Local-First Software for Green IT

Lylian Siffre*‡, Thomas Ledoux*, Renaud Pawlak†, Jonathan Guery‡
*IMT Atlantique, Inria, LS2N, UMR 6004, F-44000 Nantes, France
{lylian.siffre, thomas.ledoux}@imt-atlantique.fr
†LocalFlow, Paris, France
renaud.pawlak@localflow.fr
‡Kapela, Paris, France
{lylian.siffre, john.guery}@kapela.fr

***Abstract*—The growing energy footprint of Information and Communication Technology (ICT) services has become a critical environmental concern. While current approaches mainly focus on optimizing existing architectures, this position paper advocates for investigating a more fundamental shift: moving from Cloud-centric to Local-First Software architectures, where data and computation primarily reside on end-user devices. Through a preliminary study, we examine the energy consumption implications of such an architectural shift. We first develop a framework identifying the main potential impacts across the service stack (server, network, and client devices). We then discuss these impacts through three real-world examples, demonstrating how Local-First Software approaches could reduce energy consumption. Our analysis reveals both opportunities and challenges in this architectural transformation. While most impacts could contribute to energy reduction, particularly through decreased server and network usage, some negative impacts emerge, mainly around synchronization and client-side computation. These initial findings suggest that Local-First Software architectures, when appropriately implemented, could significantly reduce the energy footprint of digital services. This position paper aims to stimulate discussion and research in energy-efficient software architectures, laying the groundwork for future empirical studies on Local-First Software approaches in sustainable computing.**

***Index Terms*—Local-First Software, Energy Efficiency, Green Computing, Software Architecture, Distributed Systems, Sustainable ICT**

I. Introduction

Most software developed today is Software as a Service (SaaS), based on a Cloud-centric approach. In this model, the end-user visualizes the data, performs operations on it, and the provider processes it, stores it, manages the infrastructure, and so on. From an environmental point of view, the Information and Communications Technology (ICT) sector is significantly impactful. In 2021, Freitag et al. [1] estimated Green House Gas (GHG) emissions of ICT to be around 2.1 to 3.9% of global GHG emissions. This estimation precedes the explosion of artificial intelligence (AI), which is expected to have a significant and growing impact in the future.

Thus, the field of Green IT, has been very active for reducing the environmental impact of ICT. Since the global adoption of SaaS, optimizations have mostly been studied with the existence of a necessary Cloud. We think it is possible to make energy savings through an architectural shift in SaaS development.

In this paper, we try to bring a motivated intuition to the energy savings made possible through an architectural paradigm shift, and give an answer to the question:

> "*What are the impacts of an architectural paradigm shift on the energy footprint of ICT?*"

To answer the question, we identified Local-First Software [2] as a promising software architecture. Unlike the Cloud, Local-First Software advocates for "Onloading" the computing load back to the end-user devices. That is why we first take a step back by setting the background of this article through Cloud Computing before introducing the concept of Local-First Software. We give the intuition that Local-First Software can play a role in reducing the energy footprint of ICT applications. We then motivate our position with an experiment on a well-known software and its Local-First Software compliant implementation. To better grasp the extent of Local-First Software as an energy saving opportunity, we identify a list of impacts this software architecture has on service energy consumption. We finally study the impact of Local-First Software in three real-world examples.

Along with extending the concept of Local-First Software, the main contribution of this article is:

- Our work complements research efforts on the consumption of ICT services by proposing a paradigm shift in the fundamental architecture of digital services to reduce their energy footprint.
- Our work extends the understanding of the energy-related impacts of Local-First Software by examining the comprehensive effects of a complete shift towards Local-First Software, considering impacts across the entire service architecture.

The paper is organized as follows. Section II presents the Background of this paper. Section III presents the Motivation for this paper. Section IV presents the identified energy impacts of a Local-First Software architecture. Section V explores how real-world examples would benefit from this software architecture. Section VI presents the related work. Finally, Section VII concludes the paper and discusses future work.

II. Background

The increasing reliance on Cloud Computing has revolutionized collaboration, data processing, and service deliv-

ery. However, this transformation has also led to significant challenges, particularly in terms of energy consumption and environmental sustainability. In this section, we explore the historical development of Cloud Computing, its associated limitations, and the potential of Local-First Software to address these issues.

A. Cloud Approach

Before the advent of the Internet, collaboration using computers primarily involved the physical exchange of data storage media. The Internet has revolutionized this process by enabling digital sharing, and the use of the Web and e-mail have become the dominant medium for collaboration. At the end of the 2000s, Cloud computing became the standard service architecture, imposing the SaaS (Software as a Service) model [3]. The SaaS model offers numerous advantages: (i) it enables end-users to collaborate in real-time, regardless of their geographical location, leveraging the virtually unlimited computing power of Cloud servers; (ii) it also alleviates the need for end-users to manage local storage for their data, as the service provider assumes full responsibility for data storage and management.

Despite its advantages, the Cloud approach introduces challenges, as discussed hereafter.

B. Cloud Challenges

While the Cloud approach has revolutionized service delivery, it also presents significant challenges. Centralizing data and traffic introduces inherent vulnerabilities. For example, consolidated data in centralized datacenters poses cybersecurity risks, making them attractive targets for malicious attacks [4]–[9]. Similarly, routing all traffic through centralized servers results in latency, as requests and responses traverse potentially long physical distances [10]. Furthermore, this centralized dependency creates an overreliance on server availability, rendering clients non-responsive in the event of failure.

These challenges can be better understood using the CAP theorem (Consistency, Availability, and Partition tolerance), a foundational concept in distributed systems [11]. Cloud systems often need to balance the trade-offs described by the CAP theorem. In scenarios where partition tolerance is non-negotiable due to network unpredictability, systems must choose between consistency and availability. Many Cloud services prioritize availability to ensure responsiveness, even at the cost of occasional inconsistencies, particularly in distributed databases. This trade-off has implications for latency, data synchronization, and system reliability.

Beyond these considerations, as described in the introduction, the environmental impact of Cloud infrastructures is a critical concern. The field of energy optimization in Cloud and Edge computing is highly active, with numerous strategies explored [12].

C. Local-First Software

The concept of Local-First Software was introduced by Kleppmann et al. in 2019 [2], offering a novel paradigm that bridges the gap between traditional software and modern Software as a Service (SaaS) applications. This paradigm is built on seven foundational principles (called "ideals"), each designed to address common limitations of Cloud-dependent systems while prioritizing user autonomy and efficiency.

While these principles emphasize usability, collaboration, and data sovereignty, they also imply that the software must operate efficiently on end-user devices. Recognizing this, we propose an additional principle: *Sustainability*. This new "ideal" highlights the importance of designing Local-First Software to minimize energy consumption and resource utilization, aligning it with the broader goals of Green IT.

III. Motivation

We believe that Local-First Software has the potential to mitigate the environmental impact of Cloud Computing by addressing its key inefficiencies. Traditional Cloud-based systems centralize computation and storage in energy-intensive data centers, which consume significant power for operations and cooling, contributing to global carbon emissions. In contrast, Local-First Software decentralizes these processes by leveraging the computational resources of end-user devices.

A. Rationale for Local-First Software as a Sustainable Computing Approach

By minimizing the reliance on centralized infrastructure, Local-First Software decreases the frequency and volume of data transmissions. This approach directly reduces the operational burden on data centers, which are responsible for a substantial share of ICT energy consumption.

In addition, the network infrastructure supporting Cloud services is another significant energy consumer [13], requiring continuous data transfers between clients and servers. Local-First Software alleviates this by reducing the need for real-time synchronization, especially for low-collaboration workflows. This not only lowers energy consumption but also improves accessibility for users in low-connectivity regions.

Nevertheless, it is important to emphasize that Local-First Software is not intended to entirely replace Cloud-based systems. Instead, it seeks to complement the Cloud by "on-loading" computation and leveraging the power and energy efficiency of modern end-user devices. This hybrid approach acknowledges the constraints of local devices, such as limited computational capacity and storage, while utilizing Cloud resources for tasks that exceed these limitations. Such a model ensures that both paradigms work synergistically, optimizing performance, energy efficiency, and user experience [14]–[16].

As we will advocate in the remainder of this paper, Local-First Software offers a promising pathway for addressing the environmental challenges posed by traditional Cloud Computing. Its emphasis on decentralization, network efficiency, and reduced data center dependency provides an encouraging foundation for sustainable ICT solutions.

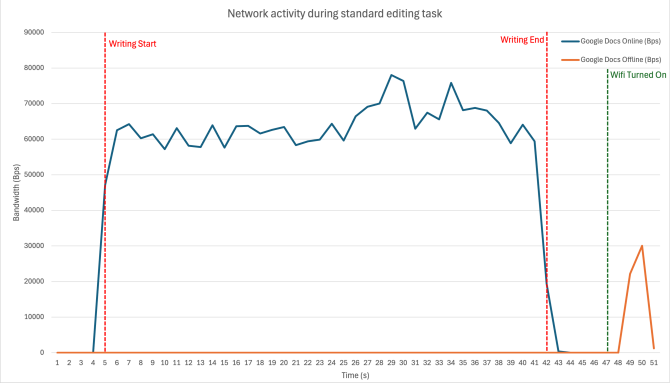
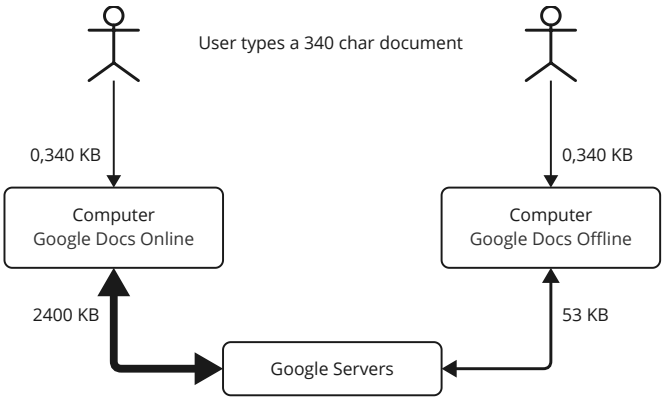


Fig. 1. Network bandwidth usage of the laptop during Google Docs experiment. Total data usage of Google Docs: Online 2400 kilobytes ; Offline 53 kilobytes.



Google Docs Online		Google Docs Offline
2400	Data transfer (KB)	53
x7000	In relation to the document's size	x156
7,6	Amount of page for $1	356

Fig. 2. Schematic representation of the interactions and data transmission between the actors of the experiment.

B. Collaborative Editor: a Motivating Experiment

To illustrate the potential of Local-First Software as a sustainable computing approach, we present a motivating experiment that highlights its advantages compared to traditional Cloud-based solutions. Our motivating experiment is the collaborative text editor. This Cloud-based service, provided by platforms such as Google Docs, enables users to collaborate seamlessly, either simultaneously or asynchronously, on a shared document hosted in the Cloud. However, this collaboration comes at a cost. We measure the resource consumption (CPU energy and network usage) of Google Docs and try to highlight the effect of a Local-First Software implementation by comparing the normal Google Docs version against the Offline version of Google Docs[1] which allows users to edit documents offline.

To measure resource consumption, we performed a brief experiment. We automate the writing of a 340-character text in a Google Docs document. During the experiment, the same Python script monitors the energy consumption of the laptop through `powermetrics`[2], network metrics are collected using Wireshark[3]. A five-second delay is introduced before and after the writing experiment, as illustrated in Fig. 1. In this figure, the blue line represents the network activity in bytes per second of the Google Docs Online experiment, and the orange line represents the Google Docs Offline experiment.

The experimental results, illustrated in Fig. 1, reveal a significant disparity in network traffic generation between Google Docs Online and Offline modes when performing identical editing operations. The Online version, depicted by the blue line, generates approximately 160 times more network traffic than its Offline counterpart. This substantial difference comes from their distinct synchronization patterns: while the Online version transmits data continuously throughout the editing session, synchronizing at every keystroke, the Offline version (shown in orange) only transmits data when network connectivity is restored, resulting in a single synchronization event after approximately 50 seconds of editing. This fundamental difference in the frequency of synchronization explains the marked contrast in the utilization of network resources between the two implementations.

Fig. 2 details the network interactions between components, highlighting this significant overhead in data transmission. This difference in network usage translates to substantial cost implications: considering current U.S. mobile data rates, editing 7.6 pages in Google Docs Online would incur the same cost ($1) as editing 356 pages in Offline mode at the same synchronization frequency.

In terms of energy, based on the measured CPU energy consumption of the laptop, our analysis found no noticeable difference in energy consumption resulting from the use of either implementation.

C. Preliminary Conclusions

Our results align with related work in [17] and highlight the significant network overhead associated with Google Docs online mode during text editing. The variation in network utilization between these two implementations is due to the remote storage of the user's document and the method of synchronization. According to our measurements, the laptop's energy consumption remains equivalent, whether storing document updates locally or transmitting them to Cloud storage. Additionally, in this single-user scenario, the server-side computational overhead is minimal as no document merging operations are required, making network traffic the primary differentiating factor between local and Cloud storage approaches.

This network overhead poses challenges in regions where bandwidth is limited or data costs are high (or during failure), potentially limiting accessibility, and increasing expenses for

[1]Google Docs Offline: https://support.google.com/docs/answer/6388102
[2]Manual page: https://www.unix.com/man-page/osx/1/powermetrics/
[3]Website: https://www.wireshark.org/

end-users who rely on mobile data. Substantial savings in data transmission (2400 KB vs. 53 KB) along with the neutrality in energy consumption of devices advocates for further study of the impacts of a Local-First Software architecture on ICT applications.

IV. Local-First Software Impacts

This section examines the implications of the Local-First Software approach on service design. By changing the locality of the data utilized by services, Local-First Software redistributes the computational load to the precise location of the data. This reallocation affects devices, networks, and servers, which represent the main sources of energy consumption in the ICT sector [13]. In the remainder of this section, we elaborate on each impact in relation to energy consumption. Each impact is denoted with either ↘ or ↗ respectively, a positive impact (i.e., reducing energy) or a negative impact (i.e., increasing energy).

A. Impact Overview

Server Impacts. The first component affected by a Local-First Software approach is the server. We identified six ways in which Local-First Software influences servers:

- (↘) **Server uptime**: Servers can be powered down while the end-user device operates autonomously. They are reactivated only when needed (e.g., for synchronization), which can decrease the service's overall energy consumption.
- (↘) **Infrastructure scale**: Because computational tasks are performed largely on end-user devices, the infrastructure can be reduced, reducing both resource and energy usage.
- (↘) **Consolidation strategies**: By defining selected time windows of activity, it becomes possible to consolidate workloads more effectively. This enables a more efficient sharing of server resources, therefore, reducing overall energy consumption [18].
- (↘) **Off-peak windows**: Providing greater flexibility in server uptime allows the servers to operate during off-peak periods. It is also desirable to take advantage of lower-carbon-intensity electricity, thus reducing the carbon footprint [19].
- (↘) **Resilience and fault-tolerance**: End-user devices act as natural redundancy nodes, reducing the need for dedicated backup infrastructure, thus reducing the energy footprint [20]. The use of conflict-free replicated data types (CRDTs) [21] further enhances system robustness while minimizing server requirements.
- (↗) **Synchronization and merge**: As Local-First Software needs to handle partitioning, servers have to merge diverging version of the same files. This load can negatively influence the energy footprint of the server.

Network Impacts. The second component is the network, where we found four main impacts:

- (↘) **Off-peak windows**: The ability to delay network usage until off-peak periods lowers the risk of packet loss due to congestion. Additionally, utilizing off-peak periods can reduce the carbon impact of data transmission [22].
- (↘) **Server to server synchronization**: With reduced reliance on server redundancy, the network overhead associated with inter-server synchronization is minimized, leading to fewer data transfers across the network infrastructure.
- (↘) **Network topology**: The ability to delay network usage until the end-user device is connected to a less energy intensive medium (i.e., Wi-Fi vs. 4G) lowers the global energy consumption of the service.
- (↘ / ↗) **Transfer sizes and frequency**: Lowering the frequency and size of data transfers saves energy in two ways: smaller data packets require less energy to transmit, and batching data transfers instead of sending them continuously can further reduce energy consumption [23]. However, in cases where data changes frequently, this might have a negative impact.

Device Impacts. The final component is the device. We identified three key impacts:

- (↘) **Networking load**: Because the dependence on remote servers decreases, devices consume less data on the network, resulting in a lower energy consumption [24], [25]. This also conserves cellular data for users with limited mobile data plans.
- (↗) **Computation load**: Under a Local-First Software paradigm, end-user devices conduct most of the computation, thus increasing their energy consumption.
- (↗) **First connection**: In a Local-First Software approach, the initial connection can be network-intensive because the device must retrieve all essential assets (e.g., product databases, JavaScript files) to operate independently.

B. Discussion

The shift towards Local-First Software architectures suggests significant potential for reducing both network traffic and server-side energy consumption. However, this naturally raises concerns about increased computational burden on end-user devices compared to traditional Cloud-based approaches.

This tension between server offloading and device care is essential to achieve genuine sustainability. While Local-First Software reduces Cloud infrastructure dependencies, it must do so without accelerating device obsolescence through excessive local computation. The environmental benefits would be negated if devices had to be replaced more frequently or suffered degraded performance.

Our experimental findings (Section III) provide an optimistic perspective on this trade-off. In the context of collaborative text editing, Local-First Software demonstrated device energy consumption comparable to Cloud-based solutions while significantly reducing network traffic. This suggests that for certain types of application, Local-First Software architectures could decrease overall system energy consumption without compromising end-user device sustainability.

The decision to adopt Local-First software is therefore presented as an optimization problem, where the objective is to minimize total energy consumption in the system as a whole, while respecting device constraints. The optimal solution depends on various factors, including application characteristics, device capabilities and usage patterns. Future work should focus on formalizing this optimization problem and identifying the conditions under which Local-First Software approaches provide the most sustainable solution.

V. Real-World Examples

To further explore how the implementation of Local-First Software affects network activity and, more generally, resource utilization, we present three real-world examples. The first is a collaborative text editor, which we have already discussed in the motivating experiment. The second is an expense management software, and the third is a catalog search engine.

A. Collaborative Text Editor

Collaborative text editors are essential tools that allow multiple users to work on the same document, often in real time. These platforms, such as Google Docs, Overleaf, and the open-source CodiMD, have become central to modern workflows in academic, professional, and personal contexts. Their ability to provide seamless updates and maintain document integrity in real-time collaboration is critical to ensuring a high-quality user experience. As demonstrated in the motivating experiment, the convenience of real-time synchronization comes at a significant cost in terms of resource consumption. Real-time editing, as seen in Google Docs, generates substantial network usage. In this section, we investigate the other effects of this real-time synchronization mechanism, along with other collaborative text editing platforms.

Methodology. To evaluate the network usage and energy efficiency of these platforms, we conducted experiments on Google Docs, Overleaf, and CodiMD. Each experiment consisted of three tasks:

- **Idle**: where the document was left open without user interaction for two minutes to simulate a reading scenario;
- **Standard Editing**: where a Python script simulated user input by typing a 340 character long predefined text;
- **Copy-Paste Editing**: where a block of text is inserted in a single action to mimic a bulk synchronization approach.

During these tasks, network activity was monitored using Wireshark[4], capturing both the total data transferred and the number of packets sent during each task. The energy consumption was measured using a Python script using `powermetrics`[5]. This setup allowed for a direct comparison of the performance of the platforms under different usage conditions.

Results. As illustrated by Fig. 3, our experiments showed that Google Docs transmitted approximately 2.4 megabytes of data for a 340-character text during standard editing – an average of 7000 bytes per character. In addition, during this experiment, Google Docs generated a significant network activity of 5973 packets, as illustrated in Fig. 4. This highlights the network inefficiency of current synchronization methods, particularly in scenarios with single-user editing.

TABLE I
Data transfer comparison between Standard Edition and Copy Paste Edition in Bytes.

Platform	Standard (Bytes)	Copy-Paste (Bytes)	Reduction
Google Docs	2380933	5386,5	**99,77%**
CodiMD	199725,4	1640,6	**99,18%**
Overleaf	15405,7	1101	**92,85%**

This problem is not unique to Google Docs. Overleaf and CodiMD, also exhibit high resource consumption, especially during Idle periods. In these periods, as Google Docs rely on HTTP requests to synchronize, if there are no updates, then no request is sent. In the other hand, CodiMD and Overleaf rely on WebSockets, explaining the network activity even during Idle. Despite the differences in their underlying technologies—Google Docs uses HTTP POST requests, while Overleaf and CodiMD rely on persistent WebSocket connections—all these platforms share the challenge of being network-intensive.

As illustrated in Fig. 3 and 4, Google Docs and CodiMD exhibit a similar network intensity, but we observe that Overleaf's network intensity is significantly lower than Google Docs and CodiMD. This difference is likely due to the synchronization mechanism employed by these platforms. In fact, Google Docs and CodiMD synchronize every keystroke, whereas Overleaf synchronizes periodically every two seconds.

Our last experiment, the Copy-Paste Editing, is conducted in order to mimic a delayed synchronization mechanism where the update is sent when the user is finished writing. This experiment demonstrated that batching updates can significantly reduce the number of packets sent and the amount of data transferred. We summarize the results for this experiment in Table I. These results show significant network activity reduction of 97.3% in average.

Discussion. This substantial network overhead is not merely a symptom of collaborative editing implementation choices, but rather stems from the fundamental architecture of Cloud-based services prioritizing real-time synchronization for fault tolerance and user data backup, and in this example, real-time collaboration. Maintaining high consistency in distributed systems requires frequent state synchronization between servers and clients. This architectural pattern, while ensuring data durability and consistency, results in significant resource consumption even in single-user scenarios.

To address inefficiencies —particularly for scenarios with a *single active* user— we propose rethinking synchronization strategies. One option, inspired by Local-First Software principles, is delayed or bulk synchronization, which involves sending updates to/from the local database only when necessary, thus reducing both transmission overheads and device energy consumption. This tactic also improves accessibility

[4]Website: https://www.wireshark.org/

[5]Manual page: https://www.unix.com/man-page/osx/1/powermetrics/

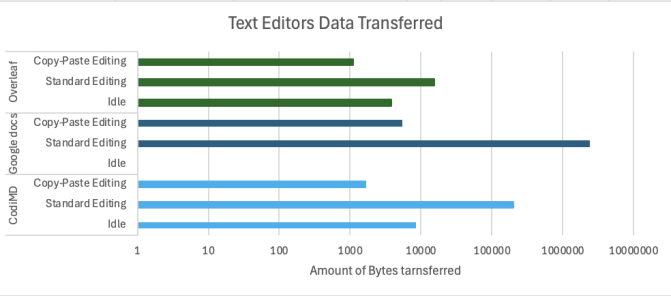


Fig. 3. Amount of bytes transferred during the experiments.

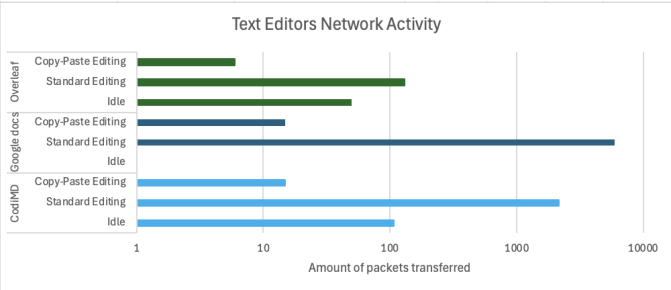


Fig. 4. Amount of packets transferred during the experiments.

in low-connectivity or high-cost environments [23], [26]–[28]. Academic efforts such as PeriText and Automerge (both leveraging CRDTs) illustrate how collaborative Local-First Software can be developed [29]–[31].

B. Expenses

Managing employee expenses is a routine task in most organizations, facilitated by software that allows employees to record their expenses and submit them for managerial review. Unlike collaborative editors, expense management software typically operates as a single-user workflow with periodic data submissions. This workflow does not require real-time synchronization or frequent updates, making it particularly suitable for Local-First Software design principles.

However, many current implementations rely on Cloud architectures, introducing inefficiencies that compromise usability and increase energy consumption. For example, during our experiment with Cleemy[6], an expense management solution, we observed that, while some static data are cached, the application still requires connectivity to the server for essential operations. If a user attempts to save an expense offline, the application displays an error message, rejecting the input and forcing the user to re-enter the data later. This dependency on uninterrupted connectivity not only has an impact on productivity, but also increases frustration, especially for employees in low-connectivity environments [32].

The Cloud approach also introduces unnecessary redundancies. In Cleemy, all expenses are stored on the server before validation, although most of the required data —such as invoices or organizational details— is readily available to

[6]Website: https://www.lucca.fr/finance/notes-de-frais/

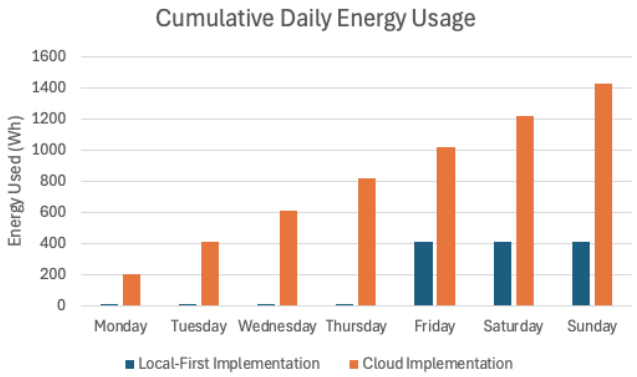


Fig. 5. Simulation of the cumulative energy consumption of the expense management software over a week for Cloud and Local-First Software implementations.

the user locally and rarely changed. By shifting to a Local-First Software architecture, the application could allow users to record and manage expenses entirely offline, synchronizing only when the user explicitly decides to submit their data. This approach reduces server interactions and network overhead while safeguarding the user experience.

A Local-First Software implementation would also enable providers to optimize server usage. For example, servers could process expense submissions during predefined synchronization windows, such as weekly or monthly. This would allow service providers to scale server resources according to demand, reducing energy consumption and operational costs. Splitting server functionality into two roles —one for serving static data and another for processing submissions— would further streamline operations. A static data server could remain available at all times, while the processing server could activate only during synchronization periods, for instance, one day a week, thus saving up to six days of energy costs per week.

Methodology. For demonstration purpose, we compare the energy consumption induced by the servers of the Cloud implementation and the Local-First Software implementation. In our simulation, expenses management software is hosted in Amazon Web Services (AWS). Our estimate of server consumption is based on the methodology of the open-source project Cloud Carbon Footprint [33].

We compare the cumulative energy consumption of the Cloud implementation with a server (Amazon EC2 M8g xlarge) that has 100% uptime vs. the Local-First Software implementation with a S3 instance serving the website statically and a server (Amazon EC2 M8g 12xlarge) turned on for four hours on Fridays.

Results. Fig. 5 represents the energy consumption of each implementation over a week, with the energy consumption in watt-hours represented on the vertical axis. This chart shows the cumulative energy consumption of the Local-First Software implementation and the Cloud implementation. Based on this simulation, we would expect a 71% reduction in energy

consumption when implementing this service following the Local-First Software approach. This illustrates the substantial reduction in the energy footprint of the expense management service if implemented according to the Local-First Software approach.

Discussion. The benefits of a Local-First Software design for expense management software go beyond usability improvements and cost savings. It represents a paradigm shift toward more sustainable and efficient software architecture. The Local-First Software implementation offers more flexibility for synchronization, allowing service providers to take into account new information, such as the electrical grid load or the energy mix [19].

While this approach may not suit all use cases, it offers a compelling alternative for applications with periodic workflows, low collaboration needs, and static data dependencies, setting a precedent for how business software can balance functionality with sustainability.

C. Catalog Search Engine

Catalog search engines typically rely on centralized servers to deliver data to users. While these platforms efficiently support data exploration and retrieval, their Cloud-centric architecture presents significant opportunities for energy optimization through Local-First Software principles. This potential is particularly pronounced because these systems exhibit two key characteristics: relatively static data and predictable usage patterns.

Consider a training management platform that includes a course catalog: most of its data (course descriptions, schedules, instructor details) changes infrequently, often following specific academic cycles. This stability makes aggressive caching not just possible but highly beneficial for energy efficiency. Instead of repeatedly requesting unchanged data from servers, clients could maintain local copies and synchronize only when updates occur, dramatically reducing network traffic and server load.

Methodology. To quantify the benefits of Local-First Software in such contexts, we developed a visualization tool that models the trade-off between data freshness and network efficiency. This tool relies on a set of parameters defined in Table II and compares two implementations: a traditional Cloud-based approach requiring regular server requests, and a Local-First Software approach combining initial data caching with periodic synchronization.

Our tool generates comparative plots (Fig. 6 & Fig. 7) showing cumulative data transfer over time. The implementations are modeled using recursive functions. $Cloud$ and $LoFi$ represents the cumulated bytes transferred by the Cloud implementation, and Local-First Software implementation, respectively.

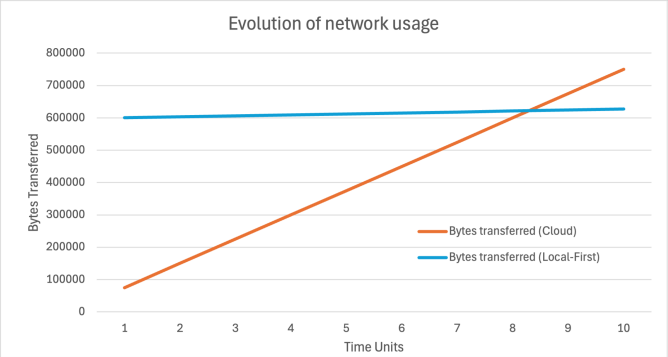


Fig. 6. Simulation of the data transferred over time for Scenario 1. Parameters: $W_r = 75000, F_r = 1, W_d = 600000, W_s = 3000, F_s = 1$

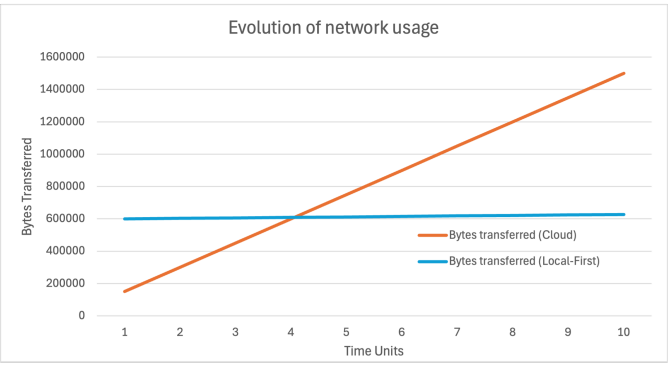


Fig. 7. Simulation of the data transferred over time for Scenario 2. Parameters: $W_r = 75000, F_r = 2, W_d = 600000, W_s = 3000, F_s = 1$

$$Cloud(1) = W_r * F_r$$
$$Cloud(t) = Cloud(t-1) + W_r * F_r,$$

$$LoFi(1) = W_d$$
$$LoFi(t) = LoFi(t-1) + W_s * F_s,$$

These functions enable direct comparison of bandwidth consumption patterns between implementations. To ground our analysis in real-world conditions, we derived our parameters from the data of an actual training management website used by French universities. Here W_r represents the typical size of JSON responses for page loads, and W_d represents a relevant subset of data for a user.

We examine two representative usage scenarios that demonstrate how user behavior affects the efficiency trade-offs:

- **Scenario 1**: Single page visits per time unit ($F_r = 1$), representing a user who checks for updates on their first page.
- **Scenario 2**: Double page visits per time unit ($F_r = 2$), simulating a more active user who explores the database.

Discussion. For our scenarios, Local-First Software implementations become more network-efficient over time when compared to Cloud-based solutions. This advantage stems

TABLE II
PARAMETER DEFINITIONS FOR DATA TRANSFER SIMULATOR

	Parameter	Definition
Cloud	Weight of the request (W_r)	The total data transferred by the request and the response
	Frequency of request (F_r)	The amount of times this request is sent for each time unit
Local-First Software	Weight of the dataset (W_d)	The size of the dataset to be downloaded and cached for the Local First implementation
	Weight of synchronization (W_s)	The total data transferred for the synchronization
	Frequency of synchronization (F_s)	The amount of times the dataset is synchronized for each time unit

from the trade-off between the initial dataset download and subsequent lightweight synchronizations versus repeated full-page downloads. The break-even point—where Local-First Software becomes more efficient—varies with user behavior: as shown in our scenarios, doubling the page request frequency (F_r) significantly reduces the time needed for Local-First Software to prove advantageous over the Cloud-based approach. However, if the synchronization frequency (F_s) and weight of synchronization (W_s) become higher than the weight of the request (W_r) and frequency of request (F_r), Local-First Software can have a negative impact.

Our analytical tool, parameterized with real-world statistics, enables service designers to make informed decisions about caching and synchronization strategies based on their specific use cases. For instance, designers can determine optimal cache or local database sizes (W_d) and synchronization intervals (F_s) by modeling their expected user behavior patterns and data update frequencies.

The results suggest that Local-First Software approaches can deliver multiple benefits in catalog search engine: reduced network traffic, lower operational costs, and improved user experience in low-connectivity scenarios. Additional energy savings can be achieved through complementary strategies, such as scheduling server downtime during predictable low-usage periods, with clients operating on cached data until the next synchronization window.

These findings have broader implications for database-driven applications across the IT industry. The principles demonstrated here—strategic caching, scheduled synchronization, and adaptation to usage patterns—could benefit various enterprise systems, from CRM platforms to inventory management tools, potentially leading to significant industry-wide energy efficiency improvements.

D. Discussion on the Impacts

In Section IV, we introduced a set of energy-related impacts. Fig. 8 illustrates these impacts in relation to our three real-world examples. Each impact is denoted by a colored star, indicating its relevance to the corresponding real-world example. In the remainder of this section, we summarize the impacts that Local-First Software has on these examples.

Collaborative Text Editor. As this service is highly collaborative and has strong real-time requirements, the server part of this service is affected in two ways: reduction of the fault tolerance and resilience infrastructure, and merge overhead. Indeed, as the client-side software can still edit and view downloaded documents, if a failure happens on the provider-side, the application is partition tolerant; thus, the user is still able to consult and edit its documents. Although, leveraging the Local-First Software approach, overall network usage can be reduced by sending batch updates when needed (e.g., a new viewer opens the document). In addition, the client can perform synchronization when connected to a less energy-consuming network, such as Wi-Fi.

Expenses Management Software. In our example, multiple energy-related impacts are identified when implementing a Local-First Software architecture. This service relies on mostly static data and does not require real-time collaboration; this allows multiple energy reduction actions, such as reducing the servers' uptime, synchronizing batch updates in off-peak hours, and consolidating the server with other services. By carefully choosing the computation hours, the service provider can reduce its electricity bill. However, in this example, if the software has a computing-heavy function such as Optical Character Recognition (OCR) for reading the user's expenses, this will impact the battery of the device.

Catalog Search Engine. This example is affected in the same way as the Expenses Management Software example. In addition, in this example, this approach has a negative impact on the end user's device, as the device will need to download all the relevant data on the first connection.

These real-world examples illustrate the complexity of the trade-offs designers will have to make to follow the Local-First Software approach. As the Local-First Software approach advocates for moving the computational load back ("Onloading") to the end-user devices, the designers will have to think in a resource-constrained framework. User- and data-centric designs must be taken into account in order to achieve this shift. This transition will require educating end-users on their impact [34] and designers [35] about their implementation possibilities [36].

VI. RELATED WORK

The environmental impact of ICT services has become a critical concern, prompting research across multiple levels of the technology stack [37]–[40]. Efforts to reduce ICT energy consumption span various domains, including hardware optimization, infrastructure management, software architecture, and user behavior analysis. Our work primarily focuses on the intersection of infrastructure optimization and software

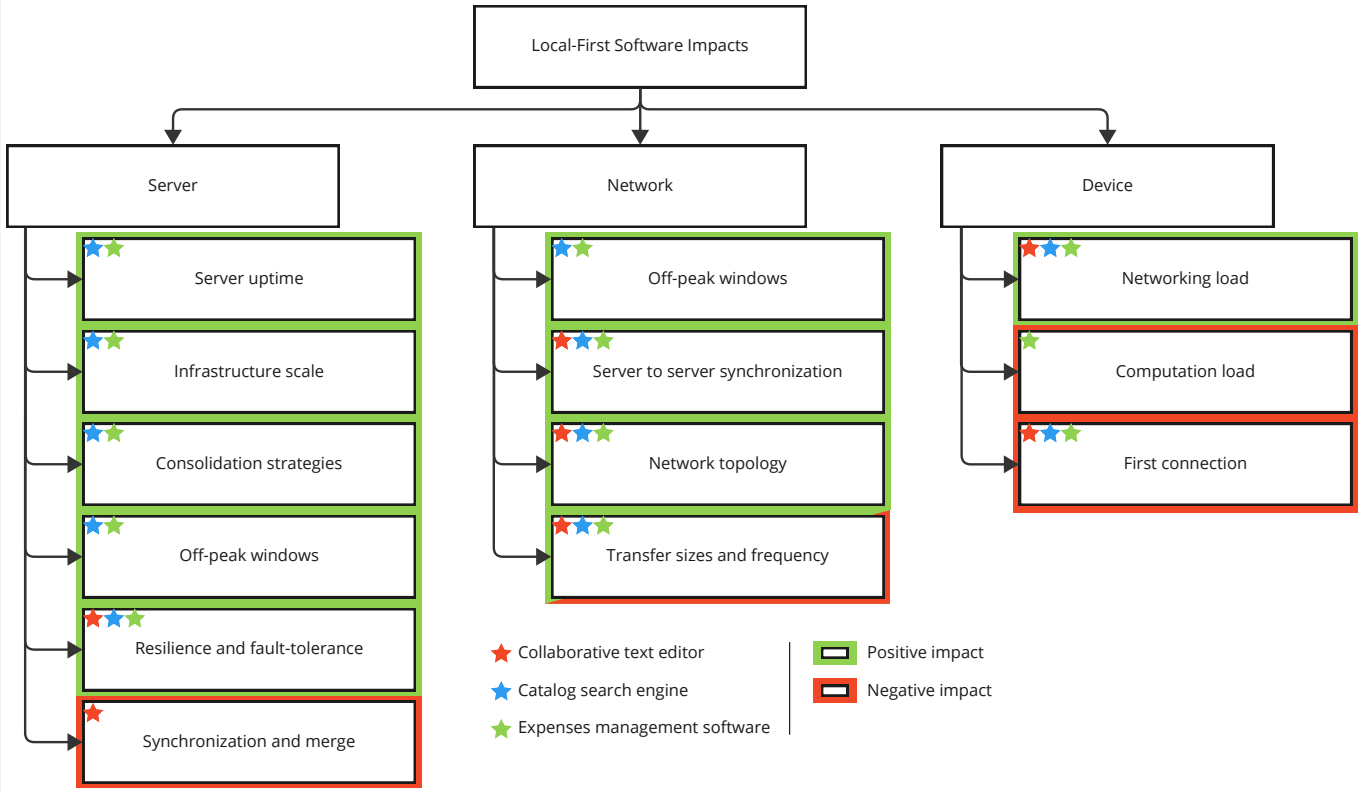


Fig. 8. Local-First Software Impacts Map.

architecture approaches, particularly through the lens of Local-First Software.

Energy Consumption in ICT Services. At the infrastructure level, research has focused on optimizing data transmission through improved routing [41], strategic timing of data transfers [22], and efficient resource allocation [42]. These studies demonstrate that relatively simple changes, such as scheduling data transfers during off-peak periods or optimizing data routes, can meaningfully reduce energy consumption.

The architectural approach has evolved from centralized Cloud computing towards Edge computing, aiming to reduce data transfer distances and processing overhead. Although Edge computing brings computation closer to end-users by deploying servers at network edges [43], it still maintains fundamental dependencies of the server. Some studies point out that the choice of offloading is not obvious: for instance, Namboodiri and Ghose [16] challenge assumptions about the energy efficiency of computational offloading for mobile devices. In addition, in [44], Ahvar et al. conclude that a fully distributed architecture consumes less energy than fully centralized and partly distributed architectures.

Software-level optimization studies have explored various strategies for reducing energy consumption through code efficiency and architectural choices [45]–[49]. However, these approaches typically focus on optimizing within existing paradigms rather than questioning the fundamental architecture of digital services.

Local-First Software Approaches. The Local-First Software paradigm, introduced by Kleppmann et al. [2], emerged primarily as a solution for data sovereignty and improved user experience. This approach emphasizes local data storage and processing. Several implementations have demonstrated its viability, particularly in collaborative editing scenarios [30] and text-based applications [29].

Although Local-First Software's primary focus was not energy efficiency, few studies have explored its potential environmental benefits. Initial investigations comparing offline versus online modes of collaborative tools show promising results [17]. Some research has specifically examined energy consumption implications [50], [51], but these studies typically focus on client-side impacts rather than considering the entire service ecosystem.

Our work bridges these domains by examining how Local-First Software principles can be leveraged specifically for saving energy. Unlike previous studies that either optimize existing architectures or implement Local-First Software for other benefits, we provide a comprehensive analysis of how this architectural shift affects energy consumption across the entire service stack. This approach represents a fundamental rethinking of digital service architecture with energy saving as a primary consideration.

VII. Conclusion and Future Work

This paper demonstrates how a shift towards Local-First Software architecture can significantly impact the energy foot-

print of ICT applications. Through the analysis of real-world case studies, we identified thirteen distinct impacts across the service stack. The most significant positive impacts include reducing the backup infrastructure and shutting down servers, while key challenges emerge from synchronization, device battery, and computing power.

Our findings suggest that Local-First Software architectures offer a promising approach to reducing ICT energy consumption, particularly as end-device capabilities continue to improve. However, this architectural shift requires careful consideration of trade-offs, especially regarding highly collaborative and real-time applications.

Future work will focus on two main directions. First, we will develop Local-First Software Patterns to provide practical guidance for engineers implementing energy-efficient applications. These patterns will help identify which functions are best suited for local processing ("onloading") based on our impact analysis framework. Second, we plan to conduct in-depth case studies by transforming existing Cloud-based applications (such as collaborative editors or enterprise applications) into Local-First Software versions. This transformation will allow us to precisely measure and analyze the energy impacts across the entire stack. We are seeking academic collaborations to explore these transformations, particularly with researchers working at the intersection of software engineering, distributed systems, and sustainable computing, as this interdisciplinary approach will provide comprehensive insights into the challenges and opportunities of Local-First Software architectures.

Acknowledgments

This work was funded by Kapela and the ANRT (Association Nationale de la Recherche et de la Technologie) through the CIFRE (Conventions Industrielles de Formation par la Recherche) program in partnership with Kapela. AI tools such as ChatGPT were used for grammar and formatting refinement. The final content remains the sole responsibility of the authors.

References

[1] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair, and A. Friday, "The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations," *Patterns*, vol. 2, no. 9, p. 100340, Sep. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389921001884

[2] M. Kleppmann, A. Wiggins, P. van Hardenberg, and M. McGranaghan, "Local-first software: you own your data, in spite of the cloud," in *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Athens Greece: ACM, Oct. 2019, p. 154–178. [Online]. Available: https://dl.acm.org/doi/10.1145/3359591.3359737

[3] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2012.

[4] G. T. Bloomberg, Matt Day and N. D. /, "Thousands of amazon workers listen to alexa users' conversations," Apr. 2019. [Online]. Available: https://time.com/5568815/amazon-workers-listen-to-alexa/

[5] N. Confessore, "Cambridge analytica and facebook: The scandal and the fallout so far," *The New York Times*, Apr. 2018. [Online]. Available: https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html

[6] K. Hill, ""god view": Uber allegedly stalked users for party-goers' viewing pleasure (updated)." [Online]. Available: https://www.forbes.com/sites/kashmirhill/2014/10/03/god-view-uber-allegedly-stalked-users-for-party-goers-viewing-pleasure/

[7] May 2019. [Online]. Available: https://www.vice.com/en/article/snapchat-employees-abused-data-access-spy-on-users-snaplion/

[8] *Reuters*, Aug. 2023. [Online]. Available: https://www.reuters.com/business/autos-transportation/tesla-says-two-ex-employees-behind-may-data-breach-2023-08-21/

[9] Jan. 2025, page Version ID: 1268241230. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List_of_data_breaches&oldid=1268241230

[10] P. Bailis, A. Davidson, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "Highly available transactions: Virtues and limitations (extended version)," no. arXiv:1302.0309, Oct. 2013, arXiv:1302.0309 [cs]. [Online]. Available: http://arxiv.org/abs/1302.0309

[11] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, p. 51–59, Jun. 2002. [Online]. Available: https://doi.org/10.1145/564585.564601

[12] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Computer Communications*, vol. 151, p. 556–580, Feb. 2020.

[13] J. Malmodin, N. Lövehagen, P. Bergmark, and D. Lundén, "Ict sector electricity consumption and greenhouse gas emissions – 2020 outcome," *Telecommunications Policy*, vol. 48, no. 3, p. 102701, Apr. 2024.

[14] H. Mazouzi, N. Achir, and K. Boussetta, "Dm2-ecop: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 24:1–24:24, 2019.

[15] N. Muslim, S. Islam, and J.-C. Grégoire, "Offloading framework for computation service in the edge cloud and core cloud: A case study for face recognition," *International Journal of Network Management*, vol. 31, no. 4, p. e2146, 2021.

[16] V. Namboodiri and T. Ghose, "To cloud or not to cloud: A mobile device perspective on energy consumption of applications." San Francisco, CA, USA: IEEE, Jun. 2012, p. 1–9. [Online]. Available: http://ieeexplore.ieee.org/document/6263712/

[17] A. Vishwanath, F. Jalali, K. Hinton, T. Alpcan, R. W. A. Ayre, and R. S. Tucker, "Energy consumption comparison of interactive cloud-based and local applications," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, p. 616–626, Apr. 2015.

[18] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, Apr. 2010, p. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/5470908

[19] A. Radovanović, R. Koningstein, I. Schneider, B. Chen, A. Duarte, B. Roy, D. Xiao, M. Haridasan, P. Hung, N. Care, S. Talukdar, E. Mullen, K. Smith, M. Cottman, and W. Cirne, "Carbon-aware computing for datacenters," *IEEE Transactions on Power Systems*, vol. 38, no. 2, p. 1270–1280, Mar. 2023.

[20] Y. Wu, M. Tornatore, C. U. Martel, and B. Mukherjee, "Content fragmentation: A redundancy scheme to save energy in cloud networks," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 1186–1196, 2018.

[21] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Stabilization, Safety, and Security of Distributed Systems*, X. Défago, F. Petit, and V. Villain, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 386–400.

[22] M. Ficher, F. Berthoud, A.-L. Ligozat, P. Sigonneau, M. Wisslé, and B. Tebbani, "Assessing the carbon footprint of the data transmission on a backbone network," in *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Mar. 2021, p. 105–109. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9385551

[23] C. Sengul, M. Bakht, A. F. Harris, T. Abdelzaher, and R. Kravets, "Improving energy conservation using bulk transmission over high-power radios in sensor networks," in *2008 The 28th International Conference on Distributed Computing Systems*. Beijing, China: IEEE, Jun. 2008, p. 801–808. [Online]. Available: http://ieeexplore.ieee.org/document/4595956/

[24] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app? fine grained energy accounting on smartphones

with eprof," in *Proceedings of the 7th ACM european conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 29–42. [Online]. Available: https://doi.org/10.1145/2168836.2168841

[25] R. Friedman, A. Kogan, and Y. Krivolapov, "On power and throughput tradeoffs of wifi and bluetooth in smartphones," in *2011 Proceedings IEEE INFOCOM*, Apr. 2011, p. 900–908. [Online]. Available: https://ieeexplore.ieee.org/document/5935315

[26] E. J. Vergara, S. Nadjm-Tehrani, and M. Prihodko, "Energybox: Disclosing the wireless transmission energy cost for mobile devices," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 2, p. 118–135, Jun. 2014.

[27] K. Naik, "A survey of software based energy saving methodologies for handheld wireless communication devices," 2010.

[28] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," 2010.

[29] G. Litt, S. Lim, M. Kleppmann, and P. Van Hardenberg, "Peritext: A crdt for collaborative rich text editing," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 1–36, 2022.

[30] M. Kleppmann and A. R. Beresford, "Automerge: Real-time data sync between edge devices," in *1st UK Mobile, Wearable and Ubiquitous Systems Research Symposium (MobiUK 2018). https://mobiuk.org/abstract/S4-P5-Kleppmann-Automerge. pdf*. sn, 2018, pp. 101–105.

[31] H.-G. Roh, M. Jeon, J.-S. Kim, and J. Lee, "Replicated abstract data types: Building blocks for collaborative applications," *Journal of Parallel and Distributed Computing*, vol. 71, no. 3, pp. 354–368, 2011.

[32] M. Hertzum and K. Hornbæk, "Frustration: Still a common user experience," *ACM Trans. Comput.-Hum. Interact.*, vol. 30, no. 3, Jun. 2023. [Online]. Available: https://doi.org/10.1145/3582432

[33] [Online]. Available: https://www.cloudcarbonfootprint.org/

[34] C. Zhang, A. Hindle, and D. M. German, "The impact of user choice on energy consumption," *IEEE Software*, vol. 31, no. 3, p. 69–75, May 2014.

[35] S. Murugesan, "Harnessing green it: Principles and practices," *IT Professional*, vol. 10, no. 1, pp. 24–33, 2008.

[36] C. Sahin, F. Cayci, I. L. M. Gutiérrez, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh, "Initial explorations on design pattern energy usage," in *2012 First International Workshop on Green and Sustainable Software (GREENS)*, 2012, pp. 55–61.

[37] E. Gelenbe, "Electricity consumption by ict: Facts, trends, and measurements," *Ubiquity*, vol. 2023, no. August, pp. 1:1–1:15, 2023.

[38] S. Arora and A. Bala, "A survey: Ict enabled energy efficiency techniques for big data applications," *Cluster Computing*, vol. 23, no. 2, p. 775–796, Jun. 2020.

[39] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: Survey on energy efficiency," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 33:1–33:36, 2014.

[40] L. M. Hilty, V. Coroama, M. O. De Eicker, T. Ruddy, and E. Müller, "The role of ict in energy consumption and energy efficiency," *Report to the European Commission, DG INFSO, Project ICT ENSURE: European ICT Sustainability Research, Graz University*, vol. 1, pp. 1–60, 2009.

[41] P. Loygue, K. Al Agha, and G. Pujolle, "Carbon footprint of cloud, edge, and internet of edges," *Annals of Telecommunications*, vol. 80, no. 1–2, p. 153–169, Feb. 2025.

[42] Z. Li, S. Tesfatsion, S. Bastani, A. Ali-Eldin, E. Elmroth, M. Kihl, and R. Ranjan, "A survey on modeling energy consumption of cloud applications: Deconstruction, state of the art, and trade-off debates," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 3, p. 255–274, Jul. 2017.

[43] B. Gill and D. Smith, "The edge completes the cloud: A gartner trend insight report."

[44] E. Ahvar, A.-C. Orgerie, and A. Lebre, "Estimating energy consumption of cloud, fog, and edge computing infrastructures," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, p. 277–288, Apr. 2022.

[45] E. Jagroep, J. M. van der Werf, S. Brinkkemper, L. Blom, and R. van Vliet, "Extending software architecture views with an energy consumption perspective," *Computing*, vol. 99, no. 6, p. 553–573, Jun. 2017.

[46] S. Maleki, C. Fu, A. Banotra, and Z. Zong, "Understanding the impact of object oriented programming and design patterns on energy efficiency," in *2017 Eighth International Green and Sustainable Computing Conference (IGSC)*, Oct. 2017, p. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8323605

[47] J. Mancebo, F. García, and C. Calero, "A process for analysing the energy efficiency of software," *Information and Software Technology*, vol. 134, p. 106560, Jun. 2021.

[48] G. Procaccianti, P. Lago, and S. Bevini, "A systematic literature review on energy efficiency in cloud software architectures," *Sustainable Computing: Informatics and Systems*, vol. 7, p. 2–10, Sep. 2015.

[49] C. Stier, A. Koziolek, H. Groenda, and R. Reussner, "Model-based energy efficiency analysis of software architectures," in *Software Architecture*, D. Weyns, R. Mirandola, and I. Crnkovic, Eds. Cham: Springer International Publishing, 2015, p. 221–238.

[50] Q. Stokkink and J. Pouwelse, "A local-first approach for green smart contracts," *Distributed Ledger Technologies: Research and Practice*, vol. 3, no. 2, p. 1–21, Jun. 2024.

[51] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, p. 1517–1530, Jan. 2022.