Sustainable Computing on the Edge: A System Dynamics Perspective

Brian Ramprasad
brianr@cs.toronto.edu
University of Toronto
Toronto, Ontario, Canada

Alexandre da Silva Veith
aveith@cs.toronto.edu
University of Toronto
Toronto, Ontario, Canada

Moshe Gabel
mgabel@cs.toronto.edu
University of Toronto
Toronto, Ontario, Canada

Eyal de Lara
delara@cs.toronto.edu
University of Toronto
Toronto, Ontario, Canada

ABSTRACT

This paper explores the CO_2 footprint of IoT applications by using system dynamics modeling to estimate the CO2 emissions over time from a wireless video analytics application. We model the impact of the application design and the mobile infrastructure on the short and long term emissions produced by running the application on both cloud and edge computing infrastructures. Our analysis shows that the base station radio and the wide-area data network are major contributors of CO_2 emissions. We find that CO_2 emissions can be reduced by 50% by placing edge centers near the base stations, exploiting new features of the 5G mobile network, and scheduling data uploads judiciously. We also analyze the long term effects of application design choices and increased user base on carbon emissions.

CCS CONCEPTS

• **Hardware → Impact on the environment**; • **Networks →** *Mobile networks.*

KEYWORDS

sustainability, mobile computing, edge computing

ACM Reference Format:
Brian Ramprasad, Alexandre da Silva Veith, Moshe Gabel, and Eyal de Lara. 2021. Sustainable Computing on the Edge: A System Dynamics Perspective. In *The 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile 2021), February 24–26, 2021, Virtual, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3446382.3448607

1 INTRODUCTION

One of the main drivers of climate change are CO_2 emissions from electricity production due to the burning of fossil fuels, such as coal, oil, and natural gas. The rapid development of information technology, with the seemingly daily introduction of new devices has the potential to exacerbate the problem. For example, continued adoption of the Internet of Things (IoT), mobile devices, and sensor networks is predicted to lead to 41.6 billion connected devices by 2025, which are expected to transmit over 79.4 zettabytes of data via the Internet [9, 19] placing additional demands on the electric power beyond the 70 billion kilowatt-hours a year that are already required to maintain the Internet [10] today.

In this paper, we use system dynamics modeling [11] to explore the CO_2 emissions of IoT applications. As our use case, we consider a cloud-based application that performs analytics on video streams captured by a network of video cameras that use the cellular network to connect to the cloud. Our model considers the contributions of the wireless network, the wide-area network, and the cloud servers to the application's overall CO_2 footprint.

We perform two types of analysis in this paper. First, a point analysis that explores the relative contributions of compute and networking to the application's CO_2 footprint. This analysis shows that the main contributor to the CO_2 footprint is energy consumed by the cellular base-stations and data transmission over a wide-area data network (e.g., the Internet). We show that the former can be significantly reduced for 5G networks by implementing a transmission schedule for video uploads which allows base stations to sleep intermittently. For the latter, the use of edge computing [18], where additional computation resources are positioned close to the edge of the network (one hop away), allows for local processing that significantly reduces the amount of data that needs to be sent to cloud servers over the Internet. Second, a long term analysis that explores the evolution of the overall application's CO_2 footprint over a period of several years. Our analysis considers the effect of increased demand due to growth in the user base, improvement in both the cloud and edge infrastructure, and the developers efforts to reduce the energy demands of the application.

The rest of this paper is structured as follows. Section 2 discusses related work on approaches to modeling the CO_2 footprint of cloud based applications. Section 3 presents the main concepts of system dynamics modeling and the CO_2 model we use in our evaluation. Section 4 presents our results. Lastly, Section 5 discusses limitations and avenues for future work.

2 BACKGROUND AND RELATED WORK

Current approaches toward measuring the ecological impact of applications lack the ability to model the long term evolution and

HotMobile 2021, February 24–26, 2021, Virtual, United Kingdom
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8323-3/21/02…$15.00
https://doi.org/10.1145/3446382.3448607

dynamic nature of IoT deployments. Preist et al. [15] use the Life Cycle Assessment (LCA) method to measure the estimated global carbon footprint of YouTube for the 2016 year. The LCA method captures the current state of the system using past information to calculate the carbon footprint. The interventions proposed by the authors suggest that users change their behavior to reduce the carbon footprint of the YouTube service. The authors noted several limitations with the LCA approach. First, they did not consider the impact of provisioning policies that could lead to an actual change in the carbon footprint rather than requiring users to change their behavior. Second, the LCA model is not dynamic. Therefore, their approach cannot capture the rebound effect (i.e., Jevon's paradox [2]) of more resources being demanded as they become cheaper, leading to more resources being deployed over long time horizons (many years).

Another approach to measuring the ecological impact of applications can be to use System Dynamics (SD) modeling. SD models can be particularly useful because they can operate at a high level of abstraction which helps to understand the behaviour of a collection of systems as one larger system even if the sub-systems have different units of measurement. For example, SD models give us the ability to convert a unit of data (e.g., GB) to a unit of CO_2 which allows us to measure system wide impacts that have cumulative effects such as CO_2 buildup. SD models can be expanded to provide lower levels of abstraction once more fine grained casual relationships between components are understood and have been used to build large models across a variety of subject areas such as wireless spectrum allocation [22], environmental impact assessment [20] and capacity planning [14].

SD models are used to understand trends in system behaviours over long periods of time such as climate change models, which are usually done over several years or decades as trends are of more interest rather than a point in time analysis. Our work proposes to use system dynamics modeling to capture the long term and short term CO_2 impact of cloud computing infrastructures and additionally the impact of the WAN and RAN which is vital towards understanding the end-to-end CO_2 footprint of an application.

3 A SYSTEM DYNAMICS PERSPECTIVE

IoT applications have many components that span from the client-side to the server-side. The components of the application are spread across physical infrastructure and they could be placed in the cloud, the edge, and within the RAN [23]. Typically, the client-side components would reside on a user device and that component will upload data which may be sent over a cellular network and be processed by components running on servers hosted in the cloud. The cellular network and the cloud provider are stake holders in the system that have impact on the CO_2 but are out control of the developer. The developers of the application must understand the stakeholders impact on their applications. For example, is the cloud provider committed to reducing emissions by giving developers the option to lease servers that are powered by renewable energy sources. System dynamics (SD) models are useful for understanding the causal dependencies between the components of the system to revel the impact of one stake holder on another so that better decisions can be made [11].

3.1 The Dynamics of an IoT Application

To explore the dynamics of an IoT application and to be able to quantify how different factors impact the CO_2 footprint of an application, we consider a video analytics application that is deployed at the edge of the RAN network as depicted in Figure 1. We consider two scenarios: First, we model a system where the images from the cameras are sent directly to the cloud for processing. Second, we consider the scenario where images are sent to edge servers that are placed near the base station. Edge computing promises to allow for faster decisions and better privacy by keeping generated data close to where it will be processed [12]. Using the video analytics application and set of deployment scenarios that consider the placement of compute resources, we can use SD models to understand the end-to-end CO_2 footprint of an application.

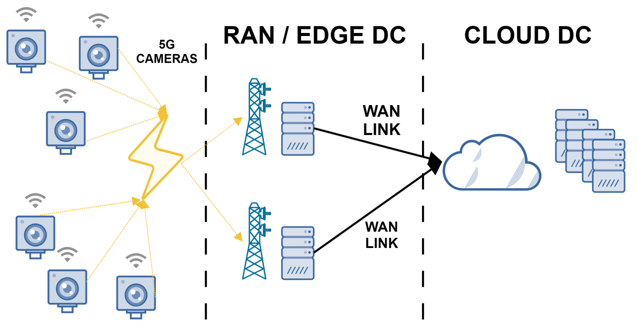

Figure 1: Wireless video application deployment.

3.2 Model Structure

SD models consist of several components that allow for the continuous flow of information over a specified time period. The most basic concept in a SD model is the notion of time. The flow is measured as the difference between the previous and current time window. For example in a simulation that lasts 10 days, the model allows us to see on the 10th day the cumulative effects of running the model for the past 9 days. The components used in our CO_2 model as depicted in Figure 2 are variables, stocks, and flows. *Variables* can be either constants or equations. *Stocks* represent a bucket that can be filled and drained, i.e., a quantity of something. The variables and stocks are connected via black lines and arrows which also denote the direction of the information flow. *Black lines* are different from the arrows as they have valves, depicted by the double triangle symbol. Valves can be instrumented with functions that determine when they are closed and by how much they can be opened, much like a pipe that carries water. When the valve is closed the flow accumulates in the pipe and back pressures the whole system. SD models should mirror real systems meaning that they should not have feedback loops that are unbounded.

Implementation

Our model is created in Vensim[1], a popular simulation tool for

[1]https://vensim.com/

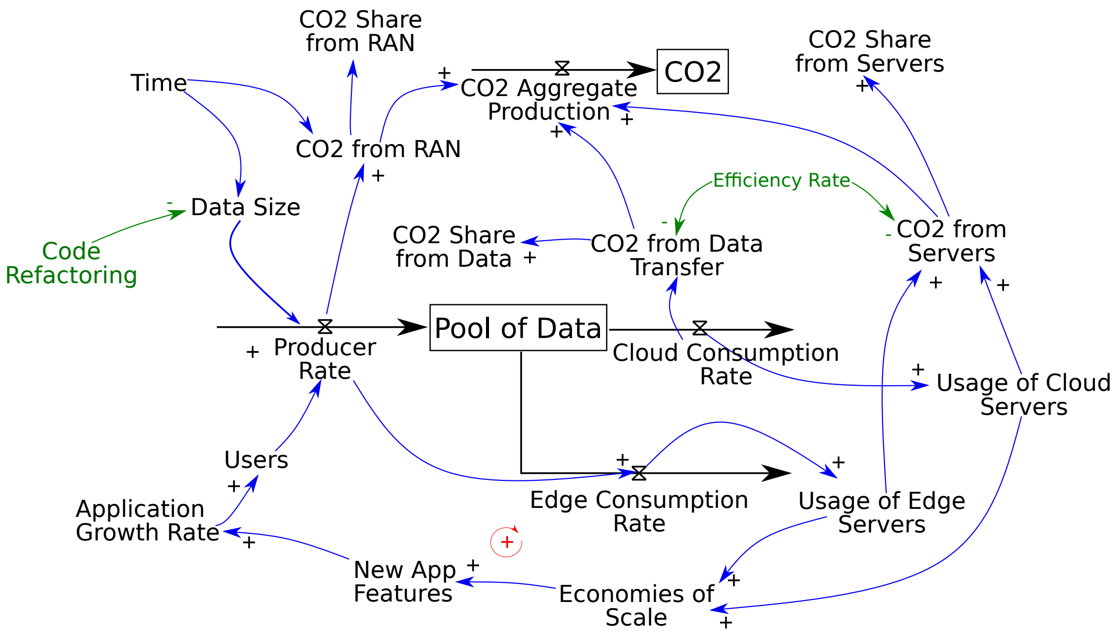

Figure 2: Our system dynamics CO_2 emissions model.

modeling dynamic systems. Figure 2 presents an emissions model, that is used to estimate the CO_2 footprint of a typical IoT application and the resulting impact of a developer's decisions. The parameters in the model can be adjusted to accommodate different types of applications. For example, video based applications are more data intensive as compared to text based sensor data. The application payload size can be set inside the *Data Size* variable. The model also considers the actions taken by the cloud service provider to improve the energy efficiency of its infrastructure and also captures the rebound effects of compute resource consumption.

CO_2 Contributing Factors

The stream of data flowing through the model is the primary driver of CO_2 emissions in the system. Emission of CO_2 occurs from three sources, *CO_2 Share from RAN*, *CO_2 Share from Data*, and *CO_2 Share from Servers*, which correspond to the CO_2 associated with data transmission over the cellular network and the wide-area wired network, as well as server processing, respectively. All of the emissions are then aggregated in the *CO_2 Aggregate Production* and collected in the *CO_2* stock. The rate of the CO_2 emissions is impacted by the *Producer Rate* which is the volume of data going into the stock variable *Pool of Data* (measured in gigabytes). The *Producer Rate* is impacted by the number of *Users* that produce a number of data units and the *Data Size* which determines the size of each data unit. The data is collected in the *Pool of Data* which delivers the data to the servers via the *Cloud consumption Rate* and the *Edge Consumption Rate* depending on whether the application needs to send the data to the cloud or the edge. When the data is sent to the cloud, the WAN must be used so the *Cloud Consumption Rate* will send a copy of the data through the *CO_2 from Data Transfer* so that the CO_2 impact can be captured. Once the data has arrived at the *Cloud Consumption Rate* and the *Edge Consumption Rate* point, the data needs to be sent to the servers for processing. The processing capacity of a server is measured as 1 unit of *Usage of Edge or Cloud Services* and can handle 10 units of network units per hour. In our simulation 1 network unit is 1 GB which means that a single server can handle 20 512 MB/hour streams or 10 x 1 GB/hour streams.

CO_2 Reducing Factors

As previously discussed, the data flowing through the system drives the CO_2 emissions and reducing the data rate will lower the amount of CO_2 produced by the application. The *Data Size* is influenced by *Code Refactoring* which is a cumulative value that overtime lowers the size of the data units. By "code refactoring", we specifically refer to modifications that aim to reduce the data rate of the application. The developer sets a data rate reduction target (e.g., 5% per year) and fulfills this through a continuous commitment to re-designing/modernising the application to transmit less data and less frequently. Examples of concrete code refactoring could be to implement newly developed data compression techniques, implementing selectivity policies for reducing app data uploads, and avoiding unnecessary heartbeats or continuous connectivity to remote cloud services. If refactoring is not feasible, our model is able to capture the effect of no reduction (0% improvement) while still capturing the other factors in the system that contribute to CO_2 footprint.

Another factor that lowers the CO_2 footprint of the application, are the efforts made by the data center providers. This is represented in the model as the *Efficiency Rate*. This is a target that is set by the cloud service provider where by they are able to provide the same compute resources using less energy or they are able to acquire energy from a renewable power source. The rate is a target similar to the *Code Refactoring* and it reduces the amount of KG of CO_2 per gigabyte of data entering *CO_2 Share from Data* and the per hour KG of CO_2 emitted by *CO_2 from Servers*

CO_2 Feedback Loop

Our model is capable of capturing the increase in the amount of CO_2 driven by the growth in the number of *Users*. The more *Users* there are in the system, the more CO_2 is produced. The user growth cycle is a feedback loop in the model. The relationship between the influencing factors in the feedback loop are as follows: more *Users* in the system leads to an increase in the *Producer Rate*, then the increased data volume leads to an increase in the number of *Usage of Edge or Cloud Services* units provisioned, which leads to the developer being able to negotiate better prices due to the *Economies of Scale*, which lowers the marginal cost of providing the application, which in turn allows the developer to invest in the application, so that more *New App Features* can be developed which then makes the application more attractive and this leads to higher a *Application Growth Rate* which then finally leads to more *Users*. This completes the feedback loop as denoted by the red circle shown in the lower portion of the model in Figure 2.

3.3 Model Assumptions and Limitations

The CO_2 emissions model we propose is intended to simulate the cumulative effects of decisions made by the developer and the other factors in the system that are outside of the control of the developer. The goal of the model is not to provide solutions to application-specific problems, but rather to quantify the CO_2 impact of decisions made by the developer. For example, the *Code Refactoring* effect is a numerical commitment target set by the developer. Likewise the impact of *Economies of Scale* on a developers decision to add new features is not deterministic since developers can just choose to pocket the savings and not improve the application any further. This model makes the assumption that developers want to grow their application feature set and broaden their user base.

4 EVALUATION

In this section, we explore the CO_2 contribution of our example smart camera video analytics application deployed on the cloud. We evaluate several approaches that developers can take towards reducing the CO_2 footprint of the application by leveraging base station power management and edge computing. A second experiment looks at the long term evolution of the applications CO_2 footprint as the application user base grows and the effects of infrastructure efficiency improvement and software restructuring are also considered.

4.1 Experimental Setup

We simulate the camera-based edge computing application described in Section 3.1: wireless cameras send video frames over a 5G cellular network for processing on local edge servers and/or on remote cloud servers. We consider the CO_2 emissions from the RAN, the WAN and the servers in the data center. The CO_2 from the cameras is not included in our model. The CO_2 parameters used in the experiments were obtained from the USA EPA Greenhouse Gas Equivalencies Calculator[8]. Table 1 presents the model parameter values used to compute the carbon footprint. For simplicity, we use the same server hardware for both the edge and cloud. In future work we plan to explore different server types since the cloud servers could be more powerful as compared to the servers deployed on the edge. The server CO_2 parameter used in the model is derived from the yearly kWh consumption for a modern dual processor server provided by Dell, rated at 1760.3 kWh per year[2]. To convert kWh to CO_2 per year we used the EPA calculator which works out to $1760.03 \times 0.707 = 1244$ KG of CO2 per year. Therefore the server will have a carbon footprint of $1244/(365 \times 24) = 0.142$ KG CO_2 per hour. We obtained the data rates using the Closed Circuit Television (CCTV) Calculator web tool[3]. A camera with a 1080p resolution has a data rate of 1.5 GB per hour and a 4K camera has a data rate of 7.2 GB per hour. The energy consumption for a 5G BS is 11 kWh [1] and transmitting 1 GB over the WAN will consumer 0.06 kWh[4]. By using the EPA calculator, this amount of energy is equivalent to 7.80 KG and 0.042 KG of CO_2 per hour respectively.

Table 1: Model parameter values

Description	Value
GB on WAN network to CO_2	0.042 KG per GB [8]
Server CO_2	0.142 KG/hour [5]
5G RAN to CO_2	7.8 KG/hour [1]

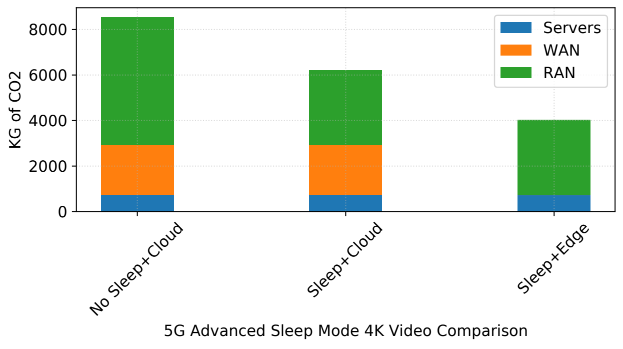

Figure 3: CO_2 emissions of the video application using a 5G network over a 30 day period.

4.2 The Contributors to an Application's CO_2 Footprint

This experiment considers 3 different deployment scenarios over 30 days using 5G radio access networks as depicted in Figure 3. The left bar (No Sleep+Cloud) represents the CO_2 footprint of our video analytics application when the data is being uploaded from the smart cameras to the cloud and the BS is not configured to sleep. The results of this experiment show that the biggest contributor to the CO_2 footprint of the application is the RAN following by the WAN and the Servers.

[2]https://i.dell.com/sites/csdocuments/CorpComm_Docs/en/carbon-footprint-poweredge-r640.pdf
[3]https://www.cctvcalculator.net/en/calculations/bandwidth-calculator/

The 5G NR standard allows for the BS to be put into deep sleep using a tiered mode design. As the RAN is the largest contributing factor to the CO_2 footprint of the application we can see in Figure 3 that comparing the cumulative CO_2 of the left bar (No Sleep+Cloud) and the middle bar (Sleep+Cloud) that our server-side sleep policy has reduced total emissions by approximately 25%. Our server-side sleep policy achieves this by having the BS request the cameras to upload data that is stored in their local buffers. Furthermore, as can been seen in the right bar (Sleep+Edge), by sending the data directly to the edge we can nearly eliminate the CO_2 emissions from the WAN and lower the total CO_2 footprint from 8000 KG to 4000 KG, which is a 50% reduction.

The Impact of Server-side Sleeping

Our server-side sleep policy is designed to address future IoT deployments where always-on devices such as cameras and sensors are continuously uploading data using a 5G connection. Approaches towards optimizing the 5G BS sleep feature from the RAN-side have yielded energy reductions between 80%-90% [16, 21]. However, these approaches consider current RAN usage patterns which mostly consist of transient connections that are downstream transmissions. An application that implements our server-side policy will no longer send continuous streams of data that will unnecessarily keep the BS at full power. Periodic transmission or batching of information has the potential to drastically reduce the energy used by the BS[7]. Our server-side sleep policy is configured to request uploads from the camera as a batch once per minute during off peak hours (12am to 12pm). By requesting the uploads from the server side we can coordinate the upload between devices to minimize the total time that the BS powered on. Our policy potentially helps the BS to go to sleep more often and limits the time that the cameras will spend uploading data and completing any application management tasks. The data upload time and BS wake up time is configured in our model to take 10 seconds. A 5G BS requires 11 kWh of power when it is in an active state and when using our policy, the BS can now sleep for 50 seconds per minute which lowers consumption to 1.87 kWh which represents an 83% reduction in energy usage for always-on type devices which is similar to what others have achieved with sleeping strategies for transient type connections [6, 16, 17, 21]. The impact of our policy is that now the base station is being powered on for only 10 minutes per hour instead of the full hour.

4.3 The Evolution of an Application's CO_2 Footprint

This experiment explores the effects on the application's CO_2 footprint from the growth in the user-base, improvements in the application's code base and the underlying infrastructure. This set of experiments measures the CO_2 footprint of our video analytics application where the cameras upload their frames over a WiFi network and the data is transmitted over the WAN to servers hosted in the cloud. We compare the resulting CO_2 impact for six different scenarios, and consider the consequences of video resolution and code refactoring to lower the data rate of the application.

Table 2 describes the developer decisions that we consider in our simulation. *Resolution* is the resolution of video frames sent by cameras, which affects the data rate. *Refactoring* represents the commitment of the developer to improving their code by reducing the data rate and in this set of experiments we set an annual target for the developer to reduce the volume of data transmitted by 10%. The *4K Switch* factor determines the year in which the developer switches from a 1080p resolution to a 4K resolution, which changes the per camera data rate from 1.5 GB to 7.2 GB per hour respectively. In our experiments, if a switch in resolution needs to occur, it will happen at the beginning of year 4. The timeline of the simulation is over a period of six years.

Table 2: Experimental scenarios and parameter points

Scenario	Resolution	Refactoring	4K Switch
4K-N-0	4K	No	Year 0
4K-Y-0	4K	Yes	Year 0
1080P⟶4K-N-4	1080p ⟶ 4K	No	Year 4
1080P⟶4K-Y-4	1080p ⟶ 4K	Yes	Year 4
1080P-N-N	1080p	No	None
1080P-Y-N	1080p	Yes	None

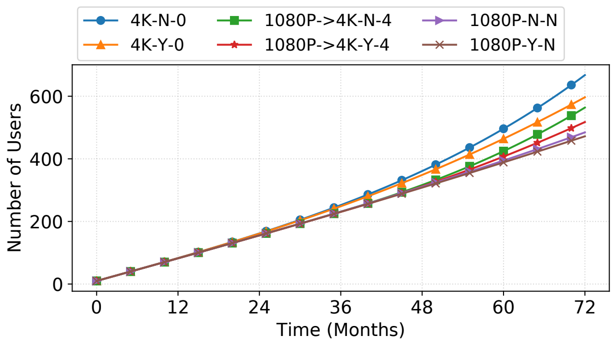

Figure 4: User base growth

The growth in the user base is the primary driver for the amount of data that is put into the system. The rate of growth is influenced by the new app features which in turn is influenced by the economies of scale factor. Figure 4 shows the growth rate for the scenarios we consider. The User base growth rate is very similar until the end of the 3rd year and then they diverge. The reason for the divergence is because the (4K-N-0), (4K-Y-0), (1080P⟶4K-N-4) experiments have a higher data rate due to the use of the 4K cameras which require more servers to process the data, which leads to the developer getting a discount on the servers and in turn allowing the developer to invest more in application features. The scenarios (1080P⟶4K-Y-4), (1080P-N-N), (1080P-Y-N) have much lower data rates and therefore suppress the impact of the economies of scale and new application features on the user base growth.

Figure 5 shows the economies of scale factor as a cumulative discount that grows overtime based on the number and length of time that the servers leased. The discount is cumulative because the cost per compute unit goes down over the years. We can see

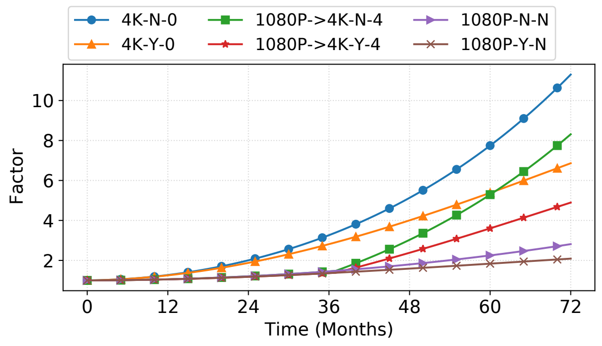

Figure 5: Economies of scale factor

the changes in the trend for experiments (4K-Y-0) and (1080P⟶4K-N-4) which cross at the end of the 5th year and (1080P⟶4K-N-4) continues to move upward because code refactoring was not used from the beginning. This is an example of Jevons paradox because in this case the higher economies of scale means the application is using more compute resources so the discount is higher and a leaves more money for the developer to invest in the application that will use more servers.

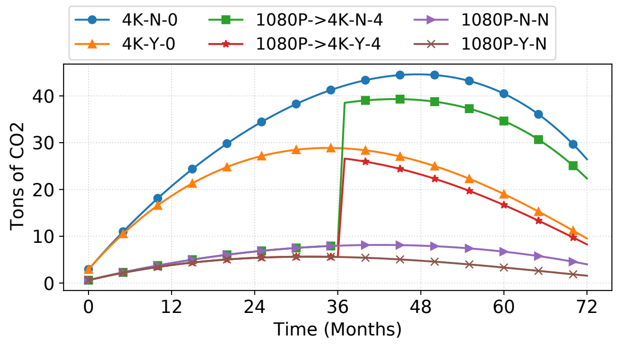

Figure 6: Monthly CO_2 for each year

Figure 6 shows an application's CO_2 footprint on a monthly basis. The figure shows that there is a significant difference between the different scenarios. In the experiment (4K-Y-0) the line eventually decreases because the data center provider efficiency is outpacing the data rate growth of the application. In the (4K-Y-0) experiment the developer has additionally committed to refactoring and we see that the monthly rate is much lower compared to (4K-Y-0) because code refactoring is lowering the data rate over the 6 years. In the (1080P⟶4K-N-4) experiment where the developer has chosen to delay the deployment of the 4K feature until the beginning of year 4 but has neglected to consider refactoring, the monthly CO_2 contribution spikes and is higher in year 4-6 as compared to (4K-Y-0) where the 4K setting was enabled from the beginning, but refactoring was used. In (1080P⟶4K-Y-4) we can see that the spike at the beginning of year 4 is lower because the effects of continuous improvement are cumulative so when the higher resolution setting was introduced at the beginning of year 4 the platform was more efficient due to the code refactoring. In the experiments 1080P-N-N and 1080P-Y-N, no switch to 4K happens and they only differ because of the impact of code refactoring.

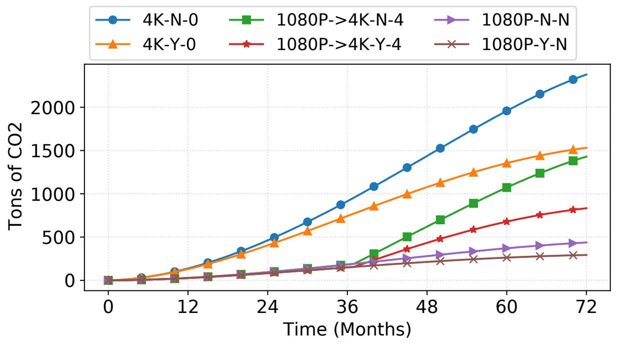

Figure 7: Cumulative CO_2

Considering the overall CO_2 footprint of the application, we can see that in Figure 7 that there are significant differences in the cumulative CO_2 at the end of the 6th year. The least optimized scenario is the (4K-N-0) experiment because it has the highest cumulative CO_2 footprint due to the developers decision to enable the 4K setting and not commit to any refactoring and is instead relying on the service provider to make all of the optimizations. Contrasting this result with the most optimal scenario which is (1080P-Y-N) we can see that the cumulative CO_2 is much lower because the lowest resolution is used and a commitment to refactoring was made by the developer.

5 DISCUSSION

This paper takes a first step into investigating the CO_2 footprint of IoT applications. We consider the use case of a video analytics application deployed both in the cloud and edge. Our evaluation highlights the importance that programmer design decisions have on the overall CO_2 footprint of applications, and the need for continuous improvement to offset increases in CO_2 emissions driven by growth in an application's user base.

In the future, we plan to explore other application archetypes which have different deployment scenarios. For instance, application components can run on multiple machines in the path from the source to the cloud in a hierarchy of data centers [13]. Additional factors would then become a concern such as increased synchronization between components and encryption that can add to the transmission overhead [3]. These factors may have a significant effect on CO_2 production. More work is needed to study scenarios with these types of data access patterns to generate a set of recommendation guidelines for mobile edge computing initiatives that minimize the environmental impact of applications running on the edge.

REFERENCES

[1] 5G Power 2020. 5G Power: Creating a green grid that slashes costs, emissions and energy use. https://www.huawei.com/ca/publications/communicate/89/5g-power-green-grid-slashes-costs-emissions-energy-use

[2] Blake Alcott. 2005. Jevons' paradox. *Ecological Economics* 54, 1 (2005), 9 – 21.

[3] K. Andersson, X. Chen, C. Esposito, and E. Rondeau. 2020. Editorial: IEEE Transactions on Sustainable Computing, Special Issue on Cryptography and Data Security in Sustainable Computing (Part 1). *IEEE Transactions on Sustainable Computing* 5, 02 (jul 2020), 160–160.

[4] Joshua Aslan, Kieren Mayers, Jonathan G. Koomey, and Chris France. 2018. Electricity Intensity of Internet Data Transmission: Untangling the Estimates. *Journal of Industrial Ecology* 22, 4 (2018), 785–798.

[5] Dell Technologies [n.d.]. Product Carbon Footprints: Dell Technologies. https://corporate.delltechnologies.com/en-us/social-impact/advancing-sustainability/sustainable-products-and-services/product-carbon-footprints.htm

[6] A. El-Amine, H. A. Haj Hassan, M. Iturralde, and L. Nuaymi. 2019. Location-Aware Sleep Strategy for Energy-Delay Tradeoffs in 5G with Reinforcement Learning. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 1–6. https://doi.org/10.1109/PIMRC.2019.8904155

[7] P. Frenger and K. W. Helmersson. 2019. Energy Efficient 5G NR Street-Macro Deployment in a Dense Urban Scenario. In *2019 IEEE Global Communications Conference (GLOBECOM)*. 1–6.

[8] Greenhouse Gas Equivalencies Calculator 2018. Greenhouse Gas Equivalencies Calculator. https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator

[9] Andrea Hamm, Alexander Willner, and Ina Schieferdecker. 2019. Edge Computing: A Comprehensive Survey of Current Initiatives and a Roadmap for a Sustainable Edge Computing Development. arXiv:1912.08530 [cs.DC]

[10] Christopher Helman. 2019. Berkeley Lab: It Takes 70 Billion Kilowatt Hours A Year To Run The Internet. https://www.forbes.com/sites/christopherhelman/2016/06/28/how-much-electricity-does-it-take-to-run-the-internet/?sh=7e284e731fff. Online.

[11] Donella H. Meadows. 2008. *Thinking in Systems: A Primer*. Chelsea Green Publishing.

[12] Seyed Hossein Mortazavi, Mohammad Salehe, Moshe Gabel, and Eyal de Lara. 2020. Feather: Hierarchical Querying for the Edge. In *IEEE/ACM Symposium on Edge Computing (SEC)*.

[13] Seyed Hossein Mortazavi, Mohammad Salehe, Carolina Simoes Gomes, Caleb Phillips, and Eyal de Lara. 2017. Cloudpath: A multi-tier cloud computing framework. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–13.

[14] Derek L. Nazareth and Jae Choi. 2017. Capacity Management for Cloud Computing: A System Dynamics Approach. In *AMCIS*.

[15] Chris Preist, Daniel Schien, and Paul Shabajee. 2019. Evaluating Sustainable Interaction Design of Digital Services: The Case of YouTube. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. 1–12.

[16] F. E. Salem, Z. Altman, A. Gati, T. Chahed, and E. Altman. 2018. Reinforcement Learning Approach for Advanced Sleep Modes Management in 5G Networks. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. 1–5. https://doi.org/10.1109/VTCFall.2018.8690555

[17] F. E. Salem, A. Gati, Z. Altman, and T. Chahed. 2017. Advanced Sleep Modes and Their Impact on Flow-Level Performance of 5G Networks. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. 1–7. https://doi.org/10.1109/VTCFall.2017.8288125

[18] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. 2009. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing* 8, 4 (2009), 14–23.

[19] Help Net Security. 2019. Connected IoT Devices. https://www.helpnetsecurity.com/2019/06/21/connected-iot-devices-for. Online.

[20] Michal Sedlacko, Andre Martinuzzi, and Karin Dobernig. 2014. A Systems Thinking View on Cloud Computing and Energy Consumption. In *ICT4S*.

[21] R. Tano, M. Tran, and P. Frenger. 2019. KPI Impact on 5G NR Deep Sleep State Adaption. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 1–5. https://doi.org/10.1109/VTCFall.2019.8891171

[22] Rikin Thakker, S. Sarkani, and T. Mazzuchi. 2012. A system dynamics approach to demand and allocation of wireless spectrum for mobile communication. In *CSER*.

[23] Abhishek Tiwari, Brian Ramprasad, Seyed Hossein Mortazavi, Moshe Gabel, and Eyal de Lara. 2019. Reconfigurable Streaming for the Mobile Edge. In *20th International Workshop on Mobile Computing Systems and Applications (HotMobile)*. Santa Cruz, CA.