Time-based Coordination in Geo-Distributed Cyber-Physical Systems

Sandeep D'souza and Ragunathan (Raj) Rajkumar
Carnegie Mellon University

Abstract

Emerging Cyber-Physical Systems (CPS) such as connected vehicles and smart cities span large geographical areas. These systems are increasingly distributed and interconnected. Hence, a hierarchy of cloudlet and cloud deployments will be key to enable scaling, while simultaneously hosting the *intelligence* behind these systems. Given that CPS applications are often safety-critical, existing techniques focus on reducing latency to provide *real-time* performance. While low latency is useful, a shared and precise notion of time is key to enabling coordinated action in distributed CPS. In this position paper, we argue for a global *Quality of Time* (QoT)-based architecture, centered around a shared *virtualized* notion of time, based on the *timeline* abstraction [1]. Our architecture allows applications to specify their QoT requirements, while exposing timing uncertainty to the application. The *timeline* abstraction with the associated knowledge of QoT enables scalable geo-distributed coordination in CPS, while providing avenues for fault tolerance and graceful degradation in the face of adversity.

1 Introduction

Cyber-Physical Systems (CPS) [2] involve the *cyber* components of computing and communication interacting with and controlling elements in the *physical* world. Emerging CPS are connected to the Internet, and consist of multiple networked *sensing*, *actuation* and *computational* components spanning large geographical areas. These systems range from small-scale multi-robot systems [3][4][5] to city-scale connected vehicles and planetary-scale collaborative augmented reality [6]. The scale of these systems makes the cloud well-suited for enabling coordination among multiple smaller entities [8].

The future holds promise for CPS with possibly even inter-planetary-scale coordination. Consider the recently launched Breakthrough Starshot Initiative [9]. The initiative proposes to send laser-propelled nano-spacecraft to Alpha Centauri. While there are a number of engineering challenges still to be solved, one can envision a global network of laser arrays being used to send such nano-spacecraft into deep space. To propel fleets of nano-spacecrafts, it will be essential to precisely coordinate the direction and intensity of these geographically-distributed laser arrays, while taking into account the effects of the earth's rotation and atmospheric interference.

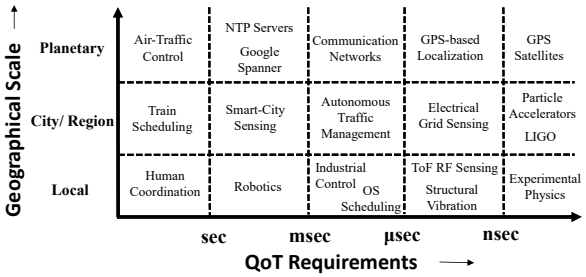

Figure 1: The scale of coordination in time and space

Reliable planetary-scale coordination requires a shared and precise notion of time [1]. However, existing cloud-enabled CPS design methodologies [10] focus on reducing the latency of computation, and overlook the importance of shared time. While technologies likes GPS and the Precision Time Protocol [11] have enabled networked devices to share a precise notion of time, trends like networking delays [12], multi-core processors [13] and virtualization [14] introduce greater timing uncertainty. This uncertainty is rarely visible to applications, and most systems rely on best-effort time synchronization.

In [1], we introduced the concept of *Quality of Time* (QoT) as the end-to-end uncertainty in the notion of time delivered to an application by the system. Building on the notion of QoT, the QoT Architecture [1], centered around a shared *virtualized* notion of time, allows applications to specify their timing requirements, while delivering the required QoT and exposing timing uncertainty to applications. We argue that the knowledge of QoT enables applications to adapt and be fault-tolerant, while allowing the system to manage resources efficiently.

Figure 1 shows CPS applications with diverse QoT requirements, spanning different geographical scales. Hence, we posit that there is a strong need for a QoT-based CPS framework for distributed coordination at scale, with APIs based on a shared notion of time.

The contributions of this paper are as follows:

1. We argue for a global shared-time architecture to enable local to planetary-scale coordination in CPS.
2. We illustrate the utility of Quality of Time in providing fault tolerance and reliable performance in geo-distributed and coordinated CPS.
3. We discuss the challenges in extending our timeline-based QoT Architecture in the context of enabling reliable cloud-based coordination at scale in CPS.

We build on prior work [1], and focus on the requirements and architectural choices which would enable cloud-

based scalable coordination in geo-distributed CPS, using a shared notion of time along with the associated QoT.

2 Background

In this section, we review the trends in cloud-enabled CPS, and describe the QoT Architecture proposed in [1].

2.1 Cyber-Physical Systems and the Cloud

Embedded platforms enable the deployment of computation and data-intensive CPS applications in resource-constrained environments. These platforms range from low-power micro-controllers [15][16] and sensor nodes [17][18] to powerful multi-core platforms [19][20][21].

While these platforms in conjunction with available software enable rapid prototyping and deployment, they are not sufficient to enable the deployment of complex applications at large geographical scale. This is especially true when coordinated action must be performed by multiple distributed nodes [3]. To achieve coordination at scale, it is essential that an intelligent controller, which may be centralized or distributed, dictate coordination policies and perform the required orchestration.

The emergence of the public cloud has made possible the deployment of elastically-scalable software services. Public cloud [22][23] providers such as Amazon, Microsoft and Google offer a range of *virtualized* computing services, thus lowering the entry bar for innovation in the software services domain. A similar trend is emerging for the Internet of Things (IoT) [24], with major cloud service providers offering specialized services for IoT [25][26].

Most commercially available frameworks focus on directly connecting sensing/actuation endpoints to the Internet [25][26]. Given that the number of such endpoints is poised to grow faster than network bandwidth, this approach is inherently not scalable [27]. Rather, application functionality needs to be partitioned among different components. Additionally, most CPS are safety-critical, and require low-latency computation [2]. In the mobile computing domain, *cloudlets* [7] have been proposed to provide high-performance low-latency computation. Cloudlets are trusted, resource-rich computational clusters that are in close physical proximity to the mobile user [7]. The low-latency requirements of CPS make the use of a hierarchy of cloudlets and the cloud useful [10]. Additionally, deploying cloudlets at the edge reduces the upstream bandwidth demand at the cloud [28].

While low latency is critical for real-time performance, very few guarantees on latency can be provided in the Internet [7]. On the other hand, advances in clock synchronization [11] have made it possible to provide precise time over a network with good reliability [29]. Programming applications based on a shared notion of time enables distributed components to specify timing constraints in terms of the shared notion of time, enabling fine-grained coordination with fewer messages exchanged [30].

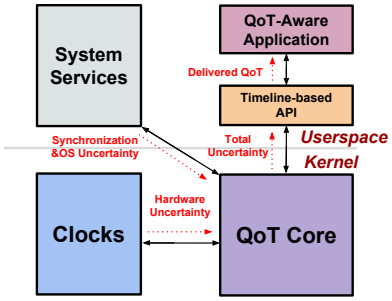

Figure 2: Timeline-based QoT Architecture [1]

2.2 Timeline-based QoT Architecture

As proposed in [1], a timeline is a *virtual* reference time base that is not necessarily tied to a specific reference device or time system. This abstraction enables developers to implement coordinated applications easily. Consider an application that needs to perform coordinated actions by its distributed components. All of these components bind to a common timeline, each specifying its respective QoT requirements. The *timeline* abstraction [1] abstracts away low-level synchronization details, while the underlying framework orchestrates the clock-synchronization services and clocks to ensure that QoT requirements are met, while making the actually achieved QoT visible to the application. This is fundamentally different from existing best-effort clock-synchronization techniques [11][12]. Application-specified QoT requirements open up the possibility of network and system orchestration to ensure that application requirements are met, while managing resources efficiently.

The QoT Architecture supports multiple timelines. This enables different coordinating sub-groups with varying QoT requirements to each have their own virtual time reference and co-exist on the same infrastructure [1]. The QoT Architecture consists of three distinct components:

1. Clocks are used to expose timekeeping hardware, and provide timekeeping and time-stamping capabilities. Clocks also expose their parameters such as accuracy, precision and drift, which enable uncertainty calculations.

2. System Services are responsible for distributing timeline meta-data, message passing, quantifying timing uncertainties, and synchronizing clocks across nodes.

3. QoT Core acts as a bridge between all the system components, applications and the operating system. It is responsible for application scheduling as well as maintaining synchronization and timeline state.

The architectural components present on each node along with their interactions are illustrated in Figure 2. Based on this architecture, we developed a prototype QoT Stack for Linux [1], which focused on implementing necessary functionality over a Local-Area Network (LAN). The existing version of our stack supports the Beaglebone Black [21] embedded platform, with clock-synchronization support for PTP [11] and NTP [12] over

Ethernet. The stack consists of kernel modules and system services, and does not modify the Linux kernel.

3 Coordination in Cyber-Physical Systems

This section outlines the key concerns to be addressed from the standpoint of enabling coordination in CPS.

1. Scalability: As illustrated in Figure 1, many emerging CPS applications will scale large geographical areas. Cisco predicts that the number of connected sensing/actuation endpoints will exceed 50 billion by 2020 [31]. We believe that CPS frameworks which support coordinated action among a large number of geo-distributed components, will enable new classes of applications.

2. Fault Tolerance and Reliability: Most CPS applications are safety-critical [2], making fault tolerance necessary. In the software services domain, fault tolerance is concerned with reducing down time, and preventing information loss [32]. Hence, most services are replicated across different fault-tolerance domains. Most CPS also utilize replication techniques to ensure fault tolerance. However, CPS interact with the real world, where the safety of humans and infrastructure is critical. Therefore, CPS may also rely on *analytical redundancy* [33], involving graceful degradation modes. When multiple components fail, the system must be able to gracefully degrade and stop without causing any harm [34].

3. Security: The presence of malicious nodes can severely impact local behavior and global coordination, which can have consequences in safety-critical applications. As compared to software services, which are hosted in a secure data center, many CPS have physical nodes deployed in public spaces. Hence, malicious nodes need to be detected and isolated from the coordination subgroup, without violating safety constraints.

4. Ease of Programmability: In the software services domain, the availability of cloud APIs enables developers to easily deploy and elastically scale applications in the cloud. Similarly, the emergence of frameworks like Map-Reduce [35] and Spark [36] have enabled scalable parallel computation over clusters in the cloud. In the CPS domain, enabling coordination across heterogeneous platforms, ranging from sensor networks to the cloud, is a challenge. Hence, there is a need for a framework, which is easy to use, and has expressive power to support coordinated CPS from local to planetary scales.

4 The Case for Shared Time and QoT

We now argue for designing coordinated CPS using a shared notion of time with the associated knowledge of QoT. In distributed software systems, a shared notion of time enables increased performance and better coordination, along with decreasing the number of messages which need to be exchanged [30]. However, there are inherent uncertainties associated with synchronizing clocks over a network. Hence, in [30], Liskov reasons that systems should rely on clock synchronization for performance but not correctness. This is true for most software systems. For example, reducing timing uncertainty decreases the transaction commit wait in Spanner, leading to better performance [29]. However, in CPS, the uncertainty tolerances are dictated by the application and the environment. If the required QoT cannot be met, then the application should be aware of it, and gracefully degrade to satisfy safety and reliability requirements.

We highlight the benefits of coordination using a shared notion of time by presenting an emerging CPS application utilizing an *idealized* solution called *TimeNet*. Subsequently, we present the practical challenges in enabling scalable coordination in CPS using shared time and QoT.

4.1 Connected Vehicles using *TimeNet*

Coordinating fleets of connected autonomous vehicles for city-wide dynamic traffic management is an example of a geo-distributed application which can benefit from using a shared notion of time. The proposed application hierarchy is illustrated in Figure 3, and consists of autonomous vehicles, *Vehicle-to-Infrastructure* (V2I) nodes [40], cloudlets and the cloud.

In an ideal world, we can assume that all components of this application are connected to a network which provides instantaneous access to an *ideal* source of time with no associated uncertainty. For the sake of simplicity, let's call this hypothetical network *TimeNet*. Let's assume that *TimeNet* can be used to perfectly time-stamp all events and messages with zero uncertainty. Hence, using *TimeNet*, a unique total ordering on all events can be derived.

In the context of our application, the infrastructure nodes can precisely measure the location of the vehicles, along with the exact timestamp associated with a vehicle's presence at that location. This timestamped information can be then forwarded to a nearby cloudlet, which receives state information from multiple infrastructure nodes in a small geographical area. Multiple such cloudlets can then forward their respective state information to the cloud, which sits atop the application hierarchy. In this hierarchy, the cloud is responsible for shaping traffic flow at a macroscopic level. Based on the macroscopic policy, the cloudlets make local decisions for their respective regions. Lastly, infrastructure nodes decide microscopic traffic policy and convey instructions to the autonomous vehicles, which implement these instructions.

In an ideal world, accurate information can be inferred from these timestamped events, which can be used to formulate plans of action, such that vehicles coordinate their actions using this ideal notion of time. Thus, vehicular traffic is dynamically managed at city scale. In the worst case, if timing constraints are violated or messages delayed, then by using the current time, components can

detect failures, and take corrective action [30].

Unfortunately, a perfect source of time does not exist, and practical systems introduce uncertainty in timing measurements. Hence, to determine the validity of timestamps, the knowledge of its associated uncertainty is essential. Based on this uncertainty information, coordination policies can order events with different degrees of confidence. If the uncertainty exceeds tolerable limits, systems can fail-over or gracefully degrade. For example, in the context of the dynamic traffic management application, if the uncertainty exceeds tolerable limits, the coordination policy can instruct all or some vehicles to temporarily change their speeds, or come to a safe halt.

Exposing the notion of QoT to applications also allows timing requirements to be explicitly specified. This enables the system to optimize for application QoT requirements, and manage resources efficiently. Hence, in the context of CPS, synchronized clocks along with QoT can deliver both performance and reliability.

The present-day GPS is a close approximation to *TimeNet*, ideally providing synchronization in the order of tens of nanoseconds. However, GPS is not accesible indoors and inaccurate in urban canyons [37]. This limits its use in many applications [38]. Hence, a practical realization of *TimeNet* may involve multiple outdoor GPS receivers equipped with chip-scale atomic clocks [39]. These receivers can distribute accurate time to subscribers both wirelessly and over the Internet [12]. To support the notion of QoT, it is crucial that each node in *TimeNet* quantify the uncertainty in its notion of time.

4.2 A QoT-based CPS-Cloud Architecture

We now address the various challenges involved in designing a *practical* extension of our QoT Architecture, in the context of using a hierarchy of cloudlets and the cloud as an enabler of scalable CPS coordination.

1. Global Timeline Service: The *timeline* abstraction is key to forming different scalable coordination subgroups on the same infrastructure. Additionally, a timeline enables different coordinating components of an application to specify their QoT requirements and subscribe to a shared notion of time. These different components may be running on nodes that are interconnected by heterogeneous networking technology. For example, in the dynamic traffic management application shown in Figure 3, the vehicles and the infrastructure nodes may be connected by a Dedicated Short-Range Communications (DSRC) V2I network [40]. On the other hand, the infrastructure nodes, cloudlets and the cloud are connected by the Internet (using 4G/5G, Wi-Fi or Ethernet). Hence, there is a need for a global timeline service, which provides a common protocol to coordinate these components interconnected by heterogeneous technologies.

2. Scalable Synchronization Service: In CPS, it

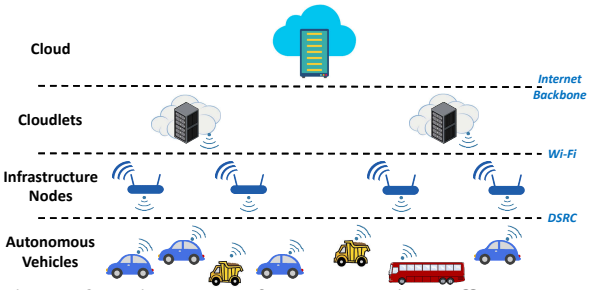

Figure 3: Hierarchy of the dynamic traffic management application

is common to find platforms ranging from ARM-based micro-controllers to server-grade x86 machines. These platforms have different resource constraints and capabilities. Hence, while most edge devices use wireless networking technologies like Wi-Fi, BLE [41], UWB [42] and Zigbee [43], gateways and cloudlets have both wireless and wired links, and servers in the cloud mostly rely on wired technologies like Ethernet. The plethora of these networking technologies implies that multiple clock synchronization protocols, such as NTP [12], PTP [11], RBS [44] and FTSP [45], might be in use within the same distributed application. Additionally, distributed application components will have different QoT requirements. For example, tight clock synchronization is required among the sensing/actuation endpoints, while computational components (cloudlets and the cloud) have less stringent QoT requirements. These factors make maintaining a shared notion of time, as well as calculating and transferring timing uncertainty information across these multiple networking domains, challenging. Given the heterogeneity in applications, platforms, networks and protocols, there is a need for a service which orchestrates the system to ensure that QoT guarantees are met.

3. Fault-Tolerance Support: The notion of QoT will facilitate fault tolerance. Hence, it is crucial that each coordinating component maintain its own notion of QoT, based on the synchronization and system uncertainties. Thus, during network or synchronization outages, each component can calculate the delivered QoT, using previously computed worst-case uncertainty estimates [1]. When the calculated QoT exceeds the specified tolerance, or timing constraints are violated, application components can fail-over to a replica or gracefully degrade.

4. Pub/Sub Messaging: The publish/subscribe paradigm has emerged as the method of choice for communicating among coordinating nodes [46], and is commonly used for coordination in many real-world applications [34][47]. Additionally, publish/subscribe technologies support a variety of communication media, and have been shown to scale. Hence, in our architecture, we allow all components bound to a timeline to publish their messages to all other nodes. Individual nodes can subscribe to subsets of messages based on content or type.

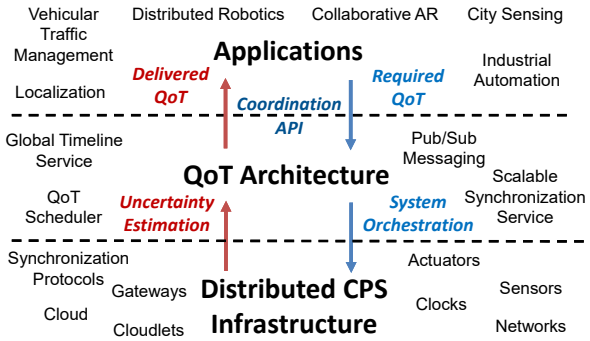

Figure 4: A QoT-based CPS-Cloud Architecture

5. Virtualization Support: Most clouds and cloudlets provide *virtualized* units of computing, which may be Virtual Machines (VMs) or containers [48]. Virtualization adds additional timing uncertainty due to relatively high jitter in clock-read and timer-interrupt latencies [14]. Thus, scheduling computations precisely on global time in a virtualized environment is challenging. Additionally, the possibility of VM/container migration raises the question of storing synchronization state in the hypervisor or the guest [14]. Hence, the use of virtualization in the cloud presents a challenge in terms of observing and guaranteeing the QoT delivered to an application.

6. QoT-Aware Cloud Scheduling: The presence of multiple virtualized units running applications with different QoT requirements adds an additional layer of complexity to the QoT-aware cloud scheduling problem. We envision our QoT scheduler working in conjunction with existing multi-level scheduling frameworks [49][50]. Hence, there is a need to provide probabilistic QoT-based Service Level Agreements (SLAs). Additionally, as indicated in the case of Spanner [29], modern data centers may have access to a GPS receiver for precise timing. We believe that placing a VM or container on a host in proximity to this receiver would enable a higher level of QoT. Hence, the QoT requirements of applications would dictate the host to which they are allocated.

7. Security: Given that our architecture involves coordination between sets of nodes, we envision using public-key based authentication [51]. Only nodes with appropriate public keys can join the timeline. Additionally, messages can be encrypted using this key.

8. Coordination APIs: To enable distributed coordination at scale, it is essential to have a core set of APIs that are independent of the platform and OS. At the same time, the APIs should be extensible to support platform-specific extensions. The APIs enable applications to (i) bind/unbind from a timeline, (ii) specify/update their QoT requirements, (iii) schedule computation, sensing and actuation based on shared time, (iv) timestamp events, and (v) pub/sub messaging for coordination. All API calls return the QoT actually delivered to the application, providing the ability to adapt to changes in QoT.

Figure 4 illustrates how the QoT Architecture can enable a host of coordinated CPS applications, running on distributed heterogeneous networked infrastructure.

5 Related Work

Most of the existing work in the literature on cloud-enabled CPS, focus on reducing latency, by bringing resource-rich computing closer to the edge of the network [7][10][27][52]. All these frameworks focus on proper partitioning of functionality among different tiers of nodes, ranging from the cloud to cloudlets, gateways and edge devices. We believe that using a shared notion of time along with the knowledge of QoT is complementary to these techniques. While low latency is crucial to achieve real-time performance, using shared time provides scalability, while the knowledge of QoT provides applications the ability to adapt, and be fault-tolerant.

The idea of using a shared notion of time in distributed systems is not new. In [30], Liskov analyzed the performance benefits of using clock synchronization in many distributed algorithms. In the embedded domain, PTIDES [53] provides a framework to model, design and deploy time-critical embedded applications, using a shared notion of time. However, these prior work do not consider the utility of the knowledge of timing uncertainty, and rely on best-effort clock synchronization. While Google Spanner utilizes uncertainty information for achieving global-scale consistency and performance [29], the use of uncertainty as an enabler of performance, scalability and fault tolerance has not been explored.

The use of cloudlets and the cloud entail the use of *virtualized* computing units which introduce additional timing uncertainty. In [14], Broomhead et al. experimentally characterized the timekeeping properties of the Xen paravirtualization platform. To the best of our knowledge, no similar study exists in the context of hardware-accelerated virtualization and container-based frameworks. Additionally, scheduling virtualized units based on application QoT requirements needs to be explored.

6 Conclusion

Deploying Cyber-Physical Systems at scale is of increasing interest to researchers. In this position paper we highlight the merits of designing scalable coordinated CPS using a shared notion of time, along with the associated knowledge of Quality of Time. We identify the challenges involved in designing a practical QoT-based CPS framework, and believe that by incorporating the proposed design choices, our stack can support cloudlets and the cloud to achieve geo-distributed CPS coordination at scale. Additionally, we believe that our architecture would also be useful for distributed software systems [29] which can reap scalability and better performance using shared time.

Discussion

Our position paper argues for coordination in cloud-enabled geo-distributed Cyber-Physical Systems using a shared notion of time, along with the associated idea of Quality of Time. Recent trends in networking hardware and clock synchronization, coupled with the availability of chip-scale atomic clocks and low-cost good-quality oscillators, have enabled distributed nodes to precisely synchronize their clocks. Hence, we posit that the use of shared-time with the added notion of QoT, will enable planetary-scale coordination, while providing additional avenues for application-level fault tolerance. This approach is fundamentally different from traditional techniques, which mostly rely on low-latency message passing to achieve coordination. We believe that this is of significant interest to the research community.

We also illustrate the design choices needed to realize a practical global shared-time based framework, to support coordination in CPS, ranging from local to planetary scales. This paper does not delve into the intricate details of our proposed framework, and focuses on the high-level requirements needed to achieve scalable coordination. Future work includes overcoming multiple challenges related to QoT-based cloud scheduling and system orchestration. Additionally, another area of future research is the possibility of our framework being useful for enabling better performance and scalability in software systems.

Acknowledgements

The authors would like to thank Fatima Anwar, Andrew Symington, Adwait Dongare, Anthony Rowe and Mani Srivastava for their efforts on the QoT Stack for Linux. This research is funded in part by the National Science Foundation under award CNS-1329644. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, or the U.S. Government.

References

[1] F. Anwar, S. D'souza, A. Symington, A. Dongare, R. Rajkumar, A. Rowe and M. Srivastava, "Timeline: An Operating System Abstraction for Time-Aware Applications", in Proc. of *IEEE Real-Time Systems Symposium*, 2016

[2] R. Rajkumar, I. Lee, L. Sha and J. Stankovic, "Cyber-Physical Systems: The Next Computing Revolution", in Proc. of *Design Automation Conference*, 2010

[3] B. G. Regula, "Formation control of a large group of UAVs with safe path planning", in Proc. of *IEEE Mediterranean Conference on Control & Automation*, 2013.

[4] J. Enright and P. Wurman, "Optimization and Coordinated Autonomy in Mobile Fulfillment Systems", In Proc. of *AAAI Workshop*, 2011

[5] O. Givehchi, H. Trsek, J. Jasperneite, "Cloud computing for industrial automation systems — A comprehensive overview", in Proc. of *IEEE Conference on Emerging Technologies & Factory Automation*, 2013

[6] S. Natarajan and A. Ganz, "SURGNET: An Integrated Surgical Data Transmission System for Telesurgery", in *International Journal of Telemedicine and Applications*, Article ID 435849, 2009

[7] M. Satyanarayanan et al., "The Case for VM-Based Cloudlets in Mobile Computing", in *IEEE Pervasive Computing*, Vol. 8, Issue: 4, Oct.-Dec. 2009

[8] E. Simmons et al., "A Vision of Cyber-Physical Cloud Computing for Smart Networked Systems", in *NIST Interagency/Internal Report (NISTIR) - 7951*, 2013

[9] Starshot Breakthrough Initiative, *https://breakthroughinitiatives.org/Initiative/3*

[10] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog computing and its role in the Internet of Things", in Proc. of the *MCC workshop on Mobile cloud computing*, 2012

[11] K. Lee, J. C. Eidson, H. Weibel, and D. Mohl, "IEEE 1588-standard for a precision clock synchronization protocol for networked measurement and control systems", in *IEEE Instrumentation and Measurement Society Standard*, 2005

[12] D. L. Mills, "Internet time synchronization: the network time protocol," in *IEEE Transactions on Communication*, vol. 39, no. 10, 1991.

[13] M. Kuperberg and R. Reussner, "Analysing the fidelity of measurements performed with hardware performance counters," in Proc. of the *ACM/SPEC International Conference on Performance engineering*, 2011

[14] T. Broomhead, L. Cremean, J. Ridoux, and D. Veitch, "Virtualize everything but time", in Proc. of *OSDI*, 2010

[15] NXP Kinetis K22, *http://www.nxp.com/assets/documents/data/en/fact-sheets/KINK2XFS.pdf*

[16] Arduino, *https://www.arduino.cc/*

[17] A. Rowe, R. Mangharam and R. Rajkumar, "FireFly: A Time Synchronized Real-Time Sensor Networking Platform", *http://nanork.org/attachments/148/nrk-chapter06.pdf*

[18] P. Levis, "Experiences from a Decade of TinyOS Development", in Proc. of *OSDI*, 2012

[19] ODROID XU-4, *http://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf*

[20] Raspberry Pi 3, *https://www.raspberrypi.org/products/raspberry-pi-3-model-b/*

[21] Beaglebone Black, *https://beagleboard.org/black*

[22] "Making Infrastructure-as-a-Service in the Enterprise a Reality", *http://www.oracle.com/us/products/enterprise-manager/infrastructure-as-a-service-wp-1575856.pdf*

[23] "Platform-as-a-Service",*http://www.intel.com/content /dam/www/public/us/en/documents/white-papers/cloud-computing-paas-cloud-demand-paper.pdf*

[24] "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything", *http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov /IoT_IBSG_0411FINAL.pdf*

[25] AWS Internet of Things, *https://aws.amazon.com/iot*

[26] Azure IoT Suite, *https://www.microsoft.com/en-us/cloud-platform/internet-of-things-azure-iot-suite*

[27] B. Zhang et al., "The Cloud is Not Enough: Saving IoT from the Cloud", in Proc. of *HotCloud*, 2015

[28] P. Simoens et. al, "Scalable Crowd-Sourcing of Video from Mobile Devices", CMU-CS-12-147 Tech Report, 2012

[29] J. C. Corbett et al., "Spanner: Google's globally distributed database", in *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, 2013

[30] B. Liskov, "Practical Uses of Synchronized Clocks in Distributed Systems", in Proc. of *ACM symposium on Principles of distributed computing*, 1991

[31] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything", *CISCO white paper*, 2011

[32] F. Schneider, "Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial", in *ACM Computing Surveys*, Vol. 22, No. 4, 1990

[33] J. Gertler, "Analytical Redundancy Methods in Fault Detection and Isolation", in *Preprints of IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 1991

[34] J. Wei et al., "Towards a Viable Autonomous Driving Research Platform", in Proc. of *IEEE Intelligent Vehicles Symposium*, 2013.

[35] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", in Prov. of *OSDI*, 2004

[36] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica, "Spark: Cluster Computing with Working Sets", in Proc. of *HotCloud*, 2010

[37] Z. Tan, L. Zhu and L. Zhong, "Vision: Cloud and Crowd Assistance for GPS Urban Canyons", in Proc. of *ACM MCS*, 2014

[38] N. Rajagopal, S. Chayapathy, B. Sinopoli and A. Rowe, "Beacon Placement for Range-Based Indoor Localization", in Proc. of *International Conference on Indoor Positioning and Indoor Navigation*, 2016

[39] Quantum SA.45s Chip Scale Atomic Clock, *https://www.microsemi.com/products/timing-synchronization-systems/embedded-timing-solutions/components/sa-45s-chip-scale-atomic-clock*

[40] SAE J2735 Standard,*https://ntl.bts.gov/lib/51000/51100 /51167/DE156ECC.pdf*

[41] "Introduction to Bluetooth Low Energy", *https://cdn-learn.adafruit.com/downloads/pdf/introduction-to-bluetooth-low-energy.pdf*

[42] "Comparison of Narrowband and Ultra Wideband Channels", *Decawave White Paper*, 2008

[43] "Zigbee: Setting Standards for Energy-Efficient Control Networks", *Schneider Electric White Paper*, 2011

[44] J. Elson, L. Girod and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts", in Proc. of *OSDI*, 2002

[45] M. Maroti, B. Kusy, B. Simon and A. Ledeczi, "The Flooding Time Synchronization Protocol", in Proc. of *ACM SenSys*, 2004

[46] OMG Data Distribution Service Standard, *http://www.omg.org/spec/DDS/*

[47] "Delivering High-Performance, Scalable and Safe Data Distribution in Next Generation Air Traffic Control and Management", *PrismTech White Paper*, 2010

[48] W. Felter et al., "An Updated Performance Comparison of Virtual Machines and Linux Containers", *IBM Technical Report*, 2014

[49] B. Hindman et al., "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center", in Proc. of *NSDI*, 2011

[50] V. Vavilapalli et al., "Apache Hadoop YARN: yet another resource negotiator", in Proc. of the *Annual Symposium on Cloud Computing*, 2013

[51] N. Ferguson and B. Schneier, "Practical Cryptography", *Wiley*, 2003

[52] "Intel Architecture at the Edge for Greater Flexibility and Scalability", *http://www.intel.com/content/dam/www/public/ us/en/documents/solution-briefs/communicationsintel-architecture-brief.pdf*, 2011.

[53] J. Zou, S. Matic, E. Lee, T. Feng and P. Derler, "Execution Strategies for PTIDES, a Programming Model for Distributed Embedded Systems", in Proc. of the *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009