Scalable Modeling of Cloud-based IoT Services for Smart Cities

Amir Taherkordi and Frank Eliassen
Department of Informatics
University of Oslo, Norway
{amirhost,frank}@ifi.uio.no

Abstract—The convergence of Internet of Things (IoT) and the Cloud has significantly facilitated the provision and management of services in large-scale applications such as smart cities. With a huge number of IoT services accessible through clouds, it is very important to model and expose cloud-based IoT services in a scalable manner, promising easy and realtime delivery of smart city services. The existing work in this area does not sufficiently address this design issue due to the adoption of a uniform and flat view to the structure of IoT services and their data in the Cloud. In this paper, we propose a framework for scalable and realtime provisioning of cloud-based IoT services in large-scale applications, such as smart cities. These two features are achieved by structuring the description of IoT services in a hierarchical model and populating them in a tree structure containing references to services and their realtime data. Such a service access structure can be obtained based on logical or contextual scopes of a service. Using this approach, smart city applications can access IoT services and subscribe to their realtime data in an scalable manner at different contextual levels, e.g., from a municipal district to a street.

I. INTRODUCTION

The convergence of the Internet of Things (IoT) and cloud computing paradigms is essentially triggered by large-scale IoT applications, such as smart cities. Such applications leverage the scalability, performance and pay-as-you-go capabilities of the Cloud and compensate their technological constraints (e.g., storage, processing, energy) [1], [2]. From the cloud perspective, the Cloud can benefit from IoT systems by extending its functionality and delivering new services in large-scale real life scenarios, such as in smart city applications [3].

During recent years, several efforts have been made towards the convergence of IoT systems and the Cloud, both in the research community [4] and in industry (e.g., [5], [6], [7]). A common characteristic of these efforts is their ability to stream data to the Cloud in a scalable and high performance manner, in addition to providing the means for managing applications and data streams. These efforts are essentially more beneficial and make more sense for smart city applications with a large number of IoT devices, spread possibly in vast geographical areas. In such a setting, the Cloud will provide a unified and integrated platform for fast, scalable and efficient development of end-user applications.

From a smart city service provision perspective, IoT devices constantly make their services accessible at the city cloud level (i.e., based on a push model and/or a pull model) [8]. On the other end of the system, interested third-party smart city applications (e.g., traffic monitoring systems) will make use of these services based on protocols for service discovery and access, and their preferences in terms of quality of service requirements. Among several challenges arising in this scenario, there exist two critical challenges that should be carefully addressed. First, with the presence of millions of smart and intermittent devices in cities, a crucial need is a scalable way to model and expose IoT services at the Cloud which promises easy and fast delivery of smart city services. Second, the event-driven nature of services in IoT systems requires a mechanism for the provision of services in realtime. For instance, in a disaster recovery system, instant changes in the environmental condition (e.g., traffic and weather sensors) should be propagated to the interested applications without any delay. Addressing these challenges is not trivial in large-scale cloud-based smart city applications.

The existing work, in this area, has mainly focused on the management of IoT resources in the Cloud. Moreover, in existing frameworks, the way to structure and access IoT services in the Cloud is either not sufficiently studied [9], [10] or only focused on semantic processing of data sets [11] and their relations [12]. Providing a scalable service access model is of particular importance with respect to the huge number of IoT devices integrated to cloud computing platforms, e.g., in the case of smart cities.

In this paper, we propose a generic IoT service access model, which is specifically designed for large-scale cloud-based IoT applications, such as smart cities. The essence of our approach is structuring the description of IoT services in a hierarchical model and populating them in a tree structure containing references to services and their realtime data entries. The way to define hierarchies of service description is based on either the logical and contextual scopes of a service (e.g., domain.physicalProperty.location.iotdevice.aResource) or the semantical description of a service. This adheres to the way that IoT devices in smart cities are organized. Using this model, the cloud-based tree ensures scalable and fast service provisioning. More importantly, in order to guarantee realtime and fresh service delivery to interested parties, the service tree supports notification-based access to service data and changes. The preliminary
implementation of the proposed framework is performed over Firebase—a cloud-based, real-time data storage platform.

The rest of this paper is organized as follows. The principles of service design in IoT cloud integration is presented in Section II. In Section III, we demonstrate a motivating smart city application, which is a basis for the framework proposed in Section IV. Then, we present the related work in Section V and conclude in Section VI.

II. MODELING CLOUD-BASED IoT SERVICES

Prior to going through the detail of our proposal for scalable and real-time modeling of IoT cloud services, it is important to clarify the design space that we define as a basis for our work. In the context of this work, we aim to highlight key modeling elements of IoT cloud services that are fundamental and relevant for scalable and real-time service processing. It should be noted that these aspects are partly inspired from the efforts made so far by the IoT research community on IoT cloud service design challenges [13], [14] and associated use-cases [15]. Figure 1 shows a service-based view of the IoT cloud integration model, inspired by the fact that today’s IoT services are often RESTful and act as a wrapper for IoT resources. With this assumption, the following design concerns should be taken into account:

- **IoT Resource Management in the Cloud:** The mechanism to register and maintain the list of available IoT resources is the key requirement, in particular with the presence of a huge number of IoT devices integrated into clouds, e.g., in smart city applications. This should also be empowered with an efficient discovery solution for end-user applications.

- **IoT Service Delivery and Processing:** The service delivery mechanism should be empowered with an efficient model of propagating event-based services and routing them to interested services or applications. Beyond that, by enabling cloud-based IoT services delivery, the scalable storage capabilities of clouds can be utilized to analyze both big data produced by IoT devices and maintain a configurable history of service data, e.g., a history of temperature changes in a given district.

- **Service Integration:** An important insight into the cloud IoT service landscape is to maintain a record of all connected service instances, which are executed at run-time to build composed services. The logic behind the creation and the management of complex services can be expressed in terms of, e.g., business processes, defined through standard languages such as the Business Process Execution Language (BPEL) [13]. Similar to the Service Delivery, the integration service should support event-based interactions.

In order to put our work in context and motivate the approach proposed in Section IV, in the following we highlight the associated challenges in large-scale cloud-based IoT services design.

![Fig. 1: An overview of IoT service model in cloud platforms](image-url)

### A. Challenges

In large-scale IoT applications, the structure of the registry for IoT resources is particularly important for the following reasons. First, the sheer number of resources in the Cloud requires a highly scalable registry mechanism to ensure swift and real-time discovery of resources. Second, unlike many conventional distributed systems, resources in IoT systems relate to each other semantically and contextually. For instance, the traffic monitoring application may be interested in invoking a service which involves resources that are located in the same environmental context, e.g., a monitoring service for a road with light and smoke sensors. This calls for a resource registration model that adheres better to the physical and contextual structure of IoT services in the environment.

The inherent event-driven nature of IoT services is the other challenging issue that should be carefully addressed in IoT cloud services (cf. Event-Driven Processing). For instance, in contextual notifications, devices populated under a pre-defined context boundary may be of interest for event reporting, e.g., listening to all sensing events produced in a given region of a city. This dynamic and multi-dimensional model of processing event-driven services cannot be easily implemented over publish/subscribe middleware solutions such as MQTT [16]. It, in fact, poses the need for a scalable model for interaction with event-based IoT services which allows event filtering at different levels, from a sensor reading, to changes in a physical context and shared notifications.

Finally, the services exposed by IoT devices are neither reliable nor available always due to their mobility and being transient due to power limitation. With respect to mobility, IoT devices may switch between different networks and operation environments, resulting in different perception of the produced data (i.e., can be interpreted as context-based data). Ideally, the applications that communicate with such types of services should be instantly notified about, e.g., presence or absence of a mobile service and can access to historical service data when, e.g., the target service is out of access.

III. MOTIVATING SMART CITY PLATFORM

In this section, we focus on a real smart city platform and investigate its design aspects with respect to the aforementioned challenges. The cloud-based smart city framework is
taken from a recent research project that is focused on IoT-based smart cities—called ClouT [17]. The overall concept of this project is to leverage the integration of IoT and cloud computing to establish an efficient communication and collaboration platform exploiting all possible information sources to make the cities smarter and to address associated challenges, such as efficient energy management. Urban context-aware applications and safety and emergency management are two of those application domains studied in the ClouT project.

The purpose of the urban context-aware application trial, deployed in the city of Fujisawa, is to detect and leverage sensorised social web and IoT for detecting city events such as festivals or accidents. Fujisawa classifies city events in terms of their property such as size and contents. Detected and classified events are useful for further services such as navigation and event recommendation. Similarly, the Genova field trial is to inform citizens about the city in general (e.g., traffic events) and information about environmental risks and emergency situations, in particular. The pilot application provides information on: seismic (earthquake), weather risk, hydraulic and hydrological (floods, flooding, landslides), fire risk, weather sensor data, and critical events.

The main sequence of actions, in the above applications, is illustrated in Figure 2. Firstly, IoT data (e.g.,) is sent to a processing service (i.e., Sensorizer module), then sensorized data will be transferred to Database. Event Detector retrieves historical sensorized IoT data from Databases, and detects events by analyzing spatial-temporal information of the IoT data. Event Detector also sends information of interest for third-party applications (i.e., City IoT Data Consumer) to Event Classifier for further analysis of event. Event Classifier classifies events in terms of, e.g., their popularity. Classified event information is used for navigation or recommendation. For instance, the Genova pilot application uses the cloud storage functionalities, provided by ClouT platform, in order to store historical data from sensors: this result will be achieved by using a specific software agent that performs data polling from Genova infrastructure services.

![Fig. 2: Sequence of interactions between components in urban context-aware application](image)

The cloud storage offers an API to get the stored data by different applications on-demand:

GET http://host:port/clout/devices/ (deviceID)/services/(serviceID)/ resources/{resourceID}/GET

For example:

GET http://93.48.18.248:8080/clout/devices/ PowerService_SmartPlug_0/resources/ status/GET

In addition, it provides an API to subscribe to data change notifications of a given resource:

POST http://host:port/clout/devices/ (deviceID)/services/{serviceID}/ resources/{resourceID}/SUBSCRIBE

For example:


From Figure 2 and the UDDI-based discovery mechanism of the ClouT platform, it can be concluded that discovering one or a set of relevant services is not a trivial task, in particular for smart city scenarios in which applications are often interested in services bounded to a location context. Additionally, in the event-based interaction model of ClouT, the event subscription model is built on a flat device-oriented description of IoT resources (cf. last POST example). However, smart city applications are naturally interested to events published within a particular context boundary such as a street or a building. Thus, there is a clear need for scalable context-based discovery of IoT services and realtime access to their data in cloud-based IoT platforms for smart cities. To the best of our knowledge, there is no approach that specifically addresses these challenges for cloud-based IoT applications with the above requirements.

IV. PROPOSED APPROACH

The overall design approach is an scalable tree-based service access model in which each node of the tree is created based on hierarchical contextual parameters of the IoT application, e.g., the location context for a smart city can be hierarchized from high level city municipal districts down to neighborhoods and streets. We name these types of nodes context nodes. At the level of leaf nodes, in a hierarchal path, IoT resources are located, called resource nodes. A context node is empowered with the ability to generate events upon adding, removing or updating its children. On the other side, context nodes can listen to events generated by other nodes in the tree in realtime, including those generated by their children. In a broader scope, third-party applications or the end-user can subscribe to a particular node of the tree. For example, in the motivating application, an information board screen can subscribe to particular event types occurring in a given street or certain districts of the city.

Prior to exploring further this design solution, we give a simple example to better understand our approach. In the context of motivating smart city platform in Section III, let us focus on safety and emergency management in the Japanese city of Fujisawa. In the famous Suban street, there are often many tourists and there is a need to provide evacuation...
information to tourists in case of emergency. This can be done through sensors deployed throughout the street, where one of them is atmospheric sensor—which includes CO, NO, O3, dust, temperature, humidity, luminance, and air contaminants sensors. Based on our tree-based design model, the realtime information about this street can be accessed through:


The obtained atmospheric information can be used by the government or the city in case of emergency, e.g., Tsunami. In addition to the above street-level monitoring, the realtime information (such as weather, traffic and day events) about the Enoshima area in this street can be displayed through the designated information board—called Enoshima info surf-board. For example, to update the day event information, the following PUT call should be made (based on our tree-based service access model):


Driven by the challenges discussed in Section II-A, this design approach brings a number of advantages to the design of cloud-based IoT systems. The proposed tree structure for resource access will enable scalable discovery and registration of resources, in addition to the fact that such a hierarchical access model adheres completely to the physical and contextual structure of IoT services. More importantly, event processing and filtering will be accomplished at divers contextual levels which is inherent to the design of large scale IoT applications. Beyond this, the realtime addition or removal of resource nodes will greatly simplify the integration of mobile or unreliable IoT devices into the system.

A. Design Model and Features

As implicitly mentioned in the previous section, our approach for scalable and realtime provisioning of IoT services revolves around two fundamental design techniques: i) how to design an efficient context-based hierarchy of IoT resources, which enable multi (contextual) levels of reading, updating, monitoring and storing IoT services data in the Cloud, and ii) how to process and publish realtime services data to interested parties at different context levels.

With respect to the first design aspect, it is important to find an appropriate and efficient way for extracting the target IoT application’s context and map it to the hierarchy of context elements. This first requires careful investigation of core context elements in IoT systems. As described in [18], the main context elements for a typical IoT system include identity, location, time and activity. There are basically derived from the primary context types introduced for computing systems in general [19]. As an example of identity, the context information may include information about the user of a smartphone. The user’s identity is made available and augmented with other information, such as the user’s profile or activity. Besides such context types which are more relevant for the user of things, location and time are the core context types linked to smart things themselves.

We propose to categorize the contextual information into temporal and non-temporal. This not only eases the support of realtime processing of IoT context, but also allows associations between other context types and temporal types. The latter is if high importance as indicated in our motivating scenario and the presented examples. Figure 3 presents the different components of the framework for realtime and scalable provisioning of IoT services. In the following, we describe these components.

![Resource Access Tree](image)

**Fig. 3**: Overview of proposed solution for realtime and scalable provisioning of IoT services

**Resource**. Each resource has a resource type, which can be an instance of either of the following types: sensor, actuator, or tag [20].

**Resource Access Tree (RAT)**. This is the core part of the system for scalable service provisioning. As discussed above, RAT is built on the context information of target IoT application. It can include the key context type of location, or other types such as user-related context and the IoT domain. We propose to add the domain to the hierarchy of RAT as IoT resources are often shared between many domains. Although location is the most relevant data type for context hierarchy, in some applications service provisioning may be based on only a domain or sub-domain context. As an example, the URI for the atmospheric resource represents a combination of domain and location hierarchy for service access.

**Temporal Feature**. As mentioned before, the temporal context is proposed along with the context hierarchy—RAT. We focus on three reference temporal aspects, including Realtime, TimeRange, and DateRange. While the last two aspects are proposed as de facto semantical modeling concepts for IoT resources [20], we add the Realtime aspect to particularly meet the realtime requirements. TimeRange and DateRange support the provision of historical service data, in contrast to the Realtime feature that publishes resource events to the interested nodes of RAT.
Central to the design model in Figure 3 is events. The concept of event in RAT can be described as: adding a child, removing a child, and updating a child, e.g., adding a new data item for a resource, removing an old item, or updating the value of a data field, respectively. In general, two types of event subscribers are envisaged: IoT devices and end-user applications. As an example for the former, the information board in the Fujisawa smart city application acts as an IoT device interested in receiving realtime data from surrounding sensors in the environment. Emergency management applications and central traffic monitoring applications are examples of the latter category, to name a few.

B. Preliminary Implementation

Our main goal in implementation is to investigate existing cloud computing platforms for efficient data storage and processing and build our framework on a platform that meets our design goal of scalable and realtime service provisioning. Among existing platforms, Firebase [21] is perhaps the most efficient and appropriate platform. Firebase is a cloud-based, realtime back-end system that allows us to build various data processing features in realtime. For example, a mobile application can upload and add its current sensory information and the changes will be saved automatically in realtime so that other users of the mobile application or third-party applications could be updated about such realtime changes. Data in a Firebase database is stored as JSON and synchronized in realtime to every connected client. In this way, Firebase enables a high-level decoupling between data producers and data consumers, while the structure for describing the data plays a key role in providing a meaningful perception of individual data items stored in Firebase.

To better understand the data processing and storage model of Firebase, as a proof-of-concept experiment, we have developed the sample scenario of Fujisawa smart city as a Node.js application with the Firebase platform. Figure 4 illustrates the Firebase data console for the SmartCityIoTCloud application we have developed using Firebase as a service provisioning and realtime data access platform.

As shown in the figure, the IoT data is organized based on the hierarchical location context of IoT devices spread in the city. Under each location context (e.g., Subana street and Shanon street), both realtime and historical data are made available to interested applications and other IoT devices.

C. Discussion

Although we argued that tree-like data structures can model efficiently services in typical smart city applications, in some cases there might be needed to maintain services within a set of disconnected trees (i.e., forest) or a graph data structure that looks like a set of trees merged together, each with its own root. For example, applications with different context roots, sharing same IoT devices, can take advantage of such data structures. In this paper, we assume that all smart city applications share one context root. With respect to the event-based notification in our framework and its implementation over Firebase, IoT devices have to support execution of Firebase libraries for IoT data storage and notification. However, some resource-constrained devices are not yet able to host those libraries. There are two solutions for such cases: i) making powerful gateway nodes as proxy for interaction with the Cloud and maintaining service data, or ii) developing customized integration libraries for target IoT nodes. It should also be noted that the proposed service modeling approach and the associated framework for scalable service provisioning are generic. In fact, we use Firebase only to show how the proposed framework can be integrated to a cloud-based storage platform.

V. RELATED WORK

IoT domain models which primarily address the issue of IoT device, service and resource modeling, are somewhat relevant in the context of this work. Besides their main goal of modeling, the secondary objective is to tackle the challenge of selecting an appropriate service which satisfies user’s requirements from a huge number of IoT services. Jine et al. [10] propose a physical service model to rate candidate physical services according to a user requirement using the proposed QoS rating functions and then select an appropriate physical service. In [22], a multidimensional resource model is proposed for dynamic resource matching in IoT, motivated by the fact that, with significant numbers of resources, selecting appropriate resources is time-consuming. Both of these works use only properties of devices and sensors to describe resources, which is different from the broader hierarchical context-based viewpoint we have adopted.

Among middleware solutions for IoT Cloud integration, Boman et al. [23] have developed a generic IoT middleware system that leverages a cloud storage service for sharing and consumption of IoT data and an open source sensor data
management system (GSN) for data acquisition from IoT devices. In [9], IoT PaaS is proposed—a cloud platform that supports scalable IoT service delivery, allowing IoT solution providers to efficiently deliver and continuously extend their services. This is achieved through the concept of virtual vertical IoT solutions and domain mediation to leverage computing resources and middleware services on the Cloud. Thus, the efficiency and scalability they promise is based on the traditional benefits of cloud integration.

Some work has focused on semantic web technology to model and structure the IoT data, including sensor data. Jie et al. [11] motivates the need for ontologies that capture the hierarchy and equivalency of information that can be gathered and inferred from sensors. They propose a hierarchical structure sensor information system and a semantic service programming model that uses automatic service planning and a service composition graph to perform optimization for resource-aware execution of the service composite. In [24], an IoT virtualization framework is proposed to support connected objects sensor event processing and reasoning by providing a semantic overlay of underlying IoT cloud. Such approaches can serve as complementary solution for semantic modeling of resource hierarchy proposed in this paper.

Some efforts have been made to use Linked Data concept [12] to model and organize IoT data, and link related sensor data sets. In [25], an approach is proposed to publish sensor data as linked data to enable dynamic discovery, integration, and querying of heterogeneous sensor data sources. Recently, the linked data concept has been used for designing cloud data storage to store IoT data, metadata, and historical sensor readings, using Linked Stream Middleware [26]. OpenIoT is a recent framework, in this category, which provides a middleware platform enabling the semantic unification of diverse IoT applications in the Cloud [2]. In [27], a semantic modeling framework is proposed to annotate streaming sensor data, based on the idea of linked data. The main focus of works in this category is on creating efficient and flexible schemes to describe the sensor streams and their attributes.

VI. CONCLUSIONS AND FUTURE WORK

Scalable and realtime provisioning of IoT services is becoming an important design aspect of large-scale IoT cloud applications, such as smart cities. In this paper, we proposed a hierarchical way of structuring IoT services and their realtime and historical data, adhering to natural hierarchical context domains in smart city applications. Moreover, the proposed framework is empowered with a notification- and context-based access to service data and changes in order to guarantee realtime and fresh service delivery to interested parties. As part of our future work, we aim at investigating further the design details of the proposed framework, such as multiple applications with different event subscriptions.

REFERENCES