp. 1359–1372.

[131] Hang Yang, Der-Jiunn Deng, and Kwang-Cheng Chen, « On energy saving in IEEE 802.11 ax », *in*: *IEEE Access* 6 (2018), pp. 47546–47556.

[132] Esther Guérin, Thomas Begin, and Isabelle Guérin Lassous, « An overview of MAC energy-saving mechanisms in Wi-Fi », *in*: *Computer Communications* (2023).

[133] André Van Hoorn, Jan Waller, and Wilhelm Hasselbring, « Kieker: A framework for application performance monitoring and dynamic software analysis », *in*: *ACM/SPEC Int. Conf. on performance engineering*, 2012, pp. 247–248.

[134] *Prometheus*, `https://www.weave.works/features/prometheus-monitoring/`, last accessed June 2023.

[135] *Docker Swarm*, `https://docs.docker.com/engine/swarm/`, last accessed September 2023.

[136] Jóakim v. Kistowski, Nikolas Herbst, and Samuel Kounev, « LIMBO: a tool for modeling variable load intensities », *in*: *ACM/SPEC Int. Conf. on Performance engineering*, 2014, pp. 225–226.

[137] Simon Eismann, Joakim Kistowski, Johannes Grohmann, Andre Bauer, Norbert Schmitt, and Samuel Kounev, « Teastore-a micro-service reference application », *in*: *IEEE International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, 2019, pp. 263–264.

[138] Johannes Grohmann, Patrick K Nicholson, Jesus Omana Iglesias, Samuel Kounev, and Diego Lugones, « Monitorless: Predicting performance degradation in cloud applications with machine learning », *in*: *20th international middleware conference*, 2019, pp. 149–162.

[139] Houssam Kanso, Adel Noureddine, and Ernesto Exposito, « Automated power modeling of computing devices: Implementation and use case for Raspberry Pis », *in*: *Sustainable Computing: Informatics and Systems* 37 (2023), p. 100837.

[140] IEEEStandards Association et al., *IEEE 802.3 ae-2002 IEEE Standard for Information technology*.

[141] Adrien Gougeon, François Lemercier, Anne Blavette, and Anne-Cécile Orgerie, « Modeling the End-to-End Energy Consumption of a Nation-Wide Smart Metering Infrastructure », *in*: *IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2022, pp. 1–7.

[142] Mohammad Mainul Islam, Fahimeh Ramezani, Hai Yan Lu, and Mohsen Naderpour, « Optimal Placement of Applications in the Fog Environment: A Systematic Literature Review », *in*: *Journal of Parallel and Distributed Computing* (2022).

[143] Yunbo Li, Anne-Cécile Orgerie, Ivan Rodero, Betsegaw Lemma Amersho, Manish Parashar, and Jean-Marc Menaud, « End-to-end energy models for Edge Cloud-based IoT platforms: Application to data stream analysis in IoT », *in*: *Future Generation Computer Systems* 87 (2018), pp. 667–678.

[144] *Électricité : combien consomment les appareils de la maison ?*, report, `https://agirpourlatransition.ademe.fr/particuliers/maison/economies-denergie/electricite-combien-consomment-appareils-maison`, last accessed September 2023.

[145] *Évolution du taux moyen d'émissions de CO2 en France*, report, `https://carlabelling.ademe.fr/chiffrescles/r/evolutionTauxCo2`, last accessed September 2023.

[146] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre, « An overview of service placement problem in fog and edge computing », *in*: *ACM Computing Surveys (CSUR)* 53.*3* (2020), pp. 1–35.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

école normale supérieure rennes

Titre : Simulation de bout-en-bout de la consommation d'énergie des infrastructures fog et de leurs applications

Mot clés : Simulation, fog-computing, énergie, Wi-Fi, microservices, simulation-flux

Résumé : Afin de résoudre les problèmes liés à l'augmentation du nombre de machines connectées et de réduire les latences des applications, le Fog consiste à placer des ressources de calcul et de stockage en bordure des réseaux. Cela conduit à la création de petits centres de données avec des capacités de calcul et de communication hétérogènes. Malgré les avantages de cette approche, des questions se posent concernant l'impact de ces infrastructures sur les émissions de gaz à effet de serre. Il est possible d'estimer cet impact en utilisant des plateformes expérimentales ou des modèles de consommation d'énergie. Cependant, il est nécessaire de trouver un équilibre entre la précision des résultats et la taille des infrastructures étudiées. L'objectif de cette thèse est d'étudier davantage ce compromis entre précision et passage à l'échelle pour évaluer des infrastructures fog de bout en bout. Nous proposons des modèles de simulation flux afin d'étudier les performances et la consommation d'énergie des communications Wi-Fi ainsi que pour les applications microservices. De plus, nous combinons nos contributions avec des modèles existants pour étudier une infrastructure fog réaliste de grande taille. Les modèles proposés sont implémentés et disponibles dans le simulateur open source SimGrid.

Title: End-to-end simulation of the energy consumption of fog infrastructures and applications

Keywords: simulation, fog-computing, energy, Wi-Fi, microservices, flow-level

Abstract: To reduce the latency of applications and to cope with an increasing number of networked devices, fog computing places processing and storage resources towards the edge of the networks. This leads to the creation of small data centers, with heterogeneous hardware, and communication protocols. Despite the advantages of this approach, many concerns arise regarding the share of the energy consumed by network infrastructures and their applications on the world's GHG emissions. To estimate this impact, existing works propose to make use of testbeds or models to better understand the cost of different parts of the network. However, existing works have to balance between accurate results for small-scale measurements and large-scale experiments with coarse-grained results. The goal of this thesis is to further study the tradeoff between scalability and accuracy to measure the emissions of end-to-end fog infrastructures at scale. We propose models based on flow-level simulation to evaluate the performance and the energy consumed by Wi-Fi networks and microservice applications. Furthermore, we combine our contributions with previous flow-level models to study a realistic fog infrastructure at scale. The models proposed in this thesis are available within the open-source SimGrid simulation framework, with reproducible validation experiments.