8

Internet-of-Things Analytics for Smart Cities

Martin Bauer, Bin Cheng, Flavio Cirillo, Salvatore Longo and Fang-Jing Wu

NEC Laboratories Europe, Heidelberg, Germany

8.1 Introduction

The Internet-of-Things (IoT) is becoming mature. It is moving from the research labs into production environments. In the initial phase, there are mainly small installations focused on a specific application, but, on the horizon, real and wide-scale deployments become visible, especially for realizing the concept of smart cities. From a few sensors providing us, for example, with weather information, we will get to large scale installations monitoring and influencing a wide range of aspects including traffic, energy, water, building infrastructures and public safety, all of which are highly relevant for the smart cities of the future. The true value of such IoT deployments can only be reached if the raw data gathered is processed and higher level information is derived that provides true insights into the real-world situation enabling humans or machines to take actions. The basis for deriving such information is provided by IoT analytics. Individual measurements, e.g., if individual cars or persons are passing a certain spot, may only provide limited benefits, but if the overall traffic situation can be derived or the behaviour of crowds can be determined, suitable actions can be taken. This requires a scalable IoT infrastructure, which can scale with the number of information sources, in particular sensors, the number of different applications and the number of users. To achieve scalability we need to look at all elements of the IoT infrastructure, from sensor nodes with limited resources, to local communication networks, gateways, networks and backend systems. Current IoT architectures typically integrate devices – using a range of different technologies–through gateways. Gateways connect the often resource constraint devices to the backend infrastructure. Information from the devices like sensor measurements are pushed into a logically centralized backend infrastructure. Cloud technologies are used for achieving scalability with respect to storage and processing power. Analytics in the backend can be provided with all the storage and processing power needed. Nevertheless, such infrastructures have their limitations:

- If the sheer amount of data overloads the network infrastructure connecting the gateways to the backend infrastructure, e.g. in the case of video cameras providing a stream of high-resolution images.
- If very fast response times are required for local actuations and the network introduces significant delays.
- If the raw data is not supposed to be stored, e.g. due to privacy information, and only processed results may be provided.
- If the frontend is provided by a different stakeholder who does not want to/is not allowed to provide the raw data.

In all these cases, IoT architectures that only support analytics in the backend are not suitable. The processing should take place in the frontend – at the edge of the network. This requires devices, gateways or specialized servers that are capable of doing the required analytics. In the case of a smart city, the IoT infrastructure needs to be able to support dynamically changing IoT devices as well as changing application requirements. In order to do so, analytic functions need to be dynamically deployed and adapted. In the following, we look at the state of the art, first with respect to the currently dominating cloud-based IoT architectures for analytics (Section 8.2) and show how analytics for crowd estimation can be supported in such a setting. Then we discuss in-depth key challenges for such architectures (Section 8.3). This is followed by a discussion of the state of the art for edge computing and a proposal for an edge-based smart city platform supporting analytics (Section 8.4). Crowd mobility is used as an example to showcase how use cases can benefit from edge-based IoT analytics. Finally, Section 8.5 provides a conclusion and an outlook on future work.

8.2 Cloud-based IoT Analytics

As first, in this section, we will investigate the cloud-based approach for IoT analytics which is the most commonly used by concrete smart cities and adopted by many projects, either funded by the European Commission or by nations, with the scope of creating smart city systems. We will describe a first example of cloud-based city platform for BigData analytics called City Data and Analytics Platform (CiDAP) and a real use-case of cloud-based data analytics such as a crowd estimation system.

8.2.1 State of the Art

Lots of efforts from both industry and academia have been made towards smart cities, but most of them focus on infrastructure construction, data collection, testbed deployment, or specific services/applications development. To support IoT analytics for smart cities, one of the key enablers is to build up a flexible and efficient big data analytics platform between connected data sources and applications. There are only a few studies already exploring big data platforms for smart cities, mainly in the Cloud environment. Examples include SCOPE [4] which is a Smart-city Cloud-based Open Platform and Ecosystem from Boston University, and FIWARE [2] which provides some building blocks for the development of a smart city platform based on the NGSI (Next Generation Service Interfaces) standard. Meanwhile, there are some ongoing projects trying to explore the opportunities and challenges of BigData for smart cities at the platform level, such as CityPulse [5], an ongoing European project exploring real-time stream processing and large scale data analytics for smart city applications. In addition, Singapore is building a new smart city platform called SmartNation [6] to enable greater pervasive connectivity, better situational awareness through data collection, and efficient sharing of collected sensor data. Several concrete smart-city architectures involving data analytics have been proposed. For example, in [19] describes the achievements of building an event driven architecture of a smart city for monitoring public area and infrastructure. All the data is seen as an event. An event can be a new measurement or a discovery of a complex event. The data coming from the Wireless Sensors Network or other subsystems (i.e. CCTV) may be filtered out or aggregated and passed to a cloud-based control center where the raw event (or almost-raw in case of aggregated data) are merged and correlated. The outcome of this processing is the creation of more and more abstracted data from less abstracted data. In case of event above certain threshold the control center would send commands back to the WSN. The analytics involved in this approach is a progressive refining of the available events till a decision. Therefore, the analysis is limited to real-time data and to very specific purposes (like event merging, event correlation or threshold checks). A similar example is described in [15], where a central reasoner is in charge of evaluating incoming aggregated data from Sensor Actuator Network

(SAN). Also, in this solution the analysis is conducted only over real-time data aggregated on the edge with the target of discovery potential critical situations. Aside from the mentioned projects above, there are industrial companies which are advertising their smart city data platforms, like IBM [25], AGT [29], Microsoft [3]. Some companies are also offering ready-to-use generic purpose IoT Platforms with embedded IoT analytics features. For example, the Amazon AWS IoT [14] is a platform capable to automatically scale in the cloud according to the load. The IoT data can be forwarded to other Amazon cloud-based services (e.g. for stream processing, for machine learning applications or for storage purposes). Another solution, more in the industrial plant context, is offered by General Electric [20]. Predix cloud offers capability to connect the gathered data from multiple Predix machines to data analytics service (time series and data analytics orchestration) and several storages options. Also, IBM offers a cloud based platform for IoT: [21]. The idea is to connect the devices or the gateways via MQTT directly to the platform. Once the data is managed in the cloud, the platform is offering integration to many services like analytics (e.g. data streaming processing, predictive analytics, geospatial analytics) and storage (e.g. SQL, NoSQL, time-series etc.).

8.3 Cloud-based City Platform

Typically, for a cloud-based smart city platform the following design issues must be taken into account: First, how to design an efficient storage system to manage a large amount of heterogeneous IoT data? Second, how to deal with both historical data and real-time data in the same platform infrastructure? Third, how to design flexible and generic application interfaces for both internal platform applications and external smart city applications? In this section we explain how these issues can be addressed in a live smart city BigData platform called CiDAP. Currently, CiDAP has been in production for several smart cities globally, such as Santander in Spain, Wellington in New Zealand, and Tokyo in Japan. The CiDAP platform is architecturally scalable, flexible, and extendable in order to be integrated with different scales of smart city infrastructures. The CiDAP platform has been deployed and integrated with a running IoT experimental testbed SmartSantander, one of the largest smart city testbeds in the world. Within the SmartSantander testbed, more than 15,000 sensors (attached with around 1,200 sensor nodes) have been installed around an area of approximately 35 square kilometer in the city. A large proportion of the sensor nodes are hidden inside white boxes and attached to street infrastructure such as street lamps, buildings and utility

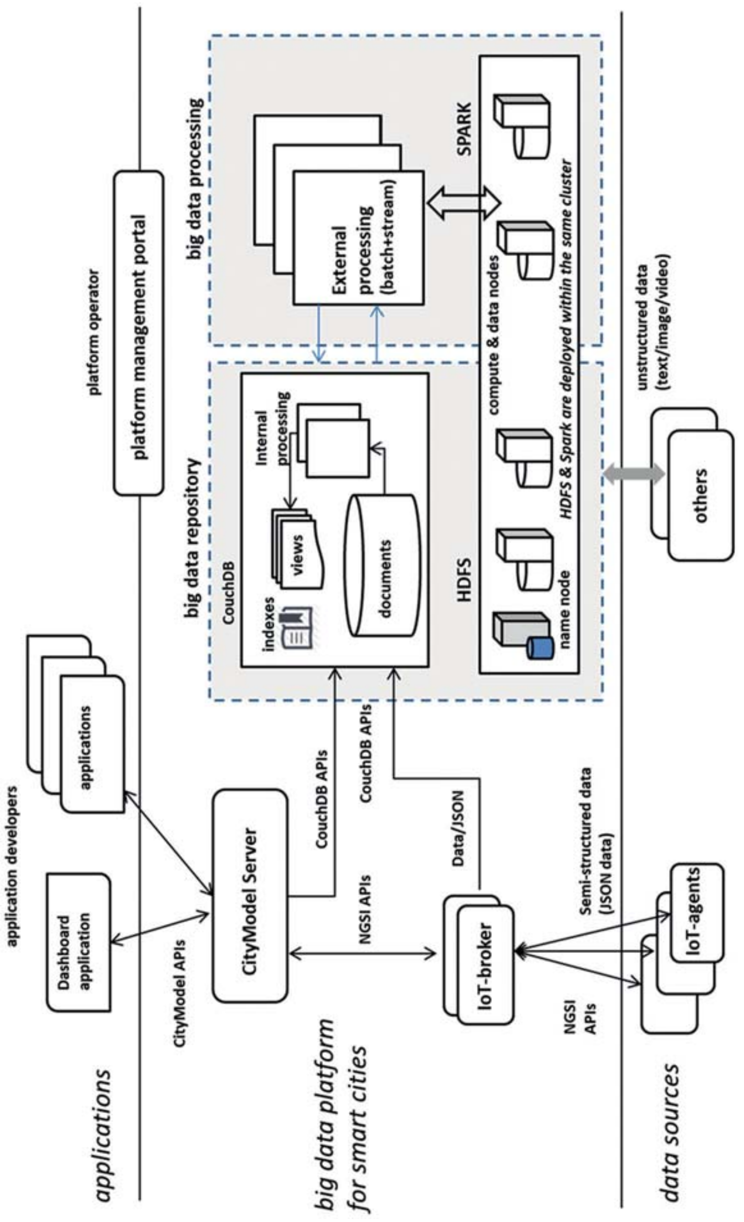

Figure 8.1 System architecture of the CiDAP platform.

poles, while others are buried into the pavement, e.g., parking sensors. Not all of the sensors are static; some are placed on the city's public transport network, including buses, taxis and police cars. The deployed sensors provide real-time information regarding different environmental parameters (light, temperature, noise, CO2), as well as other parameters like occupancy of parking slots in some downtown areas.

Here is how the CiDAP platform works at the high level (see Figure 8.1). First, data with different formats are collected via the IoT-broker [1] from multiple data sources, then forwarded to the BigData repository CouchDB, which is a document based NoSQL database. The collected data are then processed and aggregated by a set of pre-defined or newly launched processing tasks. The simple processing tasks can be performed by the BigData repository internally, such as transforming data into new formats or creating new structured views/tables to index data. Any complex or intensive processing tasks, such as aggregating or mining data via advanced data analytics, must be separated from the BigData repository so they can be efficiently and externally executed over the BigData processing module, which provides more flexible and scalable computation resource based on a Spark [12] cluster with a large number of compute nodes. Since the BigData repository can already handle lightweight processing tasks in a scalable and incremental manner, the BigData processing module can be optional if we do not need intensive data processing or analytics. By fetching generated results from the BigData repository or forwarding messages directly from data sources in the smart city testbed, a CityModel server is designed to serve queries and subscriptions from external applications based on pre-defined CityModel APIs. Meanwhile, a web-based platform management portal is provided to the platform operator to monitor the status of the entire BigData platform.

All external applications communicate with the CiDAP platform via the CityModel server based on a REST based API, called CityModel API. The CityModel API allows application to do simple query, complex query, and subscription. A simple query requests aggregated results over the latest status of all sensors, which represent the latest and real-time snapshot of the entire city testbed, while a complex query can request aggregated results over the historical data collected within a specified time range. Subscription is the mechanism to keep applications always notified with the latest results so that the application does not have to query the data all the time. There are two types of subscriptions, CacheDataSub and DeviceDataSub, as illustrated by Figure 8.2. The difference is CacheDataSub goes to the data repository CouchDB while DeviceDataSub goes directly to physical

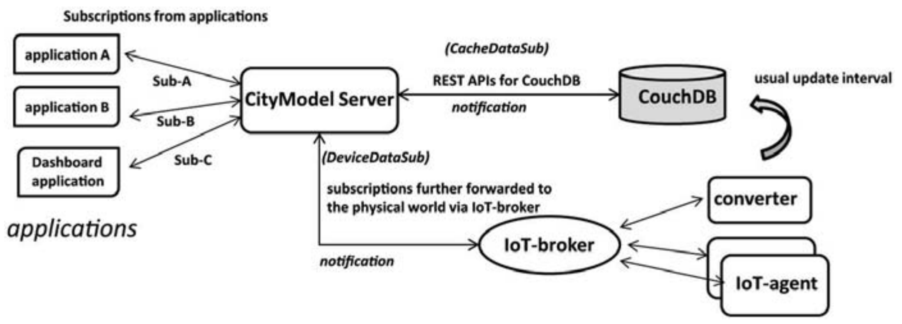

Figure 8.2 Subscription mechanisms to get real-time notifications.

devices in the IoT testbed through IoT-broker and IoT-agents. Both of them are designed to notify applications with real-time changes, but with different expected latency. The notification latency for CacheDataSub is relatively longer than the one for DeviceDataSub, because devices will fire notifications immediately after the requested changes happen, without waiting for the next report period. Unfortunately, the DeviceDataSub is not fully working in the integrated system with the SmartSantander testbed because the sensor nodes in this testbed can only report updates in a passive and periodic way.

CiDAP is just a concrete example to illustrate how a smart city platform could be designed to support flexible IoT analytics in a cloud environment. On the other hand, based on our experiments and experience with CiDAP, we have also identified certain limitations of the cloud-based solution. For example, with the cloud-based solution it is difficult to support time-critical use cases such as autonomous driving and real-time emergency detection, because the responsive time to react on real-time situation could be more than 10 seconds. However, this limitation can be overcome by edge-based solutions, which will be introduced in Section 8.3.

8.3.1 Use Case of Cloud-based Data Analytics

The crowd estimation system is to map a set of sensing readings into a certain level of crowd density. Figure 8.3 shows the system overview of cloud-based crowd estimation proposed in [23]. The system is deployed in a shopping mall, where 23 sensors are installed. The size of the shopping mall is roughly 90 square meters. The sensors continuously report ambient information (such as CO2, noise level, temperature, and humidity) to the

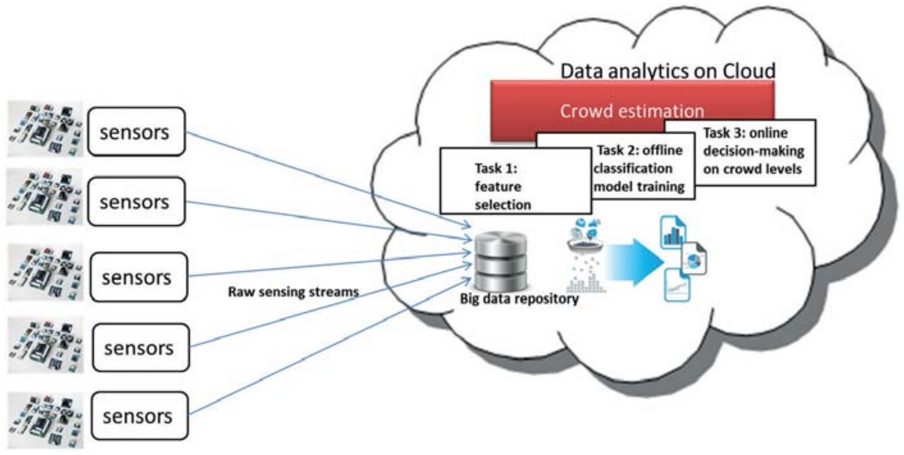
Data analytics on Cloud
Crowd estimation
Task 1: feature selection
Task 2: offline classification model training
Task 3: online decision-making on crowd levels
sensors
sensors
sensors
sensors
sensors
Raw sensing streams
Big data repository

Figure 8.3 System overview of the cloud-based analytics.

BigData repository. In this system, all tasks of data analytics are handled by Cloud. The task of feature selection chooses the most important information behind the sensing data which will be the input of the task 2. For each sensor, the task extracts the mean, standard deviation, variance, minimum and maximum from sensing readings. In addition, the slope of temperature, humidity, noise and CO_2 readings are considered. The second task trains a classification model based on the features selected by task 1, where this task considers four different supervised learning algorithms including Naïve Bayes Classifier [31], C4.5 [26], Random Forests [17], and Support Vector Machines [18]. The ground truth is established through human observations, where the system pre-defines 4 crowd levels from 0 to 3 which are mapped to "occasional passer-by", "sparse traffic", "crowded" and "overcrowded" respectively.

Based on the observations by the building management office, the crowd level 0 is mapped to 0–15 people, the crowd level 1 is mapped to 16–30 people, the crowd level is mapped to 31–45 people, and the crowd level 4 is mapped to more than 45 people. Finally, the third task performs decision-making to estimate levels of crowd density when real-time sensing data arrives. Given a location and features from multiple types of sensing readings, this task can map those information to a level of crowd density. However, IoT data contains much useless and redundant information such as zero readings. Meanwhile, an IoT platform may serve many IoT applications simultaneously and some of real-time IoT applications may have critical QoS requirements. To support real-time applications, flexible and dynamic data analytical models across the system will be preferred, where some processing tasks can be offloaded onto edge.

8.4 New Challenges towards Edge-based Solutions

Different from traditional data analytics like Web analytics and log analytics, IoT analytics must deal with the following IoT system characteristics:

1. IoT data are usually unstructured stream data and constantly generated from geo-distributed sensors over time, ranging from time series event streams to high data rate video streams; sending all raw data to the centralized Cloud for further processing will be very costly and also introduce too much traffic to the underlying network;
2. Mobility and co-location of sensors and actuators, meaning that both sensors and actuators are possible to move and actuators usually require data from nearby sensors;
3. Actuators often expect low latency results to make fast actions;
4. Raw data and derived results are also expected to be shared and consumed across different parties from anywhere, either globally from the Cloud or locally from a nearby region, because the cost to deploy the infrastructure of a large-scale IoT system could be very high and the platform and the sensor data are worth to be shared for maximizing their benefits. All of these requirements bring new technical challenges to IoT analytics since problems like data distribution, data reliability, real-time data processing, processing flexibility, and platform openness need to be considered and addressed differently.

Regarding the requirements of IoT analytics, there is currently a new trend to move processing to the edge, where IoT data are generated and analytics results are consumed. Traditional computing models collect IoT data and then transmit them to a data center for doing scalable data analytics, but this is no longer a sustainable and suitable model for large-scale IoT systems.

Our previous experimental results from CiDAP also indicate that some processing should be shifted from the Cloud down to the edge or IoT devices, especially when applications expect to have real-time analytics results within a few seconds or even a sub-second. Since IoT data are not only big, but also naturally geo-distributed and increasing over time, processing all data only in the Cloud will introduce high bandwidth cost between the network edges and the Cloud. In many cases, it would make more sense to process or compress data before transmitting the data to the Cloud, or transmit only selected data or derived results (e.g., anomalies, exceptions, averages). Therefore, for large IoT systems there is a strong need to do analytics at the network edges. For edge-based IoT analytics, the following challenges must be taken into account.

1. *Scalability*: Edge-based IoT analytics needs to be able to scale up to thousands of geo-distributed nodes over the wide area network. For example, if we consider IoT gateways and even users' mobile phones as edge nodes, the total number of edge nodes in a large scale IoT system can be easily over 1000. According to the recent report by Yahoo [7], supporting over 1000 nodes with Storm within a cluster is already problematic due to the bottleneck of its zookeeper service component.
2. *Task Optimization*: A sophisticated task scheduling algorithm is required to optimize the resource usage and minimize the latency to deliver analytics results. The underlying network topology of all IoT agents needs to be considered by the task scheduling algorithm as well, since it can affect the latency and the bandwidth consumption to produce analytics results. Also, the task scheduling algorithm needs to be aware of the geo-locations of sensors and actuators.
3. *Flexible Application Interfaces*: application developers should have enough freedom to implement their own processing tasks for any type of streams, such as event streams, text streams, and video streams. Further, they should be able to define their application requirements and to access real-time analytics results from their applications. Although this can be built on top of existing solutions, none of the latter includes inherent platform interfaces for supporting this.
4. *Multi-tenancy Support*: The designed edge analytics platform should allow multiple users to share the same edge analytics infrastructure by ensuring efficiency, fairness, and quality of service. This must be achieved by designing sophisticated task scheduling and resource orchestration mechanisms. Resource sharing across applications and users is highly important, since the deployment and maintenance cost for a large-scale IoT system is still big and its value should be maximized by enabling more sharing across various applications and users.
5. *Openness and Security*: Edge-based IoT analytics platform is provided as a PaaS for a set of IoT applications to do stream-based edge analytics. Therefore, it will be important to consider the openness and security issues at the design phase.

8.5 Edge-based IoT Analytics

In this section we examine the edge-based approach for data analytics, which is still at a very early stage in the Smart City context and in general in the Internet-of-Things world. We will introduce our edge-based solution for IoT analytics, describing the architecture in every components and the system workflow. Also for this solution we provide a real-use case of IoT analytics applied to our edge-based solution.

8.5.1 State of the Art

Fog computing is a term first advertised by Cisco, also known as Edge Computing [30]. Basically it refers to extending cloud computing to the edge by allowing data processing to happen at the network edges. As reported by the survey of [16], fog computing has been introduced mainly because of the strong needs of IoT systems for low latency results and fast decision makings. Cisco has created a platform called IOx to support fog computing by hosting applications in a Guest OS running in a hypervisor directly on the network routers. Like a virtual machine, IOx enables running scripts or even compiled code at the network edge. Although fog computing providers like Cisco establish an environment to do distributed computation at edges, to benefit from such environment enterprises still need a system that determines which data needs to be processed immediately at the edge and which data should be moved to the Cloud for further deep analysis. Currently, as compared with cloud computing, fog computing is still in the very early stage and lacks sophisticated data analytics platforms that allow us to efficiently utilize the power of the edges and the Cloud together.

As a new trend, edge-based IoT analytics aims to leverage the power of both fog computing and cloud computing to support real time stream processing. Only a few early stage studies have been done in this area, for example, a recent work from Carnegie Mellon University [27] proposes a VM-based edge computing platform for performing video analytics at the network edges, but it only focuses on video streams and does not consider how to define topologies to do customized stream processing on top of the edges and the Cloud. In addition, some industrial systems have been done to explore edge analytics, such as AGT IoT analytics platform [8], Geo-distributed analytics from ParStream [11], and Quarks from IBM [13]. However, the details of their system designs are not open. From what they advertise about their systems, none of them seems to support multi-tenancy and dynamic topology execution.

The usage of edge computing in concrete smart cities deployment is usually meant only for data aggregation [19] or for semantic reasoning on local data [15]. The computation procedures are statically installed on the edge node and only pre-defined commands (like threshold settings) can be sent by the central application.

8.5.2 Edge-based City Platform

To address the above challenges, we introduce our new edge-based city platform called Geelytics in this section. As an edge-based platform solution for IoT analytics, Geelytic is not supposed to be a replacement of the cloud based solution like CiDAP, but rather an enhancement or a complementary solution to relax the pure cloud-based solution with the capability of edge analytics.

Geelytics is mainly designed for large scale IoT systems that consist of a large number of geo-distributed data producers, result consumers, and compute nodes that are located both at the network edge and in the cloud. Data producers are typically sensors, connected cars, glasses, video cameras, and mobile phones, being connected to the system via different types of edge networks (e.g., Wi-Fi, ZigBee, or 4G, but maybe also fixed networks). They are constantly reporting heterogeneous, multi-dimensional, and unstructured data over time. On the other hand, result consumers are actuators or external applications that expect to receive real-time analytics results from sensor data and then take fast actions accordingly. Both data producers and result consumers could be either stationary or mobile. In between them there are lots of compute nodes geographically distributed at different locations. In general compute nodes are heterogeneous in terms of resource and data processing capability and they can be located at different layers of the network. For example, they could be small data centers at base stations in a cellular network or IoT gateways in factories or shops.

The Geelytics system is designed as an IoT edge analytics platform that allows consumers to dynamically trigger certain stream data processing either at network edges or in the Cloud to derive real-time IoT analytics results from a set of data providers. At very high level, it works like a distributed pub/subsystem to interact with geo-located sensors and actuators, meanwhile having a built-in stream processing engine that can perform on-demand IoT stream data analytics based on the underlying Cloud-Edge system infrastructure. As shown in Figure 8.4, the Geelytics platform includes the following components.

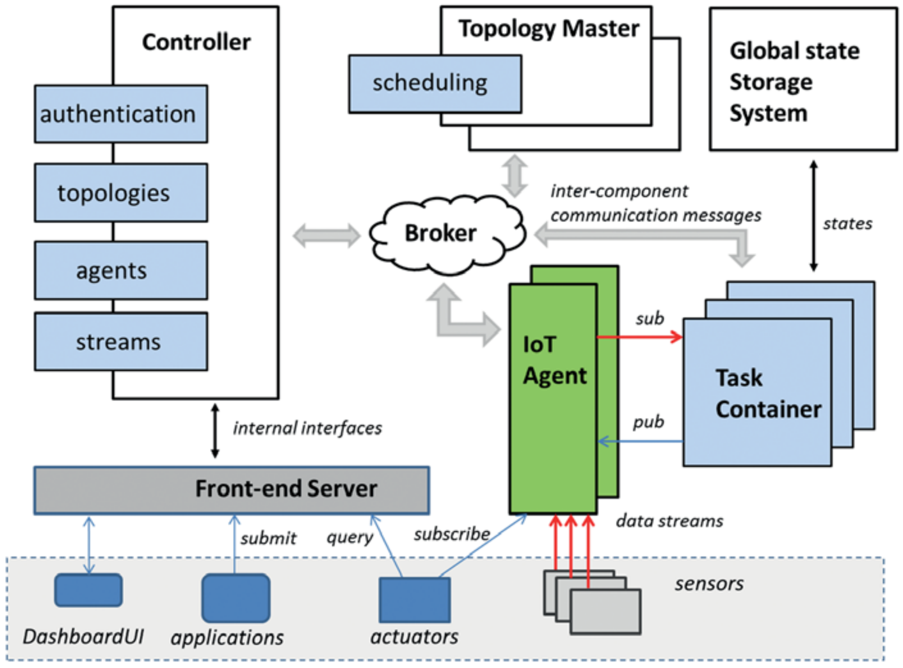

Figure 8.4 Subscription mechanisms to get real-time notifications.

IoT Agent: each IoT agent represents one worker that is capable of performing stream processing tasks. In Geelytics IoT agents are deployed on geo-distributed physical or virtual machines, either in a cluster in the Cloud or at the network edges. Each agent communicates with the Controller to report its capability and available resources, accepts incoming tasks from the topology masters, and instantiates them locally. It can also receive data streams from nearby sensors and fetch data streams from other remote IoT agents according to the requirements from its accepted running tasks. All IoT agents have the same role, but they might be heterogeneous depending on the processing capabilities and network connections of their host.

Task Container: every schedulable task is wrapped up as an application container by developers. Based on Docker [9], it can be fast deployed and executed anywhere by an IoT agent. By design, each running task within an application container will communicate with its IoT agent via a pub/sub mechanism, including subscribing input streams and publishing generated output streams. Using Docker as the environment to run IoT analytics tasks, we are able to better support multi-tenancy, because Docker allows us to do fine-grained resource allocation for each task.

Topology Master: In Geelytics, an IoT analytics application consists of a set of correlated data stream processing tasks. Each application has a dedicated topology master to manage all involved stream processing task instances running in the Cloud or at the network edges. Each topology master is responsible for monitoring and allocating tasks to different IoT agents. It requests the current state of all available resources, including all active IoT agents and their remaining capabilities, network latency and traffic across IoT agents, and then make decisions on at which IoT agent each task must be instantiated, regarding the task topology specification and optimization objectives given by the application developer and the current workload. By separating Topology Master from the Controller, Geelytics is able to achieve better scalability as compared with existing stream processing platforms like Storm. In addition, several task assignment algorithms have been applied by Topology Master to optimize task allocation between Cloud and edges during the runtime, with regards to the objectives of reducing bandwidth consumption and latency.

Controller: all system resources and core components are managed by the Controller, which is a single central control point of Geelytics running in the Cloud. It indexes all streams, agents, topologies, and users. For security reasons, it authenticates all the other components, especially IoT agents, when they join the system.

Front-end Server: application interfaces are supported by the front-end server via HTTP REST, enabling that: 1) application developers can submit the task definition, topology structure, and optimization objectives; 2) actuators can query or subscribe the analytics results generated by the submitted application; 3) sensors can register them to a nearby IoT agent; 4) a dashboard service is provided to check the status of the entire Geelytics system and also to manage users and applications.

Broker: the Broker is a distributed message exchange system to enable the communication between different components. To be scalable and flexible, the Broker must have high throughput and support topic-based message handling.

Global State Storage System: Geelytics is designed to support complex stream processing tasks, such as machine learning tasks, image or video processing tasks. For those tasks, it is important to save some of the intermediate states to tolerate unexpected failures. The same concern goes for the other components as well, such as the Controller, IoT agents, and the topology master. Therefore a global state storage system is introduced to keep all intermediate states and results, using existing NoSQL database systems such as key-value based Redis or document-based CouchBase.

8.5.3 Workflow

The system platform is initialized in the following sequence. First, the broker and the global state storage system must be set up independently as two external sub-systems. Then the Controller is started in the Cloud and it will launch the front-end server. IoT agents can be started before or after the controller, but each of them must be authenticated by the controller when they join the system. As stream data sources, sensors can be attached to an IoT agent manually or be forwarded to the nearby IoT agent by the controller when they join the system. After the system is ready, developers need to register a user account and then start to submit their customized tasks and application topologies. Once a new application is submitted, the Controller will allocate proper resources for the application according to its requirements and then return a URL address to the actuators of this application for accessing the analytics results.

8.5.4 Task and Topology

As the example in Figure 8.5 shows, in Geelytics a data analytics application is defined by a task topology, which specifies the relationship between different stream processing tasks within the application. Based on the task topology, a processing topology will be created on the fly to handle the current workload. The processing topology consists of a set of running task instances, allocated by the topology master to the network edges or the Cloud, up to where the involved data sources are located and where the results are demanded by the actuators. In Storm a processing topology is constructed when the task topology is submitted, according to the parallelism of each task defined by the developer. In contrast, in Geelytics all data streams generated by each task in the task topology are accessible to actuators and the processing topology is constructed and changed as actuators join and leave.

In Geelytics the way to implement a task is flexible. A task just needs to follow a pub/sub communication interface to fetch the input streams and publish the output streams and a set of parameters are configured with the task to decide which input streams to bring in. However, how to handle the input streams within the task is a black box for Geelytics. Developers can use

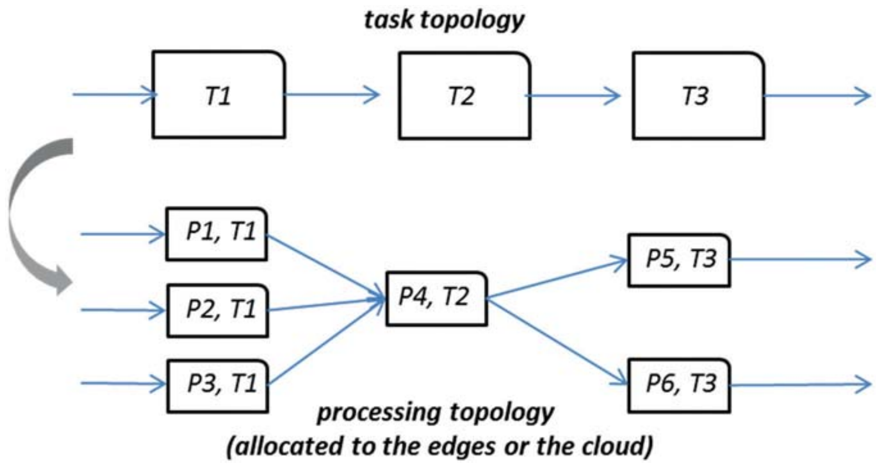

Figure 8.5 Task topology and processing topology.

any image/video processing or machine learning library to implement their tasks in any programming language, because they can wrap up all dependent libraries into a single Docker container image. In addition, all tasks can use the interface of the global state storage system to save or retrieve their state data.

8.5.5 IoT-friendly Interfaces

Geelytics also provides friendly interfaces for both data producers and result consumers to interact with the system. In Geelytics all data producers report their availability, profiles, and data streams to the system, managed by a repository based on ElasticSearch [10]. The way to fetch the data streams generated by data producers can be push-based or pull-based. In the push-based approach, data producers publish their stream data to the MQTT broker on the nearby compute node; while in the pull-based approach, data producers just announce the URLs of their streams, and later on it is up to task instances to fetch the data directly. A data producer first has to ask the controller to find a nearby worker and then registers its data stream via the nearby worker with the following details: its device ID and location, the generated stream type, and the manner to provide the stream data (push-based or pull-based). A unique ID will be returned to the producer as the global identity of its data stream. If the stream is pull-based (for example, a web camera), a URL must be provided for accessing data as well; if the stream is push-based, using the unique ID as the topic, the producer publishes the generated data to the broker provided by the nearby worker via MQTT.

A result consumer also needs to ask the controller to find a nearby worker first. After that, it sends a scoped subscription to the nearby worker for triggering some real-time data processing over the specified data sources. A subscription ID is returned to the consumer to make a further subscription to the broker. The consumer can receive the subscribed results as soon as they are produced by the triggered task instances in Geelytics. Those running tasks will be terminated once the consumer decides to unsubscribe to the result or its leaving without notice has been detected.

8.6 Use Case of Edge-based Data Analytics

A real use-case for edge-based data analytics is the study of crowd mobility pattern analysis and prediction. In the next subsections we are going to examine how we can design such application and how it is fitting in the Geelytics platform.

8.6.1 Overview of Crowd Mobility Analytics

Crowd mobility analytics investigate how many people in a certain area and how they move from one area to the others which can provide insights for various IoT applications. For example, stadium operators may need to know the number of people in a big event in case of emergencies, and airport or public transport operators may need to know passenger flows for predictable service enhancement and maintenance scheduling.

Figure 8.6 shows the overview of the crowd mobility analytics system, where the IoT platform consists of data sources, edge nodes, and cloud nodes. The data sources include Wi-Fi sensing stations and ambient sensors. Each Wi-Fi sensing station captures Wi-Fi probe requests broadcast by mobile devices from time to time, while ambient sensors include CO2, noise, temperature, humidity, and motion sensors which capture the influence of human mobility on the environment. Each edge node serves as a local data aggregator to connect to cloud nodes. Cloud nodes cooperate with edge nodes though shared backed database. Since the architecture of edge-based data analytics provides a more flexible task processing topology, it opens up more opportunities for processing data streams in a pipeline way which can speed up data analytics. Thus, the crowd mobility analytics decompose the mobility data analytics into six processing tasks based on the dependency among tasks:

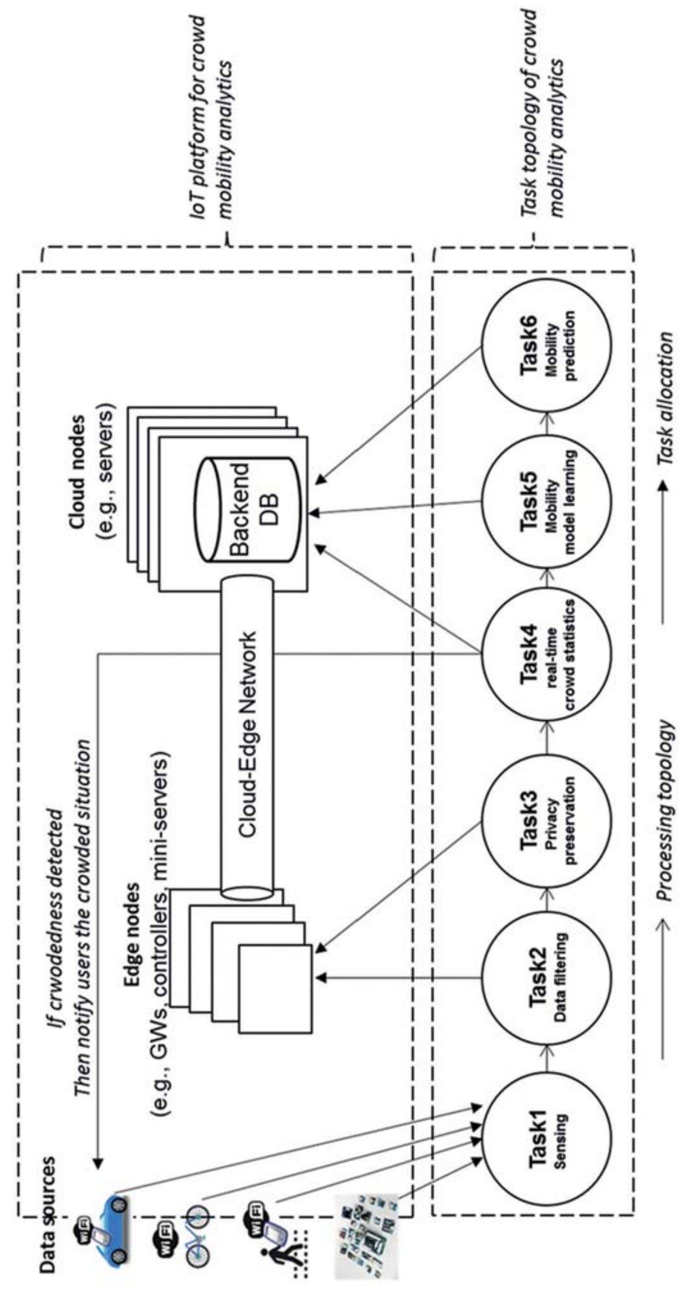
Data sources
If crwodedness detected
Then notify users the crowded situation
Edge nodes
(e.g., GWs, controllers, mini-servers)
Cloud-Edge Network
Cloud nodes
(e.g., servers)
Backend DB
IoT platform for crowd mobility analytics
Task1 Sensing
Task2 Data filtering
Task3 Privacy preservation
Task4 real-time crowd statistics
Task5 Mobility model learning
Task6 Mobility prediction
Task topology of crowd mobility analytics
Processing topology
Task allocation

Figure 8.6 A system overview of the edge-based data analytics for crowd mobility.

(1) Wi-Fi sensing, (2) data filtering, (3) privacy preservation, (4) real-time crowd statistics, (5) mobility model learning, and (6) mobility prediction. The first three ones are lightweight sensing and data pre-processing tasks which will be assigned to edge nodes, while the latter three tasks are computation-intensive which will be assigned to cloud nodes.

8.6.2 Processing Tasks and Topology of Crowd Mobility Analytics

Since edge nodes have enough capabilities to run simple routine, the processing topology of crowd mobility analytics is designed to decouple the computation complexity between cloud and edge nodes.

First, we introduce the three lightweight tasks allocated to edge nodes as follows.

- Task 1: sensing. We build passive sensing stations to capture Wi-Fi packets broadcast by mobile devices and sensing readings, where each sensing station was build using a Raspberry Pi 2 with Arch Linux, a Wireless LAN USB Adapter, and ambient sensors.
- Task 2: data filtering. Since the previous sensing task captures all of Wi-Fi packets including dense beacons, this task picks up only Wi-Fi probe requests and represents sensors readings as a common format. Meanwhile, this task transforms the raw sensing data into a structured format for further mobility analysis in cloud nodes. For each Wi-Fi probe request packet, the system extracts the BSSID, the Wi-Fi channel on which the packet has been sent, the source and destination MAC addresses, the time when the packet has been detected, and the Wi-Fi device vendor inferred from the first 3 bytes of the MAC address.
- Task 3: privacy preservation. To avoid exposing identities of mobile users, edge nodes perform MAC address anonymization for the privacy preservation purpose. Thus, each edge node sends hashed MAC addresses to the backend database using a SHA-1 algorithm [22].

Afterwards, cloud nodes perform the following three key tasks: real-time crowd statistics, mobility model learning, and mobility prediction.

- Task 4: real-time crowd statistics. This task performs feature extraction and statistical analysis based on the results from many instances of Task 3 in a real-time way. The extracted features include the mean, maximum, minimum, and standard deviation of sensing readings from ambient sensors. The statistical analysis results include the number of mobile

devices, distribution of device brands, and the number of mobile devices moving from one sensing area to the others based on the Wi-Fi.
- Task 5: mobility model learning. Based on these features extracted in Task 4, this task trains a classification model to estimate the number of mobile users in a certain area.
- Task 6: mobility prediction. Based on captured Wi-Fi probes, the task models human mobility as a finite Markov Chain [28] which represents mobility behaviour of public crowds instead of focusing on each individual's mobility trajectories. The behavioural characteristics of crowd mobility can be mapped to a level of crowd which explains how many people staying in a certain area. Furthermore, we can use this model to predict crowd levels based on the statistical analysis of mobility flows among multiple areas.

8.7 Conclusion and Future Work

In this article we discuss the technical challenges to support flexible IoT analytics for smart cities from a platform perspective. As the scale of IoT devices in a smart city is fast growing and fast response time is highly demanded by more and more smart city use cases, for IoT analytics there is a new technology trend to move data processing from the cloud to the network edges. With two concrete platform examples, namely CiDAP and Geelytics, we illustrate this new technology trend and show use cases can benefit from them.

For the time being, CiDAP focuses more on the cloud side while Geelytics focuses more on the edge side. However, Geelytics is not supposed to replace CiDAP as an alternative solution, but rather enhance it as a complementary solution. For example, Geelytics is good at processing stream data both in the cloud and at edges, but it is not a good choice for dealing with large scale historical data in the cloud, which is the strength of CiDAP. Therefore, it makes sense to integrate CiDAP and Geelytics to have a more advanced and unified platform for IoT analytics, which can utilize both edge computing and cloud computing. This is one of the future steps in the short term. In addition, we are further working on the task assignment algorithms in Geelytics to support mobility aware IoT analytics for moving objects, such as connected cars and flying drones.

In the long term, we are working on the issue of semantic interoperability to support advanced IoT analytics that can utilize the data from various data sources across different application domains. In a smart city, relevant data could come from various data sources, either in the same IoT system, from other IoT systems or even from more traditional IT systems whose content may be entered by humans. Semantic interoperability will allow us to interact with various data sources with ensured consistency of the data across systems regardless of individual data format. The semantics can be explicitly defined using a shared vocabulary as specified in an ontology.

For IoT to be successful, standardized solutions are needed – be it formal standards or de-facto standards developed as part of industry alliances or open source communities. In CiDAP we are making use of the OMA NGSI Context interfaces that are at the core of FIWARE [2] Platform. We are also actively participating in the oneM2M [24] standardization. Ultimately, important functionalities developed and explored in our research prototypes need to become part of standardization. Different standards have to be aligned and gaps in standardization have to be identified and closed.

Regarding semantic interoperability, we have integrated basic semantic functionality into oneM2M. Based on this we have done an experiment to show how semantic information can be used for converting IoT data in oneM2M into the NGSI data format used in FIWARE. We are now planning to generalize the approach using the concept of mediation gateways.

For the future work, we would also like to consider the security and privacy issues in IoT analytics for smart cities. We have done some work to ensure the secure communication between different components in both CiDAP and Geelytics, but this is still the basic step to ensure security. With the support of edge analytics, the IoT analytics platform is now geographically deployed with the extension further down to the edges, like mobile base stations, IoT gateway, and even some endpoint devices as well. In this case, it is becoming more challenging to secure the platform and IoT data.

For example, some intrusion detection might be needed to detect attacks and potential threats in real time. The research on privacy in IoT is still at an early stage, but will be an essential point for the adoption of IoT on a large scale.

References

[1] Aeron open source project. https://github.com/Aeronbroker/Aeron/

[2] FIWARE Open Source Platform. http://www.fi-ware.org/

[3] Microsoft CityNext Solution. http://www.microsoft.com/global/en-us/citynext/RichMedia/Safer_Cities/CityNext_Brochure_Safer_Cities_SML_FY15.pdf

[4] SCOPE: A Smart-city Cloud-based Open Platform and Ecosystem. http://www.bu.edu/hic/research/scope/

[5] The citypulse project, 2014.

[6] Singapore smart nation platform, 2014.

[7] The Evolution of Storm at Yahoo and Apache. http://yahoohadoop.tumblr.com/post/98751512631/the-evolution-of-storm-at-yahoo-and-apache/, 2015.

[8] AGT IoT Analytics Platform. https://www.agtinternational.com/iot-analytics/iot-analytics/analytics-the-edge/, 2016.

[9] Docker. https://www.docker.com/, 2016.

[10] ElasticSearch. https://www.elastic.co, 2016.

[11] ParStream Geo-distributed Analytics. https://www.parstream.com/product/parstream-geo-distributed-analytics/, 2016.

[12] The Apache Spark project. https://spark.apache.org, 2016.

[13] The Quarks Project. http://quarks-edge.github.io, 2016.

[14] Amazon. Amazon AWS IoT. https://aws.amazon.com/iot/.

[15] A. Attwood, M. Merabti, P. Fergus, and O. Abuelmaatti. Sccir: Smart cities critical infrastructure response framework. In Developments in E-systems Engineering (DeSE), 2011, pages 460–464, Dec 2011.

[16] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pages 13–16, 2012.

[17] L. Breiman. Random forests. Mach. Learn. 2001.

[18] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. In ACM Trans. Intell. Syst. Technol., 2011.

[19] L. Filipponi, A. Vitaletti, G. Landi, V. Memeo, G. Laura, and P. Pucci. Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. In Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on, pages 281–286, July 2010.

[20] General Electric. GE Predix. https://www.predix.com/.

[21] IBM. IBM IoT FouT. D. Pettersen, "Variation of energy consumption in dwellings due to climate, building and inhabitants," Energy and buildings, vol. 21, no. 3, pp. 209–218, 1994.

[22] IETF-RFC. Us secure hash algorithm 1 (sha1). https://tools.ietf.org/html/rfc3174, 2001.

[23] Salvatore Longo and Bin Cheng. Privacy preserving crowd estimation for safer cities. In Proceedings of the 2015 ACM International Joint

Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, pages 1543–1550. ACM, 2015.

[24] oneM2M. oneM2M – Standards for M2M and IoT. http://www.onem2m.org/aboutonem2m/why-onem2m

[25] M. Kehoe P. Fritz and J. Kwan. IBM Smarter City Solutions on Cloud. 2012.

[26] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[27] Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. Edge Analytics in the Internet of Things. IEEE Pervasive Computing, 14:24–31, June 2015.

[28] J. Laurie Snell. Introduction to Probability. Random House, New York, 1988.

[29] Martin Strohbach, Holger Ziekow, Vangelis Gazis, and Navot Akiva. Towards a big data analytics framework for iot and smart city applications. In Modeling and Processing for Next-Generation Big-Data Technologies, pages 257–282. Springer, 2015.

[30] Shanhe Yi, Cheng Li, and Qun Li. A Survey of Fog Computing: Concepts, Applications and Issues. In Proceedings of the 2015 Workshop on Mobile Big Data, pages 37–42, 2015.

[31] H. Zhang. The optimality of Naive Bayes. In Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, 2004.