The Integration of Scheduling, Monitoring, and SLA in Cyber Physical Systems

Awatif Alqahtani, Khaled Alwasel, Ayman Noor, Karan Mitra, Ellis Solaiman, and Rajiv Ranjan

1 Introduction

Cyber-physical systems (CPS) are new engineering structures that involve interdisciplinary system components and a human interaction in order to link platforms to the physical world for higher productivity, optimal decision-making, lesser operational costs, controllable environment, etc. [1]. CPS integrates platforms, applications, computation systems, communication systems, devices, and sensors/actuators to build a new generation of smart environments, such as smart city, smart grid, and smart industry. In a general sense, CPS harnesses the power of Internet of Things, computing paradigms (public datacenters, private datacenters, and/or edge datacenters), and the Internet to represent and control the actual physical environments (Fig. 1). For example, a smart city can embed thousands of sensors and IoT devices such as the Spanish smart city of Santander, which has deployed more than 20,000 sensors, to measure everything ranging from trash containers, to parking spaces, to air pollution, to smart traffic management [2].

A. Alqahtani
Newcastle University, Newcastle upon Tyne, UK

King Saud University, Riyadh, Saudi Arabia

K. Alwasel · E. Solaiman · R. Ranjan
Newcastle University, Newcastle upon Tyne, UK

A. Noor (✉)
Newcastle University, Newcastle upon Tyne, UK

Taibah University, Medina, Saudi Arabia

K. Mitra
Department of Computer Science Electrical and Space Engineering Luleå University of Technology Skellefteå, Sweden

R. Ranjan et al. (eds.), *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*, Scalable Computing and Communications,
https://doi.org/10.1007/978-3-030-43795-4_10

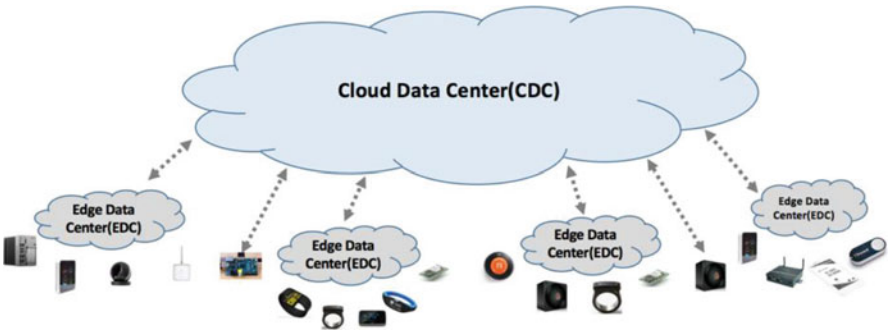

Fig. 1 Cyber-physical systems (CPS) architecture

Internet of Things (IoT) is an integral part of CPS providing the ability to capture and improve real world environments by using certain objects to sense, process, and communicate based on particular conditions and configurations [3]. Smart homes, smart transportation, and smart industry are a few examples of IoT environments, which consist of different types of things such as sensors, actuators, smartphones, etc. Each IoT environment produces a large amount of real-time data that needs to be analyzed, stored, and delivered to intended receivers [4, 5]. However, performing intensive computation and communication operations on IoT devices/environments is not easy, if not impossible, due to the limited capabilities of hardware and software [3]. To overcome such limitation, IoT devices traditionally delegate most of their computing and storing operations to Cloud datacenters [1, 3].

The advances in both IoT and Cloud technologies have led to a Big Data paradigm in order to handle the large amount of data generated from various resources (social networking, IoT devices, etc.) with diverse speed, volume, and type [6]. Big Data is derived due to the needs of discovering and finding specific information of a massive amount of data that would be so valuable for individuals, organizations, and governments. Prior Big data, traditional processing frameworks (e.g., MySQL and Oracle) had been used where they failed to handle a large amount of IoT data [6]. Therefore, big data programming frameworks and models (e.g., Hadoop, Storm, MapReduce, etc.) have been introduced to deal with IoT data integration and batch/stream big data analytics.

Traditionally, IoT has utilized Cloud computing as the main infrastructure for its data operations, such as computation, filtering, and storing [1, 3]. However, many papers [4, 5, 7] had argued that the exhaustive migration of IoT data operations to the Cloud alone is not sufficient due to many reasons. First, IoT delay-sensitive applications (e.g., natural disaster monitors) cannot only depend on the Cloud because of the unpredicted data transfer time of the Internet, known as best-effort data delivery. Second, cloud datacenters operate on geo-distributed manners, resulting in unstable and unpredictable decision-making time. Third, IoT data might be useless and such data would result in the waste of resources. Therefore, Edge computing has emerged to allow distinct computing operations to be executed at or closer to data sources, resulting in optimal decision-making and saving of resources.

Figure 1 represents the architecture of CPS. In IoT environments, devices (e.g., sensors, actuators, cameras, cars, etc.) sense, capture, and send the behaviors of the physical world as raw data to Edge Data Centers (EDCs) and/or Cloud Data Centers (CDCs) for further processing. EDCs and CDCs perform computational and analytical operations (e.g., filtering, analyzing, detecting, etc.) on the received data in order to make automatable actions on physical environments and ultimately forward visualized results to end-users. EDC contains a small-scale datacenter to perform lightweight tasks, often on IoT streaming data in order to foster decision-making. In contrast, CDC consists of large-scale distributed datacenters to perform intensive tasks on historical and real-time data, if needed.

Despite the great achievements of CPS' individual components, integrating distinct technologies and platforms in CPS remains challenging; *Scheduling, Monitoring, and End-to-End SLA* (SMeSLA) in CPS are still open issues, which need to be investigated to cope CPS performance deficiencies. Thus, the objective of this chapter is to identify major challenges and issues CPS should address especially towards (1) developing a dynamic multi-level CPS scheduler, (2) deploying an accurate and fine-grained monitoring system, (3) implementing end-to-end Service Level Agreement (SLA) mechanisms.

1.1 Motivation Example

Consider a city has deployed hundreds of sensors and cameras for early flood detection/predication by analyzing sensed data and images as well as comparing current water level with the historical water level (Fig. 2). The city council subscribed to different cloud service providers to detect potential flood within an optimal time, which consequently minimizes damages such as an early evacuation of citizens to safer zones. To perform flood monitoring, compute- and data-intensive operations must be executed by leveraging the power of CPS according to the following steps:

1. Smart gateways should forward sensed data generated from different types of sensors (e.g., water level and gauge sensors) to available EDCs while forwarding images to available CDCs for further processing due to the need of massive computation resources;
2. EDCs run computational frameworks (e.g., Strom and/or Hadoop) in a distributed fashion to filter sensed data based on given threshold values and then forward filtered data to CDCs for further processing;
3. CDCs then start analyzing received data from EDCs to detect rain and flood levels by comparing the pattern of data with the pattern of historical data. CDCs then combine the analyzed results to make an appropriate action, such as:

   (a) If the analysis shows a low possibility of a flood, then one of possible action is to change sending data interval time from a pre-defined time (e.g., every 60 s) to real-time and notify interested parties.

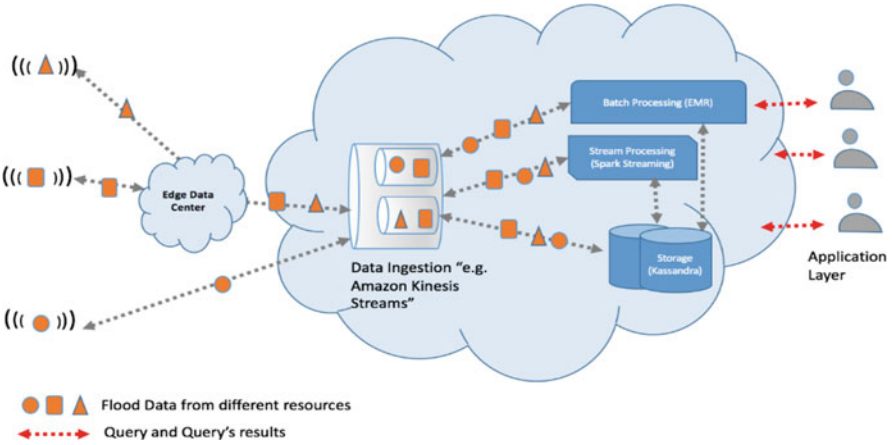

Fig. 2 Data flow in flood monitoring (Use Case)

(b) If the analysis shows a medium/high possibility of a flood, then one of possible action is to notify interested parties, such as city council, as well as send commands to smart gateways to change data sampling rate, sending data interval time, and update actuators to turn on siren alarms so people start evacuation.

From the above-mentioned scenario, having a cooperation among scheduling, monitoring, and SLA-management components on end-to-end basis (cross and within CPS components) is essential. SLA specifies when, where, and what to consider, scheduler translates SLA constraints at run time for provisioning purposes while monitoring keeps track of resources states and notify scheduler and SLA whenever there is a violation (Fig. 2).

2 Cyber-Physical Systems (CPS) Architecture

IoT revolution has been utilized in many fields, such as industries, government, and health-care, which has led to the needs of leveraging Cloud computing resources due Cloud's unbounded capabilities. The centralized architecture of Cloud computing plays an important role in its success in terms of economic perspectives; yet, considering a logical extreme scenario of a full centralisation approach could result in unexpected drawbacks. In [8], authors mentioned four fundamental issues of centralized approaches. First, there is a need to make a trade-off between releasing personal and sensitive data to centralized services (e.g., social networks, location services, etc.) and privacy. Second, utilizing cloud services allows only a one-sided trust from clients to the Clouds, while there is no trust from one client to another. Third, Cloud providers neglect the fact that new generations of Edge

devices are embedded with high computational capacity and sufficient storage space. Last, Cloud-based centralisation hinders human-centered designs, which limits the interactions between humans and machines. Thus, moving computations to the Edge under certain conditions will minimize the Cloud-based centralisation issues as well as take the advantages of Edge devices' capabilities. Figure 1 illustrates the data flow in CPS, which, generally, consists of the following layers:

A. **Sensing/actuating layer**

Represents devices (e.g., sensors, actuators, cameras, smart mobiles, etc.) that are used to sense, capture, and send the behaviors of the physical world as raw data to EDCs and/or CDCs for further processing.

B. **Edge computing layer**

EDCs are small-scale datacenters used to perform lightweight-computational and -analytical operations (e.g., filtering, analyzing, detecting, etc.) on the received-IoT data to improve the performance, save unnecessary data transfers, accelerate decision-making, and make automatable actions on physical environments. EDCs are more secured and private compared with CDCs, in which sensitive data can be processed ad stored more properly [8].

C. **Cloud computing layer**

CDCs consists of large-scale distributed datacenters to perform intensive tasks on historical and real-time data, if needed. CDCs mainly consists of infrastructure hardware and Big Data frameworks:

- **Big Data layer:** this layer deals with a massive volume of data generated from various resources at different rates. It consists of the following components [6]:
  - Data ingestion: accepts data from multiple sources, such as online services and back-end system logs.
  - Data analytics: consists of many platforms (e.g., stream/batch processing frameworks, machine learning frameworks, etc.) to ease the implementation of data analytics operations.
  - Data storage: to store intermediate and final datasets. The ingestion and analytics layers make use of different databases during execution.
- **Cloud Infrastructure layer:** provides consumers with different capabilities, such as processing of data, access to networks, and store of data. It also enables end-users to run arbitrary software, such as applications and operating systems [8].

D. **CPS Management layer**

CPS consists of multi-layers, multi-environments, and multi-users where a new management layer must be integrated into its overall architecture (Fig. 3). CPS components and layers depend on each other (an output from one layer might become an input to another layer), which makes CPS management a daunting task.

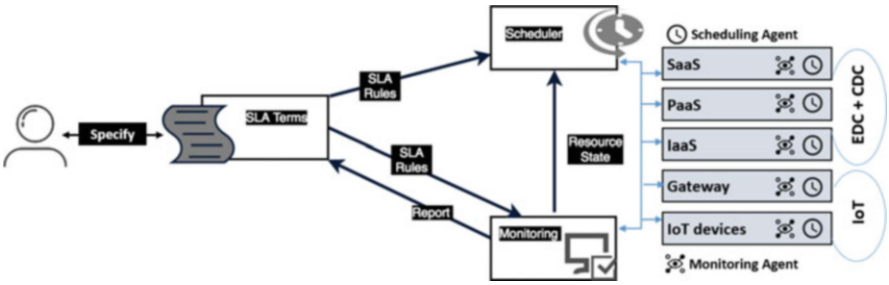

Fig. 3 Conceptual interaction between scheduling, and monitoring, and SLA managers/agents (SMeSLA) in CPS

For example, real-time data can be forwarded to the ingestion layer using Kafka framework. Data can be then sent to the stream processing layer to be analyzed and stored on the fly. Further, the batch processing layer can execute Hadoop framework to process data received from the stream processing layer. Last, machine-learning frameworks can be executed to identify similarities between Hadoop's generated data and historical data in order to detect/predict interesting events.

Moreover, the management layer empowers CPS system to meet consumers' satisfaction and achieve optimal performance (e.g., saving energy, cost reduction, etc.). The management layer can also ensure the correctness of SMeSLA, including system components, algorithms, layers, Quality of Services (QoSs), etc. The interaction among SLA, scheduling, and monitoring managers leads to an effective management mechanism (Fig. 3). For instance, end-users can specify their QoS preferences (e.g., sensor data rate, processing latency of gateways, etc.) and submit them to the SLA manager. As a result, the SLA manager will translate and submit the preferences to the scheduling and monitoring managers to be deployed and reported, respectively. Last, by deploying an agent within each layer, layer status and commands can be reported and executed on behave of its respective manager. Figure 3 reflects the conceptual interaction between SLA, scheduling and monitoring managers.

3 Studies Related to Scheduling, Monitoring, and End-to-End SLA (SMeSLA) in CPS

This section represents related-work that had been conducted toward Scheduling, Monitoring, and SLA in CPS.

3.1 Scheduling in CPS

According to the a forementioned motivation section, CPS scheduling becomes a substantial factor, in which the real-time adaptation can be achieved/deployed. CPS scheduling can also handle issues including, but not limited to, resource contention, performance enhancement, optimal resource utilization, and cost reduction. As CPS consists of IoT, CDC, EDC, environments, developing a multi-objective smart scheduler is essential, which allows the deployment of real-time scheduling policies and algorithms according to users' and administrators' QoS objectives within (intra) and across (inter) environments. However, building a CPS scheduler is so complex due to (1) the complex interlinked among cyberspace components (e.g., IoT devices, gateways, network and computational devices, frameworks/applications, EDC, CDC, etc.); (2) cyber-physical space (e.g., movement, sensor random deployment, etc.); and (3) different CPS environments (flood monitoring systems, traffic management systems, etc.).

On the topic of CPS scheduling, most of the existing works mainly focus on specific layers/applications in an individual datacenter. For example, authors in [9–11] have contributed and proposed many CDC scheduling algorithms (e.g., dynamic priority, deadline, and delay-sensitive) to enhance the performance of big data applications/frameworks, such as Hadoop. However, they only consider server-level utilization and discard network-level utilization, which leads to issues such as flow congestion, network overload, packet loss, etc. By leveraging Software-Defined Networking technology (SDN), authors in [12–15] developed scheduling algorithms in accordance with server and network levels, resulting in optimal job completion time in terms of computation-intensive and/or bandwidth-intensive jobs. Moreover, authors in [16–18] proposed SDN-IoT architectures where sensing intelligence are decoupled from sensor's physical hardware and placed on SDN controller(s), providing application-customizable, a flexibility of policy changes, and ease of management.

There are also recent studies that have explored and tried to make scheduling decisions in a global fashion. For instance, authors in [7] practically extended a Storm framework to operate in a geographically spread out environment (Cloud and Edge), outperforming the Storm centralized-default scheduler. Moreover, many gateway types, schemes, and designs have been proposed in many papers. For example, authors in [19] proposed a smart gateway container-based virtualization (e.g., Docker) that integrates networking, computation, and database functions to allow users to configure IoT computation modes and schedule data analytical tasks. Authors in [20] proposed a scheme where SDN controllers act as smart gateways, yet providing full control of IoT network infrastructure. However, all the a forementioned scheduling contributions do not target a CPS infrastructure, but rather toward one or two layers/environments.

3.2 Monitoring in CPS

Monitoring is an essential element to consider in the CPS architecture to detect improper behaviors of one or more CPS components, layers, and environments as well as respond to such behaviors accordingly [21, 38]. There are numerous commercial and open source monitoring tools/frameworks (e.g., Monitis, Nimsoft, LogicMonitor, CloudWatch, etc.), which are used for different purposes, such as tracking the utilization of virtual machines' CPU and memory. However, those tools cannot be deployed in CPS because of the diverse components, layers, and environments CPS spans. For instance, Amazon CloudWatch is a cloud monitoring tool used only in Amazon datacenters to monitor hardware-level whereas LogicMonitor is a commercial tool used to perform monitoring in application-level.

3.3 SLA in CPS

As scheduling and monitoring are important aspects for CPS to operate efficiently, a service-level agreement (SLA) is an important element to consider in CPS in order to assure that consumers' quality of service (QoS) requirements are observed. SLA plays a significant role in specifying the required level of quality at which a service should be delivered [22, 39]. SLA has been used in many IT-related fields and platforms over many years [23]. For example, Web Service Level Agreement (WSLA) was introduced in 2003, which is a framework composed of SLA specifications and a number of SLA monitoring tools for web services [24]. In [25], authors proposed web service agreement (WS-Agreement), which defines the specification of web service agreement as a domain-specific language. However, WSLA and WS-Agreement are specific for web services, and cannot be directly applied to CPS systems.

There are a number of works related to SLA in Cloud computing. For example, the European and National projects [26] developed a Blueprint concept where SLA specification is presented as a descriptive document to express the service dependencies across Cloud layers as well as within each layer. As well, Blueprint can also define provisioning and management rules related to elasticity and multi-tenancy. In [27], another SLA specification project so-called SLA@SOI was developed to address multi-level multi-provider SLA lifecycle management within service-oriented architecture and Cloud computing. It provides an abstract syntax for describing functional and non-functional characteristics of a service. However, the available SLA frameworks are either being too specific or too generic, while in CPS systems there is a need to aggregate QoS requirements from the perspectives of IoT, EDCs, CDCs environments.

Traditional SLAs that focus on availability and reliability are not enough for CPS applications due to the requirement of strict SLA guarantees [28]. Therefore, specifying contractual terms of SLA on an end-to-end basis is important, not only

to specify end-to-end QoS requirements but to develop end-to-end SLA-aware scheduling and monitoring algorithms to assure consumers that their quality of service (QoS) requirements will be observed across computing environments in order to deliver services that match consumer expectations, such as to complete a required job with maximum latency equal to x time units. Delivering applications which rely on SLA-aware services, such as SLA-aware resource allocation and SLA-aware monitoring, will minimise the risk of violating the terms of the service agreement, especially for real-time applications which require stable factors in order to operate properly. For example, Flood Monitoring System applications (FMS) must respond immediately and correctly to suspicious events in order to prevent serious damage. FMS requires collecting real-time data from different types of information, such as from sensors and gauges that measure rainfall-levels and water level of rivers, respectively. FMS would then analyse collected data and indicate any abnormal data patterns (e.g. flood possibility) by comparing new collected data with historical/stored data. However, this type of IoT application is time-sensitive, which means any unpredicted delay in one or more of the data flow stages (e.g. collecting, transferring, ingesting, analysing, etc.) will affect the accuracy and suitability of the actions taken. This example shows how the performance of FMS applications relies not only on the functionality but also on the quality of offered services across Edge or/and Cloud computing environments. Undoubtedly, SLAs need to be observed across all layers of Cloud and Edge, for example: at which rate data should be collected, transferred and ingested, how fast and accurate the analysis should be, etc.

Having an individual SLA management mechanism for each layer of CPS is inadequate because of the huge dependency across layers [29]. Within the SLA, there is a need to express such constraints/policies which determine when and which data can be processed within the Edge data centers as well as when and which data need to be exported to be processed/analysed in Cloud data centers under certain time limitations. Back to flood monitoring scenario to explain the dependency among its required services across computation layers: Data from rain gauges and water level sensors can be analysed within edge layer. If one or both of the readings exceed the threshold value, then edge layer can make a decision, such as to increase the sampling rate for rain gauges, river level sensors and bridge cameras. This will increase the incoming data from edge devices. Therefore, based on the computation capabilities of edge layer, some data can be analysed within Edge, such as if water level measurement exceeds the *threaten* threshold, which has been pre-calculated and specified based on previous experiences reflecting high possibility of flooding, and then edge layer can send immediate notification and alert interested parties/destinations. On the other hand, if measured data has not exceeded the *threaten* threshold but it exceeds a specific threshold, then a combination of water level, rain level as well as captured images need to be exported to Cloud for two reasons, but not limited to these reasons. Firstly, to be analysed and compared with historical data for early flood prediction purposes and then send an appropriate command back to the edge layer such as change sample rate. For example, change sample rate from a periodical sensing to instant one, i.e. rain fall and water level

sensing and bridge picturing are performed instantly. Secondly, to be saved as historical data if a flood has occurred to be used as data training to develop/enhance a predictive model of flood prediction application for future usage.

From this scenario, there is a need to specify when data need to flow within and across layers and under which speed using an end-to-end SLA. Furthermore, there is a need to build a cross-layer multi-provider SLA-based monitoring system for the CPS to enhance SLA compliance. This will aid service providers to operate their services at an adequate level, which then will increase consumers' trust, and also help to avoid SLA violations.

4 SMeSLA Technical-Research Challenges in CPS

The following subsections represent research challenges and facts SMeSLA encounter in CPS. In order to successfully deploy SMeSLA, end-users should be able to specify QoS preferences by means of SLA terms in every point of the following subsection. The SLA manager should then submit those preferences to the scheduling and monitoring managers to operate upon (Fig. 3):

4.1 Sensing Devices

IoT devices (e.g., sensors) must communicate with one another and with gateways in order to send their sensing data to end-receivers. However, the diverse characteristics of wireless and wired gateways' interfaces, the movement of IoT devices (fixed or mobile), and properties of IoT devices (e.g., lifetime, coverage, etc.) impact on latency, bandwidth, and speed of data. For example, sensors often come with low-level electronic interfaces (e.g., I2C, SPI, 6LowPAN, ZigBee, etc.) to communicate with one another and with gateways. Selecting the appropriate interface is an essential factor that determines the efficiency of IoT environment. For instance, Inter-Integrated Circuit (I2C) is a wired gateway interface allowing sensors to send data up to 3.4 Mbit/s in its fastest mode whereas in a typical mode it is limited to 400 kbit/s for most cases [30]. In contrast, Serial Peripheral Interface (SPI) is more efficient in terms of power consumption and full-duplex communication, leading to energy saving and a higher throughput.

Nevertheless, wired communications are inefficient due to their difficulty of deployments, regular maintenance, higher costs, etc. Therefore, wireless gateway interfaces are more practical in IoT environments, such as 6LowPAN and ZigBee. Ipv6 over Low-Power Wireless Personal Area Networks (6LowPAN) enables IPv6 packets to be transferred within a small link-layer frame, has a transfer rate of up to 250 kbit/s for a distance of 200 m, and supports up to 100 nodes for every network. On the other hand, ZigBee is a preferable wireless standard because of its low-cost and low-power consumption. It has a transfer rate of up 250 kbit/s for a distance of

100 m and it supports a network of 100 nodes [30]. However, since wireless allows the mobility of devices, it might affect data link availability and reliability such as message loss, data inaccuracy, and higher latency.

4.2 Gateways

Typically, gateways link IoT devices to their intended EDC and/or CDC environments where data can be further processed, stored and analyzed. IoT devices can operate without gateways if they have the ability to communicate directly to the Internet. However, many IoT devices (e.g., sensors) have minimal functionalities; therefore, they outsource the missing functionalities to gateways. Gateways, the new generation of routers, are integrated with rich functionalities such as traffic management, local database storage, data aggregation, and data analyzing [30, 31]. When specifying QoS for any IoT application, consumers must indicate the types of IoT environments in their SLAs in terms of with or without gateways. There is trade-off between with and without gateway deployments, which can influence on overall delays.

4.3 Big Data Analytics Tools

Since IoT environments often generate extremely large raw data, they often rely on big data frameworks and computational models to accelerate decision-making and lesser response time. Every IoT application requires different big data analytical ecosystems (e.g., Kafka, Hadoop, Storm, etc.), heterogeneous big data workflows (static, streaming, etc.), and different QoS objectives (low cost, certain latency, etc.). The following are some challenges that should be considered in the overall development of SMeSLA:

Batch Processing: Apache Hadoop[1] and Apache Spark[2] are a few big data batch-processing tools, which deploy a MapReduce programming model and are used to process, analyze, and visualize data. End-users should be able to select the preferred batch processing frameworks based on their needs and objectives. For example, some users might prefer Hadoop due to its low cost (less use of memory and network) while they are time-tolerant in terms of processing finishing time (up to minutes, hours, or days) [32]. In contrast, other users might prioritize Spark to perform their data analyses for machine learning due to its fast processing mechanism, which is up to 100 time faster than Hadoop [3]. In addition, there

[1] http://hadoop.apache.org/

[2] http://spark.apache.org/

are other factors that need to be considered, including data size, number of map tasks and reducing tasks, the diverse architecture of HDFS (e.g., the number of data nodes, the number of resources assigned to each node, and the replication number), and the type of machine learning algorithms (e.g., Mahout) [33].

Distributed databases: Storing and handling a massive amount of data (real-time or archived) requires powerful storage platforms. They are different types of storage platforms with distinct capabilities. For example, Hadoop Distributed File System (HDFS) can be used to store data in a distributed manner, but it does not ensure a high level of data management (e.g., storing, accessing, querying, etc.) [3]. On the other hand, Druid,[3] open-source distributed databases, can support real-time data ingestion and query with low latency.

Stream processing: Real-time stream applications perform operations on stream data generated from different sources where the time of results should be in a fraction of a second. Stream processing is crucial for real-time applications and critical systems (e.g., disaster management) that require real-time decision-making. Spark Streaming[4] is a stream and batch analytics tool, which provides low-cost but it might produce some delay. Apache Storm[5] is another stream processing tool outperforming Spark, only in terms of delay [3].

Distributed queues: It is used to ingest data from different sources and distribute the data to interested parties/components. Some of available open source messaging queues are Apache Kafka[6] and RabbitMQ.[7] Kafka, which is based on a publish/subscribe model, provides high throughput and low latency. In contrast, RabbitMQ is more flexible and provides packet-loss guarantees by means of message acknowledgment mechanisms, but it has less throughput and higher latency [3].

Considering the a forementioned subsections in the overall SMeSLA-CPS development process is important to meet consumers' satisfaction (e.g., big data framework types, latency in each layer, etc.). It can also provide CPS optimal performance (e.g., saving energy, increasing profits, etc.). In fact, every subsection has some effects on SLA terms (e.g., query response time), which in return will affect CPS scheduling and monitoring.

[3] http://druid.io/

[4] https://spark.apache.org/streaming/

[5] https://storm.apache.org/

[6] http://kafka.apache.org/

[7] https://www.rabbitmq.com/

5 SMeSLA General-Research Challenges in CPS

Besides technical challenges, there are general challenges that need to be considered in the development of SMeSLA.

5.1 The Heterogeneity and Random Distribution of IoT Devices

Various sectors deploy IoT technologies for delivering smart services by spreading sensors everywhere within their target environments. The heterogeneity of IoT devices and sensors in terms of capabilities, network protocols, applications, and vendors along with the distinct nature of every IoT environment makes the deployment of SMeSLA a challenging task. For example, administrators should be able to dynamically configure sensing modes (e.g., transmission time and data rate) and change activity mode (on/off) as an individual or group to save energy, reduce costs, etc. Moreover, in the motivation example, administrators should be able to control cameras that are in zones where a flood might occur should send full HD videos while cameras in unexpected zones should send non-HD videos to avoid network congestions, save power, etc.

There are also other factors that make the deployment of SMeSLA very challenging. First, every cluster of sensors has different manufacturing features, such as lightweight functionalities in temperature sensors compared with human body detection sensors that require persisting data along with sophisticated security mechanisms [30]. Second, the physical distance between IoT sensors and where data are analyzed, processed, and stored, vary from one case to another. Third, since IoT devices are often equipped with low capabilities, querying such devices/sensors for monitoring purposes cannot be easily achieved because they do not provide querying functionalities, such as querying memory utilization of sensors.

5.2 Lack of Standardization

As the architectures of CPS consists of different EDC and CDC providers, it raises a serious standardization problem. There is a need for standardizing the terminologies of service level objectives (SLOs) as well as QoS functions, which specify how QoS metrics are being measured. In CDCs, for example, there are no standard vocabularies to express SLAs – take availability as an example and how it is expressed differently among well-known Cloud providers: Amazon EC2 offers availability as a monthly uptime percentage of 99.95%, Azure offers a monthly connectivity uptime service level of 99.95%, and GoGrid offers a 100% server uptime and a 100% uptime of the internal network [34]. In contrast, EDCs describe

the rate at which sensors send data in many terms, such as sampling rate [35] or sampling frequency [36]. Indeed, unifying terminologies and metrics as well as proposing a taxonomy will lead to a well-designed SMeSLA; which in turn would provide a successful interaction between consumers and providers and minimize the amount of time required to write and negotiate SLAs.

5.3 Heterogeneity of Key QoS Metrics Across CPS Environments

Understanding key performance metrics and their variation in CPS environments and within their layers is crucial. In other words, there is a need for building a coherent taxonomy that considers various QoS metrics among CPS's layers while data is flowing. There are different QoS metrics for each layer [35, 40]:

- Cloud environment:
  (a) SaaS: contains event detection and decision-making delays.
  (b) PaaS: includes QoS of big data frameworks (e.g. throughput and response time).
  (c) IaaS: includes QoS of infrastructure layer (CPU utilization, memory utilisation, network latency, network bandwidth, etc.).
- Edge environment:
  (a) It has similar layers and QoS metrics as Cloud environments.
- IoT environment:
  (a) Perception layer: includes data quality, precession, and freshness.
  (b) Network layer: includes latency, throughput, and availability.

5.4 Heterogeneity of Application Requirements

Every IoT application has specific requirements according to its predefined purpose and domain-specific application. For instance, traffic applications have a high-priority for data accuracy while environmental prediction applications have a high-priority for data accuracy and action response time. The heterogeneity of high-level application requirements certainly hinders the development process of IoT applications-SMeSLA-aware.

5.5 Lack of Methods for Collecting QoS Metrics

The nature of CPS technology requires the interaction of cross-computing and cross-layers delivered by different providers. Each component should have its own SLA to clearly specify QoS capabilities, which monitoring and scheduling should operate upon. One of the main challenges is how to collect and integrate metrics from those different providers in order to monitor end-to-end SLAs at an application level, without the necessity for understanding the complex format of components/platforms [37].

5.6 Selection of Datacentres

IoT computation tasks (e.g., filtration, aggregation, analyzing, etc.) can be performed in EDCs or/and CDCs. Determining the computation environment of IoT raw data is not easy due to many factors. First, users' QoS properties (e.g., higher security, priorities, completion time, etc.) play important roles in determining the preferable computation environments. Second, IoT devices might generate useless data where the importance of data should be indicated in early stages; therefore, the selected type of datacenters could have high impacts on computational and network costs. Third, smart gateways might face traffic bursting (e.g., all sensors continuously send data), which makes gateways a single point of failure.

6 Design Goals of SMeSLA in CPS

Developing a CPS software platform that provides SMeSLA is essential to enable performance optimization and prediction, prevent failures, and meet QoS expectations. The platform should provide the following attributes:

(a) *Connectivity & Interoperability:* every device in CPS (e.g., sensors, gateways, SDN controllers, etc.) should be connected and accessed using remote interfaces/middleware where CPS platform can dynamically adapt algorithms and policies as well track behaviors and SLA in a local and global manner. By deploying the notion of virtualization (e.g., application, data, server, network, etc.), CPS-SMeSLA infrastructure can be easily developed, managed, and instrumented while real-time responsiveness in terms of dynamic adaptation and reconfiguration can be achieved.

(b) *Availability:* every EDC and CDC should advertise its available resources (e.g., applications, VMs, networks, etc.) along with the actual utilization in real-time. This feature allows scheduling decisions to be made in a global fashion while achieving optimal performance in accordance with different QoS needs (e.g.,

network priority of video applications compared with computation-priority of machine learning applications).

(c) *Scalability:* as the number of IoT, EDC, and CDC devices are increasing to accommodate new configuration and deployment requirements, having a scalable CPS-SMeSLA infrastructure is required to handle the growing number of devices, data, and requests without losing its overall performance. Moreover, CPS-SMeSLA infrastructure should be able to address varying loads that might occur in IoT, EDC, and CDC environments.

7 Conclusion

In this chapter, we discussed the benefits and challenges of deploying real-time Scheduling, Monitoring, and End-to-End SLA (SMeSLA) in Cyber-Physical Systems (CPS). CPS is a very complex system where a new management layer based on SMeSLA must be developed. We proposed an SMeSLA conceptual architecture where end-users submit their QoSs to an SLA manager; as a result, scheduling and monitoring managers would operate accordingly. Every layer of CPS must deploy scheduling and monitoring agents in order to enforce policies/changes and keep track of CPS in a global manner. In order to successfully deploy SMeSLA in CPS, many technical and general challenges must be addressed such as the heterogeneity of IoT devices, gateways, and big data, lack of standardization, etc.

References

1. R. Ranjan et al., Cyber-physical-social clouds: Future insights. IEEE Tech. Comm. Cyber-Phys. Syst. **1**(1), 11–14 (2016)
2. L. Sanchez et al., SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities. Futur. Netw. Mob. Summit (FutureNetw), 1–8 (2011)
3. M. Díaz, C. Martín, B. Rubio, State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. J. Netw. Comput. Appl. **67**, 99–117 (2016)
4. M. Villari, O. Rana, Osmotic computing: A new paradigm for edge/cloud integration. IEEE Cloud Comput. **3**(6), 76–83 (2016)
5. M. Nardelli, S. Nastic, M. Villari, Osmotic flow: Osmotic computing + IoT workflow. IEEE Cloud Comput. **4**(2), 68–75 (2017)
6. R. Ranjan, Streaming big data processing in datacenter clouds. IEEE Cloud Comput. **1**(1), 78–83 (2014)
7. V. Cardellini, V. Grassi, F. Lo Presti, M. Nardelli, Distributed QoS-aware scheduling in storm, in *Proceedings of the 9th ACM international conference on Distributed Event-Based – DEBS '15*, (2015), pp. 344–347
8. P. Garcia Lopez et al., Edge-centric computing: Vision and challenges. ACM SIGCOMM Comput. Commun. Rev. **45**(5), 37–42 (2015)
9. A. Verma, L. Cherkasova, R.H. Campbell, {ARIA:} automatic resource inference and allocation for mapreduce environments, in *Proceedings of the 8th ACM International Conference on Autonomic Computing {ICAC} 2011, Karlsruhe, Ger. June 14–18, 2011*, (2011), pp. 235–244

10. M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, I. Stoica, Delay scheduling, in *Proceedings of the 5th European Conference on Computer System – EuroSys'10*, (2010), p. 265
11. J. Tian, Z. Qian, M. Dong, S. Lu, FairShare: Dynamic max-min fairness bandwidth allocation in datacenters, in *2016 IEEE Trustcom/BigDataSE/ISPA*, (IEEE, 2016)
12. P. Qin, B. Dai, B. Huang, G. Xu, Bandwidth-aware scheduling with SDN in hadoop: A new trend for big data. IEEE Syst. J. **11**(99), 1–8 (2015)
13. C.X. Cai, S. Saeed, I. Gupta, R.H. Campbell, F. Le, Phurti: Application and network-aware flow scheduling for multi-tenant MapReduce clusters, in *Proceedings of the. – 2016 IEEE International Conference on Cloud Engineering IC2E 2016 Co-located with 1st IEEE International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, (2016), pp. 161–170
14. L.W. Cheng, S.Y. Wang, Application-aware SDN routing for big data networking, in *2015s IEEE Global Communications Conference (GLOBECOM 2015)*, (2016)
15. B. Peng, M. Hosseini, Z. Hong, R. Farivar, R. Campbell, R-storm: Resource-aware scheduling in storm, in *Proceedings of the 16th Annual Middleware Conference – Middleware. '15*, (2015), pp. 149–161
16. Ethics form completed for project: Exploiting software – Defined networking to enhance performance of big data programming models on cloud datacentres, p. 2019, (2017)
17. T. Luo, H.P. Tan, T.Q.S. Quek, Sensor openflow: Enabling software-defined wireless sensor networks. IEEE Commun. Lett. **16**(11), 1896–1899 (2012)
18. Y. Zhu, F. Yan, Y. Zhang, R. Zhang, L. Shen, SDN-based anchor scheduling scheme for localization in heterogeneous WSNs. IEEE Commun. Lett. **7798**(c), 1–1 (2017)
19. R. Petrolo, R. Morabito, V. Loscrì, N. Mitton, The design of the gateway for the Cloud of Things. Ann. Telecommun. **72**, 1–10 (2016)
20. O. Salman, I. Elhajj, A. Kayssi, A. Chehab, Edge computing enabling the Internet of Things, in *IEEE world forum Internet Things, WF-IoT 2015 – Proceedings*, (2016), pp. 603–608
21. K. Alhamazani et al., An overview of the commercial cloud monitoring tools: Research dimensions, design issues, and state-of-the-art. Computing **97**(4), 357–377 (2015)
22. G. Gaillard, D. Barthel, F. Theoleyre, and F. Valois, "SLA specification for IoT operation – The WSN-SLA Framework Research Report – INRIA," 2014
23. P. Bianco, G. a Lewis, and P. Merson, "Service Level Agreements in Service-Oriented Architecture Environments.," no. September, p. 42p, 2008
24. A. Keller, H. Ludwig, The WSLA Framework: Specifying and monitoring service level agreements for web services. J. Netw. Syst. Manag. **11**(1), 57–81 (2003)
25. A. Andrieux et al., Web services agreement specification (WS-Agreement). Glob. Grid Forum GRAAP-WG **192**, 1–80 (2004)
26. M.P. Papazoglou, W.J. Van Den Heuvel, Blueprinting the cloud. IEEE Internet Comput. **15**(6), 74–79 (2011)
27. E. Commission, S.L. Agreement, M. Framework, *Project Goals*, pp. 1–2
28. M. Wang, R. Ranjan, P.P. Jayaraman, P. Strazdins, P. Burnap, O. Rana, D. Georgakopulos, A case for understanding end-to-end performance of topic detection and tracking based big data applications in the cloud, in *EAI International Conference on cloud, Networking for IoT Systems, Roma, Italy*, (2015)
29. A. Alqahtani, E. Solaiman, R. Buyya, R. Ranjan, End-to-end QoS specification and monitoring in the internet of things, in *Newsletter, IEEE Technical Committee on Cybernetics for Cyber-Physical Systems, vol. 1, no. 2*, (2016)
30. R. Buyya, A. V. Dastjerdi, Internet of things.
31. A.M. Rahmani et al., Smart e-Health Gateway: Bringing intelligence to Internet-of-Things based ubiquitous healthcare systems, in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC 2015)*, (2015), pp. 826–834
32. E. Solaiman, R. Ranjan, P.P. Jayaraman, K. Mitra, Monitoring internet of things application ecosystems for failure. IT Prof. **18**(5), 8–10 (2016)

33. R. Giaffreda, D. Cagáňová, Y. Li, R. Riggio, *Internet of Things. IoT Infrastructures*, vol 1 (Springer, Cham, 2015), pp. 427–438
34. F. Alkandari, R.F. Paige, Modelling and comparing cloud computing service level agreements, in *InternatIonal workshop on Model-drIven Engineering for High Performance and Cloud Computing – MDHPCL'12*, (2012), pp. 1–6
35. P.P. Jayaraman, K. Mitra, S. Saguna, T. Shah, D. Georgakopoulos, R. Ranjan, Orchestrating quality of service in the cloud of things ecosystem, in *Proceedings of the – 2015 IEEE International Symposium on Nanoelectronic and Information System iNIS 2015*, (2016), pp. 185–190
36. X. Liu, Q. Wang, L. Sha, W. He, Optimal QoS sampling frequency assignment for real-time wireless sensor networks, in *In RTSS*, (2003)
37. X. Zheng, P. Martin, K. Brohman, L. Da Xu, Cloud service negotiation in internet of things environment: A mixed approach. IEEE Trans. Ind. Inform. **10**(2), 1506–1515 (2014)
38. A. Noor, D.N. Jha, K. Mirta, P.P. Jayaraman, A. Souza, R. Ranjan, S. Dustdar, A framework for monitoring microservice-oriented cloud applications in heterogeneous virtualization environments, in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, (IEEE, 2019), pp. 156–163
39. A. Alzubaidi, E. Solaiman, P. Patel, K. Mitra, Blockchain-based SLA Management in the Context of IoT. IT Prof. Mag. **21**(4), 33–40. ISSN 1520-9202, E-ISSN 1941-045X
40. A. Noor, K. Mitra, E. Solaiman, A. Souza, D.N. Jha, U. Demirbaga, P.P. Jayaraman, N. Cacho, R. Ranjan, Cyber-physical application monitoring across multiple clouds. Computers & Electrical Engineering **77**, 314–324 (2019)