A Cyber-Physical OS for Enabling Spatio-Temporal Coordination at Geo-distributed Scale

Sandeep D'souza
Carnegie Mellon University
sandeepd@andrew.cmu.edu

Ragunathan (Raj) Rajkumar
Carnegie Mellon University
rajkumar@andrew.cmu.edu

ABSTRACT

Cyber-physical systems (CPS) are increasingly distributed and perform coordinated sensing and actuation over large geographical areas. Examples include local-scale industrial robots, city-scale traffic management, and regional/continental-scale smart grids. To enable coordination at such *geo-distributed* scales, these systems will be deployed across a hierarchy of platforms ranging from embedded sensing/actuation platforms to edge cloudlets, and the cloud. At the same time, multiple applications with different requirements may coexist on the same infrastructure. In this position paper, we argue for a distributed *cyber-physical* OS which can aid in simplifying the development, deployment and management of CPS across geo-distributed infrastructure. In particular, given that cyber-physical coordination occurs in both *space* and *time*, low-latency message-passing, accurate location estimates, along with a shared notion of time are key for safe and reliable operation. Therefore, we advocate for exposing time and location as first-class entities to applications, and focus on the system-level challenges and user abstractions required to enable coordination in CPS at geo-distributed scale.

CCS CONCEPTS

• **Computer systems organization → Cyber-Physical Systems**.

KEYWORDS

cyber-physical systems, spatio-temporal coordination

ACM Reference Format:
Sandeep D'souza and Ragunathan (Raj) Rajkumar. 2019. A Cyber-Physical OS for Enabling Spatio-Temporal Coordination at Geo-distributed Scale. In *Proceedings of NGOSCPS '19.* , 3 pages.

1 INTRODUCTION

Coordination is key to the successful operation of many distributed CPS. This coordination occurs at different spatial and temporal scales, ranging from factory-scale robotic coordination – occurring at the timescale of hundreds of microseconds to a few milliseconds, and city-scale connected vehicles coordinating to improve traffic flow – at the granularity of hundreds of microseconds to a few milliseconds, to regional/continental-scale coordination in the smart grid – which requires clocks to be synchronized to within a few hundred nanoseconds of each other [3].

The common thread binding the above-mentioned CPS is the need for low-latency decision making. The nature of decision making is usually dependent on the analysis of *sensed* data by an intelligent *computational* entity, which in real-time decides a course of coordinated *action/actuation* by its distributed entities. The data-intensive and low-latency nature of decision making makes the cloud in tandem with a hierarchy of edge cloudlets, and embedded endpoints well suited for hosting such applications. The inherent complexity of cyber-physical applications along with the heterogeneity of the infrastructure makes their development, deployment and management a challenging proposition. This has often resulted in the development of application-specific siloed solutions. Therefore, what is required is a distributed framework which solves a key set of challenges common to a range of these systems. Given the spatio-temporal nature of CPS, we posit that *time* and *location* are the two key elements which need to be exposed as first-class entities to cyber-physical applications. Figure 1 highlights diverse distributed applications coordinating in space and time.

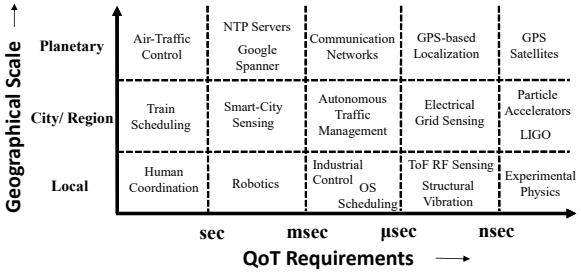

Figure 1: The scale of coordination in time and space [8]

Time plays an important role in enabling coordination among distributed entities [10]. This is especially true for CPS which interact with the real world. A shared notion of time enables (i) events to be ordered at distributed scale, and (ii) coordinated actuation to be scheduled at/by specific time instants [7]. Therefore, maintaining a shared notion of time is critical to the performance and reliable operation of many CPS. While time is key for CPS, the *spatial* aspect is equally important as: (i) sensing and actuation must be scheduled at endpoints (static or mobile) at a particular location, (ii) computation needs to be scheduled at a location within some specified spatial proximity to the sensing/actuation endpoint, so as to meet application-specific low-latency decision-making requirements.

The knowledge of both location and time is often necessary for safe and reliable coordination. However, there is always some uncertainty in an application's perception of the *true* time and location with respect to a specified reference. Often, cyber-physical applications cannot operate safely if this uncertainty exceeds a specified limit. Hence, for both time and location, it is essential that the system expose quality metrics which allow applications to (i) specify their uncertainty tolerances, and (ii) observe the uncertainty in the time/location estimate provided to them by the system. Allowing distributed application components the ability to specify their uncertainty tolerances enables the underlying system to orchestrate the infrastructure to meet them. Exposing the delivered uncertainty

NGOSCPS '19, Apr 15 2019, Montreal, Canada

to applications, allows them to be fault-tolerant and adaptive in the event where the delivered uncertainty exceeds specified limits.

The contributions of this paper are as follows:

- We advocate for a distributed cyber-physical OS to enable spatio-temporal coordination in CPS.
- We advocate for quality metrics which expose time and location as first-class primitives to cyber-physical applications.
- We illustrate the requirements and challenges involved in realizing this cyber-physical OS at geo-distributed scale.

2 CITY-SCALE CONNECTED VEHICLES

We now motivate the utility of a cyber-physical OS for distributed coordination. Coordinating fleets of connected autonomous vehicles (CAVs) for dynamic traffic management at city-scale is an example of an application involving geo-distributed coordination.

The proposed application can be deployed across autonomous vehicles, *Vehicle-to-Infrastructure* (V2I) nodes, edge cloudlets and the cloud. Infrastructure nodes receive location and destination information from vehicles, along with a timestamp associated with a vehicle's presence at that location. This timestamped information is forwarded with *low latency* to a nearby edge cloudlet, responsible for making intersection-level traffic-control decisions. Each cloudlet can periodically summarize and forward their respective state information to the cloud. The cloud is responsible for shaping traffic flow at a macroscopic level. Based on this macroscopic guidance, cloudlets make local decisions for their respective intersections. In addition, a cloudlet can also communicate with cloudlets at its neighboring intersections in order to make local coordinated decisions. Lastly, infrastructure nodes convey directional/velocity instructions received from a cloudlet to the autonomous vehicles, which follow these instructions to improve safety and throughput.

In an ideal world, relevant vehicular information can be inferred from the timestamped events provided by the vehicles. The aggregated information can be used to build traffic state, which in turn can be analyzed to formulate plans of action, such that vehicles coordinate their actions. However, there is always some uncertainty in all time and location estimates. If time and location uncertainties exceed application tolerances, it can lead to unsafe decisions.

We propose that time and location be exposed as first-class entities in a cyber-physical OS. These OS-supported abstractions for time and location, make the associated uncertainty specifiable and observable. For example, more accurate (less uncertain) location and time estimates can be relied upon to formulate policies which lead to tighter vehicular formations, leading to better efficiency, while meeting all safety requirements. On the other hand, during transient increases in uncertainty, exposing this information to the application, allows it to adapt and be fault-tolerant. For the above-mentioned application, adaptation can take many forms including (i) increasing vehicular spacing and/or lowering vehicular speed, or (ii) in an adverse case bringing the system to a safe halt.

Localization often relies on clock synchronization [12], and in cases like GPS, both time and location are derived from the same source. Additionally, there may also be a relationship between an application's time and location uncertainty tolerances. Consider a fleet of moving autonomous vehicles. Given the set of vehicular velocity and inter-vehicular spacing constraints derived from safety requirements, a higher localization accuracy (lower uncertainty) allows a decision-making system to tolerate less-accurate timestamps (higher uncertainty). The converse is also true.

In addition to spatio-temporal coordination, the complexity and geographical scale of the above-mentioned system, also makes the development, deployment and lifecycle management of such cyber-physical applications a challenging problem.

3 ENABLING SPACE-TIME COORDINATION

We now briefly illustrate some of the challenges involved in realizing a cyber-physical OS to enable geo-distributed coordination, and briefly illustrate directions in which possible solutions lie.

3.1 Clock Synchronization and Localization

Clock-synchronization technologies such as GPS, Network Time Protocol (NTP), and Precision Time Protocol (PTP) [8] have made it possible to achieve a reliable and accurate shared notion of time. Similarly, technologies like GPS and UWB [12] provide precise outdoor and indoor localization capabilities. Therefore, given that these technologies are mature, we can often rely on existing technologies and standards. Instead, what is required is a higher-level adaptive software layer which takes in application timing/location requirements, and based on these requirements and system capabilities, orchestrates the system to meet them. In addition, this layer is also tasked with computing the uncertainty in every node's notion of time/location with respect to a reference. This layer is best implemented as a collection of distributed micro-services through which applications can obtain time and location estimates with their associated uncertainties. However, OS-kernel-level abstractions and mechanisms are necessary for (i) enabling *precise* time/location-based scheduling of sensing and actuation, and (ii) exploiting platform-specific hardware features for precise clock synchronization and localization

The scale and complexity of these systems make it desirable for these distributed time and location services to be *autonomous* and *adaptive*. As such, these services alleviate a developer's burden of setting up the infrastructure to meet application-specific needs.

3.2 Fault-tolerance

As CPS operate in the real world, fault-tolerance and adaptability are essential. While, replication is necessary, CPS may also rely on *analytical redundancy*, involving graceful degradation modes. This is useful in the event of the timing or location uncertainties exceeding application tolerances. In such scenarios, the ability of the system to notify applications about uncertainty violations allows applications to adapt. Therefore, it is essential that a cyber-physical OS detect such incidents and notify applications accordingly.

3.3 Security & Privacy

The real-world implications of CPS make security an important concern. Unlike software services, which are hosted in a secure data center, CPS may have physical nodes deployed in public or semi-private spaces. Hence, malicious nodes need to be detected and isolated without violating safety constraints. From a spatio-temporal coordination aspect, time or location spoofing can also adversely impact coordination and result in unsafe decisions. Therefore, for a cyber-physical OS, essential security features include (i)

access control to resources, (ii) isolation between applications and users, (iii) anomaly detection and (iv) a verifiable implementation.

From a privacy standpoint, significant amounts of personal data may be used by CPS for distributed decision making. It is important that all personally-identifiable data be sufficiently anonymized [11] to reduce the risk of exposure in case of data breaches.

3.4 Application Deployment

In the context of CPS, physical access to each embedded endpoint and edge device is challenging. Hence, there needs to be a management layer which makes it seamless to deploy applications across multiple nodes by using an administrative front-end. This layer should also provide the ability to (i) manage and configure the infrastructure, and (ii) distribute and manage application artifacts.

Virtualization technologies like containerization [1] simplify the deployment and life-cycle management of complex applications. Therefore, containerization in conjunction with container-orchestration technologies like Kubernetes [2] can be used to deploy applications at scale. In addition, these technologies can be used as building blocks for developing (i) user-space containerized micro-services to provide a core-set of capabilities, and (ii) management layers for deploying and monitoring applications at scale.

The use of containerization also makes applications portable across a range of platforms, by bundling the application binary and its dependencies as a portable image. This image can be deployed on any platform which (i) has binary compatibility, and (ii) sufficient resources. This significantly decreases the complexity of application deployment. In addition, if application components are built as containers, based on latency specifications and dependency constraints, an intelligent system can automatically choose where/when application components are deployed/scheduled, i.e., to relevant platforms ranging from embedded endpoints, edge cloudlets to the cloud.

3.5 Application Development

Distributed cyber-physical applications generally consist of multiple components which coordinate together in both space and time to achieve a specific outcome. The development of these distributed components is generally a complex endeavor. Therefore, analogous to the POSIX API for a single node system, there needs to be an API that provides a key set of spatio-temporal primitives and services common to a number of application domains, and allows applications to: (i) specify the set of coordinating distributed components, (ii) specify their spatio-temporal (time and location) uncertainty tolerances, (iii) specify latency constraints, (iv) read their current time and location along with the associated uncertainty, (v) schedule events (sensing, computation and actuation) in space and time, and (vi) exchange messages among distributed application components.

To make uncertainty observable and specifiable the following abstractions are useful (i) *Quality of Time* (QoT) [4], i.e., "the *end-to-end* uncertainty bounds corresponding to a timestamp, with respect to a clock reference", and (ii) *Quality of Location* (QoL), i.e, "the uncertainty radius in a location estimate, with respect to a reference". These are quality metrics which expose both time and location as first-class entities. Therefore, all API calls should be centered around these metrics. The work in [4] also introduced the *timeline* abstraction, which allows a distributed application to specify its coordinating components. As defined in [4], a timeline provides a shared *virtual* clock reference to all the distributed components of an application. Therefore, each application component that needs to perform coordinated action, binds to a common timeline, each specifying its respective QoT requirements. The timeline abstraction can also be extended to provide a common location reference to all the bound components, each specifying their desired QoL.

4 RELATED WORK

The utility of a shared notion of time in CPS has been well explored. PTIDES [5] is one such hardware-software framework to model, design and deploy time-critical embedded applications. For automotive and aerospace systems, the Time-Triggered Architecture (TTA) [9] provides a deterministic way to deploy systems using a shared clock. However, both PTIDES and TTA are designed for embedded applications and cannot scale to geo-distributed cyber-physical applications which run in heterogeneous environments including the cloud and the edge. To solve these issues, the open-source Quartz [6] framework, centered on the notion of QoT and timelines, provides a micro-service-based containerized architecture to provide Time-as-a-Service to geo-distributed applications. However, the notion of providing both time and location as first-class primitives is novel, and will enable flexible and adaptive geo-scale CPS.

5 CONCLUSION

Emerging CPS increasingly coordinate at geo-distributed scale. Hence, a distributed cyber-physical OS is essential to enable the rapid development, deployment and life-cycle management of these applications. Since, cyber-physical coordination occurs in both space and time, it is key to expose both as first-class application primitives along with their associated quality metrics. We posit that the efforts required to realize such a framework, and provide a core-set of essential services, creates multiple avenues for research.

REFERENCES

[1] [n. d.]. Docker: Enterprise Container Platform. https://www.docker.com/.

[2] [n. d.]. Kubernetes: Container Orchestration. https://kubernetes.io/.

[3] Jason Allnutt, Dhananjay Anand, Douglas Arnold, Allen Goldstein, Ya-Shian Li-Baboud, Aaron Martin, Cuong Nguyen, Robert Noseworthy, Ravi Subramaniam, and Marc Weiss. [n. d.]. Timing Challenges in the Smart Grid. NIST Special Publication 1500-08.

[4] Fatima Anwar, Sandeep D'souza, Andrew Symington, Adwait Dongare, Ragunathan Rajkumar, Anthony Rowe, and Mani Srivastava. 2016. Timeline: An Operating System Abstraction for Time-Aware Applications. In *Real-Time Systems Symposium (RTSS), 2016 IEEE*. IEEE, 191–202.

[5] Patricia Derler, Thomas H Feng, Edward A Lee, Slobodan Matic, Hiren D Patel, Yang Zheo, and Jia Zou. 2008. *PTIDES: A programming model for distributed real-time embedded systems.* Technical Report. University of California Berkeley.

[6] Sandeep D'souza, Heiko Koehler, Akhilesh Joshi, Satyam Vaghani, and Ragunathan (Raj) Rajkumar. [n. d.]. Quartz: Time-as-a-Service for Coordination in Geo-Distributed Systems. https://bitbucket.org/sandeepdsouza93/quartz/.

[7] Sandeep D'Souza and Ragunathan Rajkumar. 2018. QuartzV: Bringing Quality of Time to Virtual Machines. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 49–61.

[8] Sandeep D'souza and Ragunathan (Raj) Rajkumar. 2017. Time-based Coordination in Geo-Distributed Cyber-Physical Systems. In *9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*. USENIX Association, Santa Clara, CA.

[9] Hermann Kopetz and Günther Bauer. 2003. The time-triggered architecture. *Proc. IEEE* 91, 1 (2003), 112–126.

[10] Barbara Liskov. 1991. Practical Uses of Synchronized Clocks in Distributed Systems. In *Symposium on Principles of Distributed Computing (PODC '91)*. 1–9.

[11] Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. 2015. Edge analytics in the internet of things. *IEEE Pervasive Computing* 2 (2015), 24–31.

[12] Da Zhang, Feng Xia, Zhuo Yang, Lin Yao, and Wenhong Zhao. 2010. Localization technologies for indoor human tracking. In *Future Information Technology (FutureTech), 2010 5th International Conference on*. IEEE, 1–6.