Achlys : towards a framework for distributed storage and generic computing applications for wireless IoT edge networks with Lasp on GRiSP

Igor Kopestenski
Université Catholique de Louvain
Dept. of Computer Science and Engineering
igor.kopestenski@uclouvain.be

Peter Van Roy
Université Catholique de Louvain
Dept. of Computer Science and Engineering
peter.vanroy@uclouvain.be

***Abstract*—Internet of Things (IoT) has gained substantial attention over the past years. And the main discussion has been how to process the amount of data that it generates, which has lead to the edge computing paradigm. Wether it is called *fog*[1], *edge or mist*, the principle remains that cloud services must become available closer to clients. This document presents ongoing work on future edge systems that are built to provide steadfast IoT services to users by bringing storage and processing power closer to peripheral parts of networks. Designing such infrastructures is becoming much more challenging as the number of IoT devices keeps growing. Production grade deployments have to meet very high performance requirements, and end-to-end solutions involve significant investments. In this paper, we aim at providing a solution to extend the range of the edge model to the very farthest nodes in the network. Specifically, we focus on providing reliable storage and computation capabilities immediately on wireless IoT sensor nodes. This extended edge model will allow end users to manage their IoT ecosystem without forcibly relying on gateways or Internet provider solutions. The approach taken in this paper is to first introduce Achlys, a prototype implementation of an edge node that is a concrete port of the Lasp programming library on the GRiSP Erlang embedded system. We also attempt to show how Achlys could address the need for a general purpose edge that is both resilient and consistent in terms of storage and network. We conclude with a presentation of example use cases that could take advantage of integrating the Achlys framework and discuss future work.**

I. Introduction

The edge computing paradigm has been widely accepted as a high level concept for sustainability of future Cloud Service Providers (CSPs) and Mobile Network Operators (MNOs) [1], [2]. While being s considered as an emerging solution, it has been well acknowledged by both enterprise and academia as a valid approach and hence is actively under research [3], [4], [5], [6], [7], [8]. This follows as a natural consequence of the IoT expansion, and the urge to handle it. Thus newer and much more performant infrastructures are elaborated both by CSPs and MNOs [9], [10]. Concurrently, IoT devices are getting closer to being actually *ubiquitous* i.e. closer to Mark Weiser's idea of *hundreds of wireless computing devices per person*. This is already true in some scenarios e.g. planes generate around 10 Terabytes of data in 30 minutes. Such cases require very responsive and robust systems for sensor data processing, and could not rely on remote hosts for it, even if these are close to the edge.

[1]The term "Fog Computing" has been initially introduced by Cisco in 2012.

A. Common challenges

However, despite being standardized to some extent[11], [12] [2], a global production ready end-to-end solution has not yet been deployed at scales coming close to those of traditional cloud architectures. There is still a set of engineering and purely practical considerations that must be addressed as discussed in Section II-A. In this regard, the LightKone H2020 European Project[3] aims at providing a novel approach for general purpose computations at the edge. The intrinsic complexity that lies in the dissimilarity of IoT devices makes genericity at the edge a very desirable property.

B. The Achlys framework

The following work describes Achlys[4], an implementation of a resilient generic distributed application platform running on the GRiSP base embedded system [5]. Achlys builds upon the principle that application developers should be able to implement edge IoT programs that are self-sufficient, able to run as an independent network. For this reason, we integrate the Lasp[6] programming library for resilient distributed storage. Additionally, we propose an implementation that also distributes functions inside the cluster. This is a fundamental feature that has been a central part of GrispLasp[13], a Master's thesis project that has lead to the first generation of this distributed task model. We take advantage of this unique feature in Achlys in order to improve it and propose a second version of a general purpose computing platform for wireless IoT sensor networks.

[2]etsi.org
[3]lightkone.eu
[4]ikopest.me
[5]grisp.org
[6]lasp-lang.org

The remainder of this article is structured as follows. First we portray the landscape of the current edge computing state of the art and some key enabling technologies. Then we present a structural overview of Achlys followed by relevant use cases. Finally, we discuss further improvements and draw conclusions based on current evolution of the project.

II. Context and related work

A. Edge computing

The substantial amount of efforts towards standardized edge computing deployments has lead to a particularly dynamic research domain. Industry actors such as Intel, Huawei and Nokia [14], [15] are providing strong incentives for catalysts such as the 5G network [1], [2], [16], [17], [18]. Moreover, the continuous growth of Internet of Things (IoT) with exponential momentum [19], [20] induces several strains on traditional cloud architectures :

- Enormous volumes of data collected at the edge requiring persistent storage
- High level of heterogeneousness among IoT applications
- Intense computing loads and network traffic

B. Conflict-Free Replicated Data Types

Extensive research efforts are dedicated to solve one of the central problems that hold distributed and decentralized applications back[21], [22], [23], [24]. That is, resolving conflicts when multiple actors can modify the same data entity at any time. Our implementation relies on the Lasp library that provides a wide range of CRDT types that abstract this difficulty to application developers.

C. Wireless Big Data Analytics

It is also suggested that partial big data analytics at the edge in WSN configurations[25] could perform much better rather than only centralized cloud analytics. The Achlys framework will be enabling big data analytics and machine learning by definition as it offers distributed knowledge (Lasp) and computing (Erlang) at the edge in a generic way.

III. Contributions

In this section, we briefly discuss the requirements of the Achlys application framework and its purpose in relationship with the global edge computing paradigm.

A. Fault tolerance

Ensuring fault tolerance is an essential part for generic edge computing[26]. In order to fit the vision of the LightKone project, Achlys strives to guarantee this property.

This implies that Achlys must be able to continue functioning even in case of system failure. These failures can be, but are not limited to :

- **Network partition** : a node or a set of nodes that are isolated from the rest of the network must be able to run and to preserve their interoperability with other nodes in case the network is repaired.
- **Hardware failure** : if a hardware component becomes dysfunctional, it should be contained such that the application preserves a maximum amount of features.

B. Genericity

Achlys provides a general purpose task model solution using Erlang high order functions. Since Erlang functions are no different to objects or variables, they can be propagated over a network in the same manner. Hence the goal of Achlys is to provide programmers with an API that allows them to easily disseminate generic tasks in a cluster and be able to retrieve the results if desired. As handling heterogeneousness is a highly complex task for smart services at the edge[27], [28], [29], [30], the Achlys prototype aims at bringing genericity at application level. This integrates in a larger vision of future Internet, in which physical components will be virtualized[31], [32], [33]

C. Data consistency

Since distributed storage is one of the main goals of Achlys, it should be able to handle concurrent modifications and guarantee that entries remain eventually consistent. This requires a valid programming model that preserves the δ-CRDT properties inherited through Lasp. Fulfilling this requirement is essential as the edge model's purpose is to enable storage closer to end users, which cannot be achieved with inconsistent data.

IV. Overview of the system design

In the following section, we present a high level description of *Achlys*, an Erlang[7] implementation of a framework that combines the power of δ-CRDTs[24] with the Lasp[34], [35] library, and the GRiSP Runtime software. It provides application developers a way to build resilient distributed edge IoT applications.

A. GRiSP base

The GRiSP base board is the embedded system used to deploy Achlys networks in the current experimental phase. Its main advantage over other hardware is that it fits the right size[36] to run relevant Erlang applications, that is[8] :

- Microcontroller : Atmel SAM V71, including :
  - ARM Cortex M7 CPU clocked at 300MHz
  - 64 MBytes of SDRAM
  - A MicroSD socket
- 802.11b/g/n wireless antenna
- SPI, GPIO, 1-Wire and UART interfaces

B. GRiSP

Figure 1 depicts how the GRiSP architecture is designed. The RTEMS[9] (RTOS-like set of libraries) component is embedded inside the Erlang VM and makes it truly run on *bare metal*. Achlys greatly benefits from this unique design since

[7] erlang.org
[8] for full specifications please refer to grisp.org
[9] rtems.org

it allows a much more direct interaction with the GRiSP base hardware.

The GRiSP board can be equipped with Digilent Pmod[10] modules. The latter offer a very wide range of sensing and acting features that can be accessed at application level in Erlang. It is no longer necessary to write drivers in C in order to add new hardware features to extend the range of functionalities.

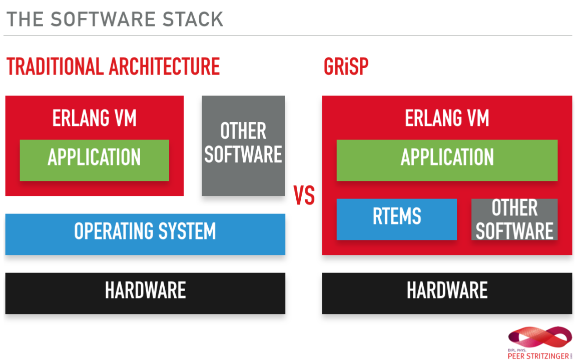

Fig. 1. The GRiSP software stack compared to traditional designs. The hardware layer for GRiSP refers to the *GRiSP base* board that is currently available. Reprinted from GRiSP presentation by Adam Lindberg.

C. Lasp

Lasp is central in the Achlys framework as it is able to materialize the CRDTs that hold distributed data across all nodes in a cluster. It is used in order to support redundancy accross replicas and guarantees that values will eventually converge on all nodes[34], [35].

As Achlys is still under active development, it requires every incremental change to be tested and validated. With Achlys it also implies that we must deploy applications on our embedded systems and verify that they perform correctly using Achlys. A substantial amount of this experimental work has been the object of a Master's thesis[13]. This work has led to lots of adjustments including a more efficient Lasp configuration for the proposed system. The **delta-based dissemination mode** available in the Lasp library unlocks a particularly valuable benefit of δ-CRDTs by propagating only *delta-mutators*[37], [24] i.e. update operations instead of the full state. This way we achieve significantly less traffic between nodes with a much smaller memory footprint.

Partisan[38] is another innovation that Lasp build on top of that is a core component in Achlys. It provides Achlys with a highly resilient alternative communication layer used instead of the default distributed Erlang module. This layer combines the HyParView[39] membership and Plumtree[40] anti-entropy algorithms to enable hybrid gossiping between nodes.

[10]digilentinc.com

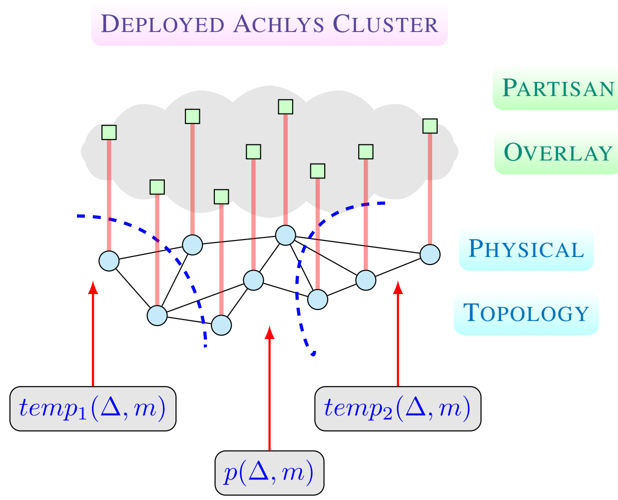

Fig. 2. An example of an Achlys network measuring temperature and pressure.

Figure 2 shows a conceptual overview of how a WSN setup of Achlys nodes that highlights several elements :

- The bottom layer consists of functions $temp_{1,2}(\Delta, m)$,$p(\Delta, m)$ that represent input streams of data based on environment variables measured by the nodes, in this example temperatures and pressure.
- The physical topology reflects the real world configuration of the nodes where each edge implies that the two vertices are able to establish radio communication.
- The virtual overlay that we are able to build using partisan. Achlys provides functions to clusterize GRiSP nodes through Partisan and therefore partitions such as shown by dotted blue lines are abstracted away by the eventual consistency and partition tolerance properties. Physically isolated parts of the network keep functioning as if they would before, and seamlessly recover once the links are reestablished.

D. Local aggregation

The vast majority of raw IoT sensor data is usually very short-lived inside systems and ultimately leads to unnecessary storage. Hence in Achlys we introduce configurable parameters for aggregation of sensor data. This way programmers can still benefit from distributed storage but also take advantage of local memory or MicroSD cards to aggregate raw measurements and propagate mean values. The network loads and global storage volume are thus decreased and overall scalability is improved.

E. Generic task model

Achlys will provide developers the ability to embed Erlang high order functions through a simple API as shown in

Table I. The first implementation of this feature in GrispLasp was the first step towards generic computing using replicated high order functions i.e. gossiping custom subprograms as a feature. Each node was listening for tasks and upon receiving one could decide based on load-balancing mechanisms and destination targeting information if it needed to execute it.

This prototype was used to generate replicated meteorological sensor data aggregations via generic functions supplied with specific tasks. A live dashboard of the currently converged view of the data was built and could run on a laptop. As long as that web client host was able to reach any node in the network, it could output its live view of the distributed storage. Unfortunately, this prototype was facing memory issues that introduced severe instability in the system and ultimately node failures. Nonetheless, it serves as a starting point for a more elaborate task model that builds upon enhancements made with a thorough focus on stability and memory management.

The full implementation and a very precise architecture overview are still available[11]. It is a starting point for Achlys as the model itself has proved to work on hosts that were not constrained by memory. Therefore it will be reused in order to construct a more robust and reliable model which will be added in future releases of the Achlys framework.

V. Use cases

As mentioned in Section IV-E, Achlys is the successor of GrispLasp, that has served as a proof of concept in concrete deployments. In this section, we display use cases based on example applications done previously.

A. Live IoT sensor dashboard

Since our framework is implemented in Erlang, it is also possible to integrate it in Elixir[12] applications. Elixir is a programming language that is built on top of Erlang and adds several very popular web development features. This makes it possible to implement a web server that runs only at the edge and that can interact with an entire Achlys cluster as soon as a single node is reachable.

Our previous work has already allowed us to display a minimal version of this use case implementation in the context of the LightKone project. Figure 3 shows an actual live display of recorded magnetic field data recorded with Digilent Pmod_NAV modules attached to GRiSP boards in the cluster. Moreover, a variety of other sensing modules are handled by the GRiSP software and hence available in Achlys :

- Temperature
- Pressure
- Acceleration
- Orientation
- Altitude
- Ambient light levels
- Object proximity

[11] grisplasp.github.io/grisp-lasp
[12] elixir-lang.org

Monitoring these variables can be extremely valuable for some systems as they can be used for analytics and cost-effectiveness improvements[41]. But with its generic task model, Achlys bypasses the need for a system stoppage for behavioral updates. This opens possibilities for developers to propagate functions dynamically from any point in the network.

Furthermore, this feature can also be integrated in an edge web client available for end users. For instance, the web client shown in Figure 3 could be extended to expose an interface that allows users to specify intervals for sensing, and to aggregate the values locally until a threshold is reached and then propagate them in the distributed database. These control instructions can be embedded inside Erlang functions and sent using the task model. Figure 4 shows an Erlang snippet that can be used to send a task to nodes that periodically try to fetch some from a CRDT.

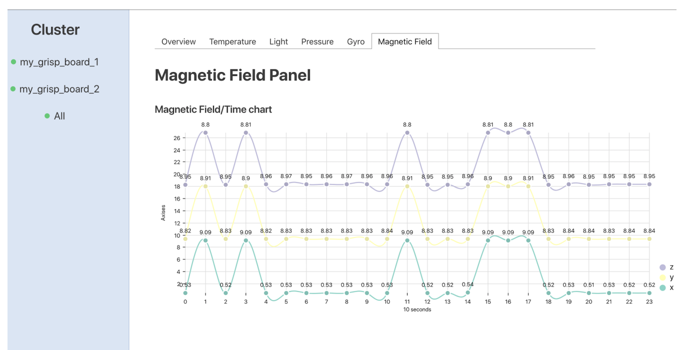

Fig. 3. A web client running on the edge and monitoring magnetic field sensor data from the distributed database.

B. Smart agriculture

The previous use case can be extrapolated into a hypothetical application in the farming sector. Existent solutions such as ThingsBoard[13] usually imply having a gateway to extract data

[13] thingsboard.io/smart-farming

```
@ Periodical calls to find
@ available tasks on each node
erlang:send_after(Cycle, self(), trig)

@ Handle the periodical message
handle_info(trigger) ->
    find_and_start_task()
...
@ From single node
@ Create and propagate new task
F = Sense(DeltaInterval, Treshold)
add_task({senseTask, Destinations, F})
```

Fig. 4. Example usage of the task model API for listening on the tasks CRDT and running available functions.

Function	Arguments	Description
`add_task`	`{Name, Targets, Fun}`	Adds the task tuple `{Name, Targets, Fun}` to the tasks CRDT
`remove_task`	`TaskName`	Removes the task named `{TaskName}` from the tasks CRDT
`remove_all_tasks`	`nil`	Removes all the tasks from the tasks CRDT
`get_all_tasks`	`nil`	Fetches all the tasks from the tasks CRDT
`find_task`	`Name`	Finds the task named `{Name}` from the tasks CRDT
`start_task`	`Name`	Starts the task named **`Name`**
`find_and_start_task`	`nil`	Fetches any available task from the tasks CRDT and executes it
`start_all_tasks`	`nil`	Starts all tasks in the tasks CRDT
`is_running`	`TaskName`	Returns true if the task named **`TaskName`** is already running

TABLE I
GENERIC TASK MODEL API FUNCTIONS.

from the IoT sensor network. Achlys is an alternative where applications could be implemented such that clients would no longer need to send their data to a single gateway, nor rely on cloud infrastructures to be able to manage their network. And research has demonstrated that precise environmental control as available with Achlys and Pmod sensors on GRiSP boards enhances growth, making farms more productive and profitable[42].

VI. LIMITATIONS

Numerous improvements have been implemented in order to achieve a more stable, reliable and scalable framework, and the current Achlys release has been able to benefit from lessons learned thanks to months of experiments in the GrispLasp project. Nonetheless, there are still open challenges that require further improvements to make Achlys more robust.

A. Power supply

Currently, we power our devices using micro-USB cables. This means that it requires either external power-banks, long distance cabling, solar power modules or other sources that can supply the nodes. It is currently not yet determined which option is an optimal fit for which case and hence it will require a more in-depth analysis.

B. Integration of task model backpressure

The general purpose task model is currently being dissected into a more fine-grained API that aims at addressing the trade-off that has to be done between genericity and system stability. A backpressure mechanism needs to be implemented to make Achlys resilient to (possibly deliberate) overloading functions.

VII. CONCLUSION

In this paper we have introduced Achlys, a novel design pattern for distributed storage and general purpose edge computing. We have implemented a framework that is deployable on a wireless network of GRiSP boards. Also, we use Lasp as both distributed database with δ-CRDTs and dynamic management tool to allow dissemination of general purpose computing functions inside the cluster. We think that the advantage of our Erlang prototype is two-fold. It can be used on bare metal GRiSP embedded systems, and directly interface with hardware at application level. And it also gains in ease of deployment with scalable web servers as the Elixir language works hand in hand with Erlang out of the box.
We believe that future work on overcoming the limitations discussed in Section VI as well as experimental deployments at larger scales will help providing overall improvements and an extended set of features to Achlys. This way, we intend to develop an open framework for building distributed IoT applications at the very edge.

ACKNOWLEDGMENT

The authors would like to thank the LightKone European H2020 project for supporting this research.

REFERENCES

[1] I.-P. Belikaidis, A. Georgakopoulos, P. Demestichas, U. Herzog, K. Moessner, S. Vahid, M. Fitch, K. Briggs, B. Miscopein, B. Okyere, and V. Frascolla, "Trends and challenges for autonomic RRM and MAC functionality for QoS provision and capacity expansions in the context of 5G beyond 6GHz," in *2017 European Conference on Networks and Communications, EuCNC 2017, Oulu, Finland, June 12-15, 2017*. IEEE, pp. 1–5.

[2] V. Frascolla, F. Miatton, G. K. Tran, K. Takinami, A. D. Domenico, E. C. Strinati, K. Koslowski, T. Haustein, K. Sakaguchi, S. Barbarossa, and S. Barberis, "5G-MiEdge: Design, standardization and deployment of 5G phase II technologies: MEC and mmWaves joint development for Tokyo 2020 Olympic games," in *IEEE Conference on Standards for Communications and Networking, CSCN 2017, Helsinki, Finland, September 18-20, 2017*. IEEE, pp. 54–59.

[3] Smart World of IoT– The Edge is Getting Smarter, Smaller, and Moving Further Out! - Part 1. [Online]. Available: http://www.embedded-computing.com/iot

[4] H. Koumaras, D. Tsolkas, G. Gardikis, P. M. Gómez, V. Frascolla, D. Triantafyllopoulou, M. Emmelmann, V. Koumaras, M. L. G. Osma, D. Munaretto, E. Atxutegi, J. S. de Puga, O. Alay, A. Brunstrom, and A.-M. C. Bosneag, "5GENESIS: The Genesis of a flexible 5G Facility," in *23rd IEEE International Workshop on Computer Aided Modeling*

and Design of Communication Links and Networks, CAMAD 2018, Barcelona, Spain, September 17-19, 2018. IEEE, pp. 1–6.

[5] A. Kumar, M. Zhao, K. Wong, Y. L. Guan, and P. H. J. Chong, "A Comprehensive Study of IoT and WSN MAC Protocols: Research Issues, Challenges and Opportunities," pp. 1–1.

[6] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," vol. 4, no. 5, pp. 1125–1142.

[7] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita, "Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources," vol. 36, no. 7, pp. 1420–1428.

[8] X. Su, P. Li, Y. Li, H. Flores, J. Riekki, and C. Prehofer, "Towards Semantic Reasoning on the Edge of IoT Systems," in *Proceedings of the 6th International Conference on the Internet of Things*, ser. IoT'16. ACM, pp. 171–172. [Online]. Available: http://doi.acm.org/10.1145/2991561.2998469

[9] Blockchain on AWS. [Online]. Available: https://aws.amazon.com/partners/blockchain/

[10] S. Ziegler and A. Brékine, "D6.3 – Second year report on standardization, dissemination and exploitation achievements," p. 59.

[11] I. ETSI, "ETSI_MEC." [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp30_MEC_Enterprise_FINAL.pdf

[12] N. ETSI, "ETSI_NGP."

[13] A. Carlier, I. Kopestenski, and D. Martens, "Lasp on Grisp : Implementation and evaluation of a general purpose edge computing system for Internet of Things."

[14] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[15] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on Multi-Access Edge Computing for Internet of Things Realization," vol. 20, no. 4, pp. 2961–2991. [Online]. Available: http://arxiv.org/abs/1805.06695

[16] V. Frascolla, B. Okyere, S. Mumtaz, J. Rodriguez, K. M. S. Huq, A. Georgakopoulos, E. A. Kosmatos, A. Skalidi, P. Demestichas, T. Oikonomou, D. Kritharidis, P. Panagiotopoulos, P.-K. Chartsias, B. Miscopein, A. D. Domenico, R. E. Chall, J. Estavoyer, K. Briggs, S. Vahid, M. Filo, K. Moessner, and U. Herzog, "Breaking the Access Technologies Silos by Enhancing MAC and RRM in 5G+ Networks," in *2018 European Conference on Networks and Communications, EuCNC 2018, Ljubljana, Slovenia, June 18-21, 2018*. IEEE, pp. 1–9.

[17] G. K. Tran, H. Nishiuchi, V. Frascolla, K. Takinami, A. D. Domenico, E. C. Strinati, T. Haustein, K. Sakaguchi, S. Barbarossa, S. Barberis, and K. Yunoki, "Architecture of mmWave Edge Cloud in 5G-MiEdge," in *2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018, Kansas City, MO, USA, May 20-24, 2018*. IEEE, pp. 1–6.

[18] M. Ulbricht, P. Dockhorn, U. F. Zia, C. Liss, E. Zetserov, K. Habel, and M. Parker, "– CHARISMA – 5G Low Latency Technologies and their Interaction with Automotion Control Loops ICINC 2018," p. 25.

[19] A. Shoker, J. Leitao, P. V. Roy, and C. Meiklejohn, "LightKone: Towards General Purpose Computations on the Edge," p. 12.

[20] . A. p. r. a. a. c. P. do not include sales tax. IoT: Number of connected devices worldwide 2012-2025. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[21] C. Baquero, P. S. Almeida, and A. Shoker, "Making Operation-based CRDTs Operation-based," p. 15.

[22] A. Barker. Implementing a Garbage-Collected Graph CRDT (Part 1 of 2). [Online]. Available: http://composition.al/CMPS290S-2018-09/2018/11/12/implementing-a-garbage-collected-graph-crdt-part-1-of-2.html

[23] R. Brown, Z. Lakhani, and P. Place, "Big(ger) Sets: Decomposed delta CRDT Sets in Riak," pp. 1–5. [Online]. Available: http://arxiv.org/abs/1605.06424

[24] P. S. Almeida, A. Shoker, and C. Baquero, "Delta State Replicated Data Types," vol. 111, pp. 162–173. [Online]. Available: http://arxiv.org/abs/1603.01529

[25] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," vol. 5, pp. 6757–6779. [Online]. Available: http://ieeexplore.ieee.org/document/7883826/

[26] K. Karlsson, W. Jiang, S. Wicker, D. Adams, E. Ma, R. van Renesse, and H. Weatherspoon, "Vegvisir: A Partition-Tolerant Blockchain for the Internet-of-Things," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, pp. 1150–1158. [Online]. Available: https://ieeexplore.ieee.org/document/8416377/

[27] M. Du, K. Wang, Y. Chen, X. Wang, and Y. Sun, "Big Data Privacy Preserving in Multi-Access Edge Computing for Heterogeneous Internet of Things," vol. 56, no. 8, pp. 62–67.

[28] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An Information Framework for Creating a Smart City Through Internet of Things," vol. 1, no. 2, pp. 112–121.

[29] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. R. Biswas, and K. Moessner, "Enabling smart cities through a cognitive management framework for the internet of things," vol. 51, no. 6, pp. 102–111.

[30] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," vol. 1, no. 1, pp. 22–32. [Online]. Available: http://ieeexplore.ieee.org/document/6740844/

[31] F. Esposito, A. Cvetkovski, T. Dargahi, and J. Pan, "Complete edge function onloading for effective backend-driven cyber foraging," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8.

[32] V. Ivanov, Y. Jin, S. Choi, G. Destino, M. Mueck, and V. Frascolla, "Highly efficient representation of reconfigurable code based on a radio virtual machine: Optimization to any target platform," in *25th European Signal Processing Conference, EUSIPCO 2017, Kos, Greece, August 28 - September 2, 2017*. IEEE, pp. 893–897.

[33] C. S. Meiklejohn and P. Van Roy, "Loquat: A framework for large-scale actor communication on edge networks." IEEE, pp. 563–568. [Online]. Available: http://ieeexplore.ieee.org/document/7917624/

[34] C. S. Meiklejohn, V. Enes, J. Yoo, C. Baquero, P. Van Roy, and A. Bieniusa, "Practical Evaluation of the Lasp Programming Model at Large Scale - An Experience Report," pp. 109–114. [Online]. Available: http://arxiv.org/abs/1708.06423

[35] C. Meiklejohn and P. Van Roy, "Lasp: A Language for Distributed, Coordination-free Programming," in *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming*, ser. PPDP '15. ACM, pp. 184–195. [Online]. Available: http://doi.acm.org/10.1145/2790449.2790525

[36] L. Adam. Wireless Small Embedded Erlang Applications with Grisp Hardware Boards. [Online]. Available: http://www.erlang-factory.com/berlin2016/adam-lindberg

[37] P. S. Almeida, A. Shoker, and C. Baquero, "Efficient State-based CRDTs by Delta-Mutation." [Online]. Available: http://arxiv.org/abs/1410.2803

[38] C. Meiklejohn and H. Miller, "Partisan: Enabling Cloud-Scale Erlang Applications." [Online]. Available: http://arxiv.org/abs/1802.02652

[39] J. Leitao, J. Pereira, and L. Rodrigues, "HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast," in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN '07. IEEE Computer Society, pp. 419–429. [Online]. Available: https://doi.org/10.1109/DSN.2007.56

[40] ——, "Epidemic Broadcast Trees," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, ser. SRDS '07. IEEE Computer Society, pp. 301–310. [Online]. Available: http://dl.acm.org/citation.cfm?id=1308172.1308243

[41] B. Qi, L. Kang, and S. Banerjee, "A vehicle-based edge computing platform for transit and human mobility analytics," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing - SEC '17*. ACM Press, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3132211.3134446

[42] Department of Instrumentation and Control Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India, V. Sravani, S. K V, Department of Instrumentation and Control Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India, S. Bhargava, Department of Instrumentation and Control Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India, V. D'Almeida, and Department of Instrumentation and Control Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India, "Design and Implementation of a Smart Controller in Agriculture for Improved Productivity," vol. 18, no. 1, pp. 45–51.