On the Design of Coordination Services for IoT

Tiago Carvalho, Antonio Casimiro, Alysson Bessani
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

***Abstract*—Complex Internet of Things (IoT) applications usually comprise many devices and servers that may differ significantly in their hardware and software capabilities. Due to the possibly large number and heterogeneity of these devices and their constant change of state, keeping an up-to-date list of the participating devices at any given time is a challenging problem. Further, it is not uncommon for these devices to coordinate with each other, requiring the execution of complex and expensive distributed protocols. At the same time, there is also the problem of ensuring that joining devices satisfy the high-level security requirements defined for critical IoT applications. However, existing work focuses on tackling these issues individually, and no complete solution exists. We propose solving these issues in a unified way by designing a coordination/configuration service to support IoT applications. We adopt fundamental ideas underlying coordination services used in cloud settings to support Internet-scale applications (e.g., Apache ZooKeeper), adapting and extending them to deal with specific and significantly different IoT application requirements, namely usability, security, and synchronization.**

***Index Terms*—Coordination, internet of things, security.**

I. Introduction

Smart environments are becoming common today, be it in the context of smart homes and buildings, smart factories, or even smart cities. In these environments, a variety of devices, such as cameras, thermometers, microphones, motion sensors, etc., gather data that is processed and used to execute certain tasks, typically associated with the physical world, e.g., using cameras in a smart car to detect pedestrians crossing the road and automatically brake. This data can be processed locally or on a central system to which it is sent. Usually, these Internet of Things (IoT) applications are composed of a multitude of devices with different hardware and software capabilities, making them very heterogeneous.

Due to the large number of devices that compose these systems and the system dynamics (movement, failures, etc.), keeping an up-to-date list of the active devices at any given time is a challenging problem. A typical solution for this kind of problem is the use of *coordination services* (e.g., Zookeeper [1]), which can track connected processes in a distributed system, i.e., its membership. Coordination services usually exhibit three main features [2]: (1) highly-available small storage, (2) interface with synchronization power, and (3) client failure detection. These features enable coordination services to seamlessly support tasks such as leader election, mutual exclusion, membership management, and configuration storage. Unfortunately, existing coordination services are designed to support applications inside the trusted and constrained perimeter of data centers.

Dealing with device dynamism in open, untrusted environments where many applications are deployed makes using existing coordination services inappropriate for IoT systems. More specifically, these devices are often used in critical applications in which the integrity and confidentiality of the data they gather must be ensured. Therefore, it becomes fundamental to develop solutions regarding security that account for all these issues and aspects to regulate the participation of devices in IoT systems properly. For example, existing coordination services cannot help an application to only accept devices from a pre-defined manufacturer if they are running the most recent version of a given software stack. One way to solve this problem is *remote attestation* (RA), a method that evaluates if a device has hardware and software that satisfy the security requirements. Although this method ensures that a certain level of security is met, RA services typically only focus on assuring the security of the *attester* (the devices joining the system) and rarely take into account the failure of the *verifiers* (the coordination service), as the latter is usually assumed to be a trusted entity. This assumption is difficult to substantiate if the verifier is deployed in an untrusted environment.

Another issue related to the use of coordination services for IoT is the mismatch between the coordination capabilities required in data center applications and the IoT-related cyber-physical systems. In the data center environment, the main problem is making the hosts agree about something (e.g., system membership, lock possession, who is the leader) despite failures. In an IoT environment, besides this problem, there are spatiotemporal aspects (e.g., the physical location of a device at some time instant), that need to be supported by the coordination service to help devices synchronize their activities in a cyber-physical system.

In the rest of this paper, we elaborate on the case for IoT coordination services, considering some motivating applications and discussing the specific requirements of those services. We conclude the paper by briefly outlining the design of an IoT coordination service and discussing some related work.

II. The Case for IoT Coordination Services

Coordination Services provide a consistent and highly available data store with enough synchronization power for client processes to execute tasks, such as mutual exclusion and leader election. Two examples of this kind of service are ZooKeeper [1], a crash-tolerant coordination service with a node structure, and DepSpace [3], a Byzantine fault-tolerant coordination service with a tuple space structure. Unfortunately, these services were primarily designed for data centers

2833-292X/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN-S60304.2024.00019

and cannot cope with the unique challenges found in an IoT environment.

A. Challenges of IoT

The IoT differs significantly from other environments due to its usage of devices that have a significant degree of interaction with the physical world (the so-called cyber-physical systems), resulting in critical applications that deal with a lot of data. For this reason, many inherent characteristics and issues pose a challenge when developing middleware and applications for this setting. Out of all the issues, we decided to focus on the following:

- **Heterogeneity** – devices used in these environments may vary significantly in their hardware and software capabilities. Any solution targeting IoT should be aware of these constraints and offer accessible interfaces.
- **High dynamism** – applications in these environments often employ a dynamic and extensive set of devices, which makes keeping an up-to-date list of participating members [4] a complicated task.
- **Critical applications** – devices are often part of applications that require interactions with the real world, either by collecting data or by executing specific tasks, meaning that faulty behavior of both devices and infrastructure could be very damaging and challenging to revert [5]. The consequence is that security (in particular, correctness assurance) and runtime fault avoidance and tolerance are typically more important than recovery.
- **Physical synchronization** – differently from data center applications, contextual information such as time, position, and velocity are fundamental for coordinating devices in many IoT applications.

Apart from dynamism, existing coordination services such as ZooKeeper are not prepared to deal with these challenges, hence creating the need for a new kind of system.

B. Use Cases

To better illustrate the need for IoT coordination services, in this section, we describe three use cases in which such systems can be useful.

1) IoT Membership Management: The membership of a system is represented by an up-to-date list of active participants, which is an especially complicated task in some IoT applications due to the scale and dynamism of the device group. In cloud-based systems, membership management is performed using a coordination service due to two fundamental characteristics: failure detection of client processes and highly available consistent storage [2]. Typically, a node that wants to join the system will request the coordination service to add it to the application's membership without any concern for its correctness. Although this might be appropriate in a data center, in an IoT scenario, accepting every device that requests to join an application can constitute a considerable threat as these environments are typically insecure. Therefore, the coordination service must support the definition of policies that specify the requirements for participating devices to ensure a certain degree of security. A perfect solution for addressing this requirement is to employ a remote attestation protocol [6] in which the coordination service is the verifier.

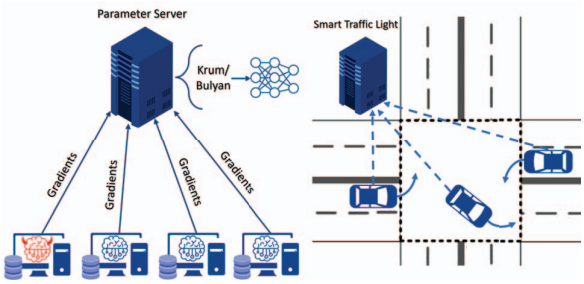

Fig. 1. Example use cases: BFT federated learning (left) and autonomous vehicle intersection management (right).

2) Autonomous Vehicle Coordination: Shortly, our streets will be filled with autonomous self-driving vehicles, which will require coordination of their maneuvers. Such coordination enables proper access to shared resources, such as intersections and parking slots, and the safe execution of complex tasks, such as platooning and ramp merging [7]. Some of these tasks, such as parking slot management, typically use a centralized infrastructure to maintain the state of both the environment and the participants (e.g., keep a list of the slots in a smart parking lot and their occupation state as well as the state of the occupying vehicles [8]). Other tasks, such as platooning, rely primarily on inter-vehicle communication to coordinate the vehicles during the task. In addition, there are also some tasks, such as crossing an intersection, which can be performed in both ways depending on the available resources (e.g., if a "smart" centralized entity does not manage an intersection, the order of crossing is decided in a distributed way between the vehicles [9]).

This autonomous vehicle coordination use case raises some challenges that are common to several other IoT scenarios. First, it requires a safety-critical design strategy as even minor faults may lead to significant costs, both in terms of human life and in a monetary sense. Therefore, assuring the correctness of every system entity or component is essential. In addition, it may involve many highly dynamic participants, making it challenging to keep an up-to-date view of the system's membership and state. Although some systems and tasks do not require it, there is always a benefit in having an up-to-date and centralized source of information available to joining vehicles, for example, or even to facilitate enabling the execution of other tasks that are not available in the system.

3) Robust Distributed Machine Learning: Machine learning models have grown in complexity and in the amount of data processed, which requires a lot of computation resources. Therefore, most machine learning implementations are now distributed. Most of these implementations rely on a core component, a parameter server, which is in charge of updating the parameter vectors, while workers perform the actual update estimation based on the share of data they have access to. In

the IoT setting, these workers can directly gather this data, which can be ensembles of simple sensors attached to a device hub, typically a regular computer. This scenario is illustrated in Figure 1.

The parameter server/aggregator can be used to tolerate further computation errors, stalled processes, or attackers trying to compromise the system by implementing aggregation rules to make the model Byzantine resilient (e.g., [10], [11]). This approach was implemented in some practical systems such as Garfield [12]. Essentially, these rules take all the gradients computed by the workers as input and discard those that differ significantly from the others, meaning that these rules are only effective if most workers are correct.

Similarly to the autonomous vehicle environment, the parameter server scenario also has a few characteristics and issues that should be considered when developing appropriate solutions. First, this scenario can be heterogeneous as the workers may be devices of any kind with a wide range of computational capabilities. Although the Byzantine aggregation rule can prevent malicious workers from impacting the parameter vector, these adversaries can still interact with the parameter server and, as such, may constitute a threat to the system. Therefore, some guarantees must be given regarding the correctness and integrity of the workers. Lastly, in some systems, the aggregation rule is executed when the parameter server collects gradients from all workers. This requires failure detection to prevent crashed workers from stalling the entire system.

C. Issues with Existing Coordination Services in IoT

To a certain extent, the tasks described earlier in this section can all be reduced to coordination problems. Therefore, one could question why we cannot simply use a coordination service to perform the intended tasks in these scenarios. The problem with regular coordination services, such as ZooKeeper [1], is that they are designed to be used in data center environments where nodes are trusted. IoT environments, however, are typically untrusted, and therefore, ensuring the correctness and integrity of the nodes (i.e., the devices) and tolerating their eventual faulty behavior (i.e., Byzantine fault tolerance) are essential features in this field. Although some coordination services are able to tolerate Byzantine faults (e.g., DepSpace [3]), they are primarily concerned with the service replicas, with little concern given to the correctness, heterogeneity, and dynamicity of the set of clients (IoT devices). Therefore, existing coordination services are not suited for IoT environments and, as such, cannot be considered appropriate solutions to perform these tasks.

III. Proposed Approach

To address the challenges identified in the previous section, we outline a holistic coordination service suited to the IoT environment with the following features:

1) Given the threat landscape to which the service will be exposed (outside the safe borders of a data center backplane), the coordination service must be *fault and intrusion tolerant* [13], just like a small-scale permissioned blockchain [14], to prevent successful attacks on the system from undermining the security guarantees that it brings to IoT applications. This requirement is essential to preserve the "trust anchor" guarantee typically provided by coordination services.
2) It must incorporate *remote attestation capabilities* [6] to verify if devices' characteristics (e.g., software stack, hardware model) are in accordance with pre-defined application requirements before accepting the device in the system.[1]
3) It must provide an *extensible coordination kernel* [2] with primitives capable of solving consensus, similar to the ones provided by ZooKeeper [1] or any other modern coordination service. These primitives should make it easy to implement synchronization tasks (e.g., consensus, leader election, mutual exclusion), membership management, failure detection, and limited storage (primarily for configuration data). Unlike ZooKeeper, these primitives must be extensible [2] to incorporate application-specific processing at the coordination service.
4) Besides traditional synchronization, the coordination kernel primitives must *support temporal and spatial aspects*. These primitives shall enable the service to solve cyber-physical synchronization tasks for which this information is fundamental (e.g., drone orchestration [16]). Given the untrusted nature of IoT environments, the coordination service might need to incorporate mechanisms for its replicas to generate meaningful timestamps in a distributed way (e.g., [17]) and check the devices' location (e.g., [18]).

The main goal of our approach is to have a system capable of providing a reliable source of information about which devices can participate in an application and to which extent these devices satisfy certain properties (e.g., their software is up-to-date) to ensure the security requirements are met. The objective is to significantly increase the trust between interacting IoT devices by regulating their participation in these applications. This system would serve as a basis for developing IoT applications as it provides multiple functionalities and features essential in various use cases and scenarios of the field.

A coordination service with the abovementioned features would be a Byzantine fault-tolerant (distributed) infrastructure that supports remote attestation, coordination primitives, and membership management. It must consider that each connected device belongs to one application, with the possibility of having multiple applications running simultaneously (multi-tenancy), as illustrated in Figure 2. Each of these applications

[1]Remote attestation is a method by which a node, the *attester*, proves to another, the *verifier*, some properties of its hardware and software [6]. To achieve this, the attester generates evidence of its properties, which the verifier will evaluate using a *policy* - set of rules that specify which devices can join the application. Typically, this is done using secure hardware components, such as a Trusted Platform Module [15], deployed on the attester.

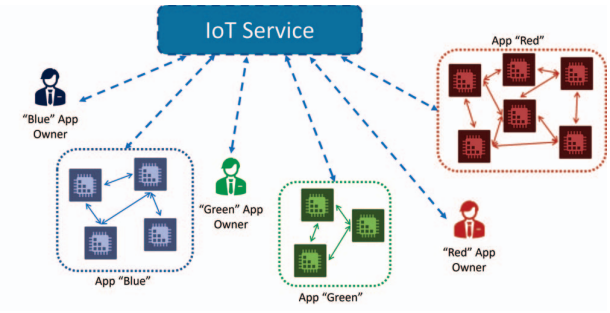

Fig. 2. Diverse IoT applications managed by the IoT coordination service (each color represents a different application and its devices).

can have different requirements, specifications, and contexts (e.g., a single IoT coordination service will be able to manage devices belonging to an autonomous vehicle application and a different set of devices belonging to a distributed machine learning application). This requires the solution to be versatile to adapt to a variety of applications. The coordination kernel's extensibility would allow the applications' owners to introduce their own code to enable the execution of the necessary tasks without modifying the whole system.

Since every application has different security requirements, the remote attestation protocol should use custom policies to define the hardware and software properties required to participate in an application. By supporting both extensions and custom policies, we can properly address the heterogeneous nature of IoT, as our system can more easily adapt not only to different applications but also to a variety of devices. Moreover, the Byzantine fault tolerance capabilities of the service enable the possibility of the verifier failing, which is a significant advantage over most attestation services as they are typically considered a trusted entity by assumption.

Whenever a device is correctly attested, it should be added to the membership of its application. This essentially turns the membership from being just a list of active participants to a list of active participants that satisfy the security requirements for its application. To ensure that registered devices are active, some form of failure detection must be employed to check their status; for example, devices need to periodically send ping messages to signal that they are still alive. This coordination kernel would allow the storage of configuration data (e.g., attestation policies for each application) or data that the devices need to coordinate. For example, in the intersection management scenario (§II-B), our solution could maintain and manage the state of the intersections, which includes the vehicles that are currently crossing as well as the vehicles that are approaching or waiting for their turn to cross. By introducing an extension to decide the crossing order, our solution could act as a "smart traffic light", with the added benefit of the security guarantees provided by the attestation procedure. In the BFT distributed machine learning scenario, our solution could act as a parameter server, with workers sending the gradients to be stored in the coordination kernel. Before computing and sending any gradients, these workers would be attested to guarantee they satisfy the security requirements. An implementation of a BFT aggregation rule would be introduced in the system as an extension to be executed when, for example, gradients from all workers have been collected.

The introduction of time and location information on the coordination kernel brings several novel issues to be addressed in IoT coordination. First and foremost, the ability to generate physical timestamps and verify provided device locations (possibly taking into account communication latency) requires the use of more powerful synchrony models than the asynchronous or partially synchronous models employed in practical fault-tolerant systems (e.g., ZooKeeper [1], BFT-SMaRt [19]), or even the use of trusted components (e.g., [20]). An interesting starting point for this new model would be the timed-asynchronous model [21], which only requires clocks to have a bounded drift rate. This is enough to build fail-aware protocols and transform a system where computation and communication latency cannot be bounded on a synchronous system with omission failures. A model like this, which is strictly weaker than the synchronous model, would allow the implementation of coordination tasks requiring accurate timing information, at least when the network is behaving well.

IV. Related Work

To the best of our knowledge, there is no other proposal for using coordination services for solving IoT problems. Nonetheless, there is some related work that addresses different aspects of our proposed solution. Here, we briefly discuss a subset of this existing work.

There are many research efforts being made on remote attestation, and, as such, multiple examples of these services can be found in the literature [22]–[24]. Typically, these services rely on trusted entities to ensure the correctness of the attestation process, with only the recently published SCRAPS [25] taking into account the possibility of the verifier failing. Although SCRAPS has some similarities with our approach (w.r.t. the use of remote attestation), it does not possess any coordination primitives and, hence, is incapable of performing membership management or storing any configuration or coordination data.

ZooKeeper [1] and DepSpace [3] are two coordination services that are closer to our envisioned solution for IoT environments. The former provides an elegant, widely-used coordination kernel, while the latter tolerates Byzantine failures and verifies interactions between clients. However, neither of these services can perform remote attestation or coordinate cyber-physical tasks, to cite two limitations.

V. Conclusion

In this paper, we outlined the challenges, requirements, and some potential use cases for an IoT coordination service. We also outlined a solution that combines remote attestation, coordination primitives, membership management, and Byzantine fault tolerance in a system adapted to the main issues and characteristics of IoT environments. This solution is by no

means complete, and there are aspects, such as the use of time and location in the coordination kernel, that need to be further refined.

ACKNOWLEDGMENT

This work was financially supported by the Fundação de Ciências e Tecnologias (FCT) through the SMaRtChain project (2022.08431.PTDC) and the LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020) and by the European Union's Horizon 2020 Research and Innovation Programme through the VEDLIoT project (agreement 957197).

REFERENCES

[1] P. Hunt, M. Konar, F. P. Junqueira, and B. C. Reed, "Zookeeper: Wait-free coordination for internet-scale systems," in *USENIX Annual Technical Conference*, 2010.

[2] T. Distler, C. Bahn, A. N. Bessani, F. Fischer, and F. P. Junqueira, "Extensible distributed coordination," *Proceedings of the Tenth European Conference on Computer Systems*, 2015.

[3] A. N. Bessani, E. A. P. Alchieri, M. P. Correia, and J. da Silva Fraga, "Depspace: a byzantine fault-tolerant coordination service," in *European Conference on Computer Systems*, 2008.

[4] S. Namal, H. Gamaarachchi, G. M. Lee, and T.-W. Um, "Autonomic trust management in cloud-based and highly dynamic iot applications," *2015 ITU Kaleidoscope: Trust in the Information Society (K-2015)*, pp. 1–8, 2015.

[5] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C. K. Chen, and S.-P. Shieh, "Iot security: Ongoing challenges and research opportunities," *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pp. 230–234, 2014.

[6] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote attestation procedures (rats) architecture," Internet Requests for Comments, RFC Editor, RFC 9334, January 2023, https://www.rfc-editor.org/info/rfc9334. [Online]. Available: https://www.rfc-editor.org/info/rfc9334

[7] S. Mariani, G. Cabri, and F. Zambonelli, "Coordination of autonomous vehicles: taxonomy and survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 1, pp. 1–33, 2021.

[8] A. Khanna and R. Anand, "Iot based smart parking system," *2016 International Conference on Internet of Things and Applications (IOTA)*, pp. 266–270, 2016.

[9] W. Wu, J. Zhang, A. Luo, and J. Cao, "Distributed mutual exclusion algorithms for intersection traffic control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 65–74, 2015.

[10] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NIPS*, 2017.

[11] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*, 2018.

[12] R. Guerraoui, A. Guirguis, J. Plassmann, A. Ragot, and S. Rouault, "Garfield: System support for byzantine machine learning (regular paper)," *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 39–51, 2021.

[13] J. da Silva Fraga and D. Powell, "A fault- and intrusion- tolerant file system," in *IFIP Security Conference*, 1985.

[14] A. Bessani, E. Alchieri, J. Sousa, A. Oliveira, and F. Pedone, "From byzantine replication to blockchain: Consensus is only the beginning," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020.

[15] Trusted Computing Group, "Trusted Platform Module library specification, family 2.0," 2019. [Online]. Available: https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[16] S. He, F. Bastani, A. Balasingam, K. Gopalakrishna, Z. Jiang, M. Alizadeh, H. Balakrishnan, M. Cafarella, T. Kraska, and S. Madden, "Beecluster: drone orchestration via predictive optimization," in *18th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2020.

[17] J. Kirsch, S. Goose, Y. Amir, D. Wei, and P. M. Skare, "Survivable SCADA via intrusion-tolerant replication," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, 2014.

[18] K. Kohls and C. Díaz, "Verloc: Verifiable localization in decentralized systems," in *31st USENIX Security Symposium*, K. R. B. Butler and K. Thomas, Eds., 2022.

[19] A. N. Bessani, J. Sousa, and E. A. P. Alchieri, "State machine replication for the masses with bft-smart," *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 355–362, 2014.

[20] J. Mendonça, A. B. Asl, F. Lucchetti, and M. Völp, "Confirmed-location group membership for intrusion-resilient cooperative maneuvers," in *Proceedings of the 99th IEEE Vehicular Technology Conference (VTC'24)*, 2024.

[21] F. Cristian and C. Fetzer, "The timed asynchronous distributed system model," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 6, 1999.

[22] M. Conti, E. Dushku, and L. V. Mancini, "Radis: Remote attestation of distributed iot services," *2019 Sixth International Conference on Software Defined Systems (SDS)*, pp. 25–32, 2018.

[23] J. Ménétrey, M. Pasin, P. Felber, and V. Schiavoni, "Watz: A trusted webassembly runtime environment with remote attestation for trustzone," *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pp. 1177–1189, 2022.

[24] H. Tan, G. Tsudik, and S. K. Jha, "Mtra: Multi-tier randomized remote attestation in iot networks," *Computers & Security*, vol. 81, pp. 78–93, 2019.

[25] L. Petzi, A. E. B. Yahya, A. Dmitrienko, G. Tsudik, T. Prantl, and S. Kounev, "Scraps: Scalable collective remote attestation for pub-sub iot networks with untrusted proxy verifier," in *USENIX Security Symposium*, 2022.