Hindawi
Mobile Information Systems
Volume 2017, Article ID 9201640, 9 pages
https://doi.org/10.1155/2017/9201640

Research Article

IoT Architecture for a Sustainable Tourism Application in a Smart City Environment

Michele Nitti,[1] Virginia Pilloni,[1] Daniele Giusto,[1] and Vlad Popescu[2]

[1]*Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy*
[2]*Department of Electronics and Computers, Transilvania University of Braşov, Braşov, Romania*

Correspondence should be addressed to Vlad Popescu; vlad.popescu@unitbv.ro

Received 5 August 2016; Revised 21 December 2016; Accepted 9 January 2017; Published 30 January 2017

Academic Editor: Claudio Agostino Ardagna

In the past few years, the Smart Cities concept has become one of the main driving forces for the urban transition towards a low carbon environment, sustainable economy, and mobility. Tourism, as one of the fastest growing industries, is also an important generator of carbon emissions; therefore, the recently emerging sustainable tourism concept is envisioned as an important part of the Smart Cities paradigm. Within this context, the Internet-of-Things (IoT) concept is the key technological point for the development of smart urban environments through the use of aggregated data, integrated in a single decisional platform. This paper performs the first analysis on the feasibility of the use of an IoT approach and proposes a specific architecture for a sustainable tourism application. The architecture is tailored for the optimisation of the movement of cruise ship tourists in the city of Cagliari (Italy), by taking into consideration factors such as transport information and queue waiting times. A first set of simulations is performed using 67-point of interest, real transportation data, and an optimisation algorithm.

1. Introduction

IoT leads a sweeping cultural change as a huge number of machines, devices, sensors, actuators, and other objects become interconnected to each other and to higher-level systems. Due to the enormous amount of variety of connectable devices and automatically collected data, entirely new services and features can arise, to form the basis of, among other concepts, Smart Cities. IoT and big data are both technology-driven developments leading to scenarios such as the one for the Smart Cities, having the potential to generate enormous market opportunities as well as make citizen lives smarter and more sustainable.

The Smart Cities architectures envisioned or implemented up to date deal mostly with use cases from the following categories: energy, waste disposal, environmental management, and transport. All these use cases can have their needs covered by means of an IoT platform connecting heterogeneous sensing systems with the upper layers dedicated to services and interfaces [1].

Tourism is not only the largest growing industry in the world but it also accounts for 5 to 12% of global greenhouse gas emissions [2]. Therefore, in the context of the passage towards a low carbon environment and sustainable economy, the term sustainable tourism was recently coined and begins to gain acceptance from both sides: tourists on one side and tour operators and the interested territory on the other side.

Without travel there is no tourism, so the concept of sustainable tourism is tightly linked to the concept of sustainable mobility which, for the specific case of an urban environment, can be included in the frame of the Smart Cities paradigm.

Based on these aforementioned concepts, this paper proposes an IoT architecture for a sustainable tourism application in a Smart City scenario. The proposed architecture is tailored for a specific use case: sustainable movement of tourists in the city of Cagliari in Sardinia, Italy.

The paper is structured as follows: Section 2 analyses briefly the key requirements for an IoT architecture operating in a Smart City environment for the specific implementation purpose; Section 3 presents the proposed architecture, while Section 4 is dedicated to the use case description

and model, tested in Section 5 through a first series of simulations. Section 6 presents the conclusions and the future work.

2. Key Requirements for an IoT Platform in a Smart City Environment

The emerging Smart City concept has many definitions and implementation approaches. However, as pointed out in the Introduction, from an infrastructural point of view, all Smart Cities have at their core a highly capable ICT system (IoT platform) connected to network of sensors, wired and wireless broadband connectivity, and advanced data analytics that settle the basis for developing intelligent applications and services for citizens [3].

Due to the current lack of fully defined standards for IoT architectures, the key requirements for their usage in a Smart Cities scenario are rather difficult to be defined. The first steps in this direction have been done by the PROBE IT EU-financed project which aimed, among others, at benchmarking IoT deployments and setting guidelines for IoT rollouts for Smart Cities [4]. Based on some of these guidelines, considering other surveys [5–9] and also the requirements fulfilled by some of the existing commercial IoT architectures and platforms [10, 11], we extracted a set of key requirements for IoT platforms operating in a Smart City environment.

The compliance of the architecture proposed in Section 3 with the identified requirements, considering also the first implementation steps, is discussed individually for each one of the requirements.

2.1. Security Requirements. There are already more connected devices than people on the planet, with the estimation that, by 2020, there will be 50 billion connected devices [11]. The inclusion of these devices in IoT platforms means that they will be readable and controllable over the Internet. Poor or misconfigured networks are potentially vulnerable to attacks and IoT environments are no different at all, as they are always connected to the Internet. Therefore, an IoT platform must have strong built in security, which, specifically for the Smart Cities environment and for the chosen application, should have the following key requirements.

(1) End-to-End Security Mechanisms and Data Encryption. Standard-based encryption from the hardware devices to the IoT platform is arguably one of the best deterrents of data theft. Many services encrypt data once it gets to their data centre, but in many ways data is more vulnerable when it is in transit. The challenge for the developer with doing this from end to end is making the entire authentication happen without the user's intervention, so the data is encrypted automatically. For the specific case of the proposed architecture, the critical links are between the real-world objects and the related virtual objects. To ensure a secured connection, the real-world objects exchange keys and IDs with the virtual objects at the association phase, avoiding any other further connection to. All the subsequent connections between the virtual objects and the upper layers are inside the cloud environment that avoids unauthorized access to the users' data.

(2) Flexible, Configurable Access and Authorization Control. Different user types need different levels of access to the data, especially for the specific scenario of live environmental data monitoring such as temperature and humidity. Therefore, an IoT platform should be able to grant to the users of the Smart City environment different access profiles in terms of ownership, security level, visibility, data polling frequency, etc. The virtual objects used in the proposed architecture can have three types of permissions: if "public," the resource can be accessible to everyone without the use of any keys; if "private," the resource can be accessed to the owner key only; if "friend," the resource can be accessed by providing the friend-key known by the friend virtual object.

All the other security requirements can be covered by the cloud environment which will be used to implement the proposed architecture.

2.2. Flexibility. The market for IoT-enabled devices, although in full extension, is still in the early stages of adoption. Developers of the next generation of connected devices and products need to rely on certain flexibility from their software counterparts aggregating their products. For these reasons, an IoT platform should comply with the following flexibility rules.

(1) Networking and Device Agnosticism. Configuration changes should not be limited to writing/changing the driver and data format when a device is added or updated, giving the freedom to hardware developers to constantly develop their products without being limited by predefined settings or compatibility issues.

(2) Device Manageability. Since a large number of smart devices and sensors can be placed geographically at large distances in a Smart Cities environment, the IoT platform must incorporate means to allow the remote management of these devices. This may include remote delivery of operating system patches, profiles, new analytics algorithms, and key management of parameters.

(3) Usage of Open APIs. Smart Cities inherently can generate large data sets originating mainly from networks of devices and sensors. Much of this data remains stored and not analysed mostly because of its unavailability to third party users other than the one intended for the original application. Therefore, as a quintessential element for the Smart Cities environment, the IoT platforms underneath should allow access to common shareable data and as such to become a motor for the development of innovative applications.

The proposed cloud-based architectural approach using virtual objects decoupled from the actual real-world objects allows for the flexibility requirements to be fully implemented.

2.3. Data Requirements. IoT data comes mainly from things but also from users in the form of metadata. Smart Cities

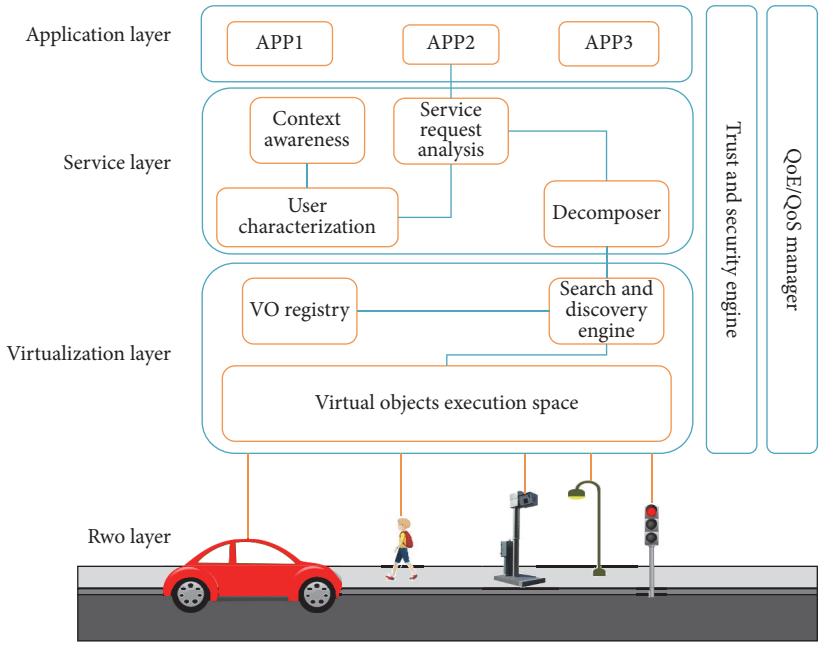

Figure 1: Cloud-based IoT architectural solution.

and IoT are more than a remote monitor of incoming data, with the key concept being data intelligence which implies some strict requirements in terms of data [10, 11]:

(1) Flexible Data Definition Tools, Data Virtualization, and Non-SQL Databases. New devices on an IoT platform need to be implicitly easy to define and set up, with no limit on the type of devices. As a consequence, there is no room for the inherent rigidity of SQL databases and the development lag to change the database schemes. The cloud platform to be used for the implementation of the proposed architectures relies on nonrigid SQL databases and flexibility in terms of data definition.

(2) Data Scalability. The data set of an IoT platform in a Smart City environment is predestined to grow on a fast pace, especially when monitoring environmental data, so the platform should have the ability to manage the warehouse of data using culling and archiviation methods, preferably in a fully automated manner. Data scalability for the proposed IoT platform is achieved by using cloud-based implementation.

(3) Component Reusability. In a Smart City environment, even though the incoming data may come from a variety of sources depending on the various application scenarios, the usage of this data is often very similar. Therefore, an IoT platform should allow at a large extent the reuse of the created components with minor adaptations. The proposed solution, through its layered architecture, gives the developers the possibility of building templates of objects and services available also to other uses different from the current one.

3. IoT Architecture for a Sustainable Tourism Application

As mentioned in the Introduction, the goal of our work is to present an IoT architecture for the specific Smart City scenario dedicated to the sustainable management of the tourist flow in the urban environment of Cagliari. To support the deployment of major applications, such as the efficiency of urban operations and services and the sustainable tourism, and to satisfy their quality requirements, an IoT architecture and its implementation as an IoT platform need to be able to address the previously listed requirements. To deal with these stringent requirements, we envision that, through the use of virtualization technologies, every real-world object participating to the Smart City, from smartphones to traffic lights and sensors, is associated with its virtual counterpart in the cloud. Associating a digital counterpart to physical objects is a common practice in the latest IoT research activities [12, 13], since through virtualization the physical devices gain augmented capabilities and are able to [12] (i) fully describe their characteristics with semantic technologies and then be able to interact with other virtual objects; (ii) identify, analyse, and manage the context related to an object surroundings and take decision accordingly; (iii) simplify the search and discovery of services and devices, which continuously join, move, and leave the network.

3.1. Proposed Architecture. The main purpose behind the proposed architecture, shown in Figure 1, is to activate the data-stream from IoT objects when needed. These data-streams are continuously processed to support end-users/ICT applications, with a set of procedures monitoring the context and producing alerts when particular conditions are met.

The proposed IoT platform foresees a four-level architecture: the highest level is the *Application Layer* in which user-oriented macroservices are deployed; the *Service Layer* is responsible for receiving the service requests and mapping them to the atomic services available in the lower layer; this layer is the *Virtualization Layer*, which interfaces directly with the real world and enhances objects' functionalities; the last level is the *Real-World Layer*, containing the real physical devices of the Smart City. Two additional cross-layers are needed to manage the quality requirements of the applications and to ensure that every communication takes place in a trustworthy and secure way, according to the previously listed requirements.

Even if this architecture is at the beginning of its implementation, we are moving towards implementation using Platform as a Service (PaaS) features, since they provide the tools needed to develop, run, and manage web applications, such as the execution containers of web services and related databases for data storage (see the Virtualization Layer subsection).

This approach has manifold motivations: (i) it enables objects to "speak" the same language at virtual level; (ii) it enhances the search and service discovery; (iii) it decouples the service requests and the actual IoT objects which satisfy the request; and (iv) it offers personalized experience to users based on their own needs and traits.

In the following paragraphs, we describe in detail the layers proposed for the architecture.

(1) Application Layer. At this level, user applications are responsible for final processing and presentation of the results. Deployment and execution of application make use of one or more services. An application at this level shows a back-end interface to the underlying layers. Additionally, applications can show a front-end interface so that requests can be triggered by users or by the objects themselves. A user interface is provided for the interaction with the system; however, in this case, the requests need to be translated in a language understandable by the platform.

(2) Service Layer. This layer is responsible for the analysis of the service requests generated by the application in the above layer and to enhance this request with a range of facts concerning the human user, including user context, profile, preferences, and policies. Each service request contains a semantic description of the input and of the desired output.

In particular, the *Service Request Analysis* (SRA) receives the query and asks for the retrieval of the current situation in which the query is performed. Moreover, it must be able to reuse the output of a previous request to respond to requests that present the same inputs and similar user information in order to reduce the burden on the Virtualization Layer and save redundant data requests.

The *user characterization* provides all the information associated with the user or the user's object which made the query. This block comprises cognitive functionalities to build user-related knowledge and act on behalf of the user when needed. This is an important component in our platform because many applications in a Smart City scenario, such as the one for sustainable tourism, are characterized by personal choices: requests coming from different users can have different solutions.

However, user characterization is only related to static user information, so a *Context Awareness* block is needed to detect, recognize, classify, and act upon the particular situation the user is involved in. A tourist looking for a museum to visit can receive different recommendations based on the time of day, the distance from different museums, or the number of people in the queue waiting to visit it.

All the collected information is then forwarded to the *Decomposer*. This block has to decide which atomic tasks (sensing, actuation, and computational) compose the query in order for the Virtualization Layer to be able to search for the right objects through the activation of subqueries.

(3) Virtualization Layer. The Virtualization Layer is where the virtual counterpart of the physical objects, namely, the *virtual objects* (VOs), reside. VOs are digital representations of the service(s) of any entity in the IoT, which are usually described in terms of semantics. VOs are implemented as web services and each of them as a related database for the data storage of the information sent by the physical counterpart.

Moreover, at this layer, the physical objects are enhanced with capabilities which enable them to perform operations otherwise hampered on real objects. Traffic lights, proximity sensors, and road cameras can communicate without any problem at this level even if they all use different communication technologies: simple technologies, such as RFID tags and NFC, can be attached to Points of Interest (POIs) to enhance the visiting experience of tourists by interpreting information about the environment and making choices accordingly, in order to push information to users via smartphones or tablets when necessary [14].

The core of this layer is represented by the *Virtual Object Execution Space* (VOES), where all the instances of VOs run. In fact, whenever a new object is detected, based on the information it provides about its resources and functionalities, it is associated with a new virtual object instance in the VOES. This instance is chosen among the possible VO models from a template repository which matches the physical device information. The VO model includes objects' characteristics; objects' location; resources, services, and quality parameters provided by objects.

Every VO is composed of two parts: on one side, it interfaces with the physical object it is associated with. This way it is possible to introduce a standardized communication procedure between the platform and the extremely variegate set of physical devices, so that the VO can switch among multiple communication channels, such as HTTP (Hypertext Transfer Protocol) or CoAP (Constrained Application Protocol) based on the situations. On the other side, the VO is able to communicate with the other VOs in the VOES through the use of a common semantic.

Thanks to this, heterogeneous objects become interoperable at the virtual level, even if they are not at the real-world level, since all the VOs "speak" the same language regardless

of the properties of their physical counterparts. Moreover, since all the data provided by objects are indexed making use of a common language, the discovery of services is simplified.

The metadata about each of the active VOs is maintained in the VO registry, which stores the semantically enriched data that are used for the description of the VOs, in order for them to be available anytime from anywhere.

Whenever the *Search and Discovery Engine* receives a request from the upper layer to find one or more services, this request is matched into VO template names for which VO instances are searched. This module then performs a search in the VO registry in order to discover potentially available and relevant VO instances of the requested VO template names.

(4) Real-World Object Layer. The lowest level of the proposed architecture is made up of real-world objects (RWOs). A RWO can be any entity in the physical domain, human or lifeless, static or mobile, solid or intangible, which is represented in the virtual world through the use of ICT objects. The capabilities and functions of these devices are then described in the Virtualization Layer in terms of sensors, actuators, and resources exposed to the other VOs in the system.

(5) Trust and Security Engine. This layer focuses on the implementation of appropriate security procedures to guarantee reliable communication at every layer. For example, at the Application Layer, it determines the level of access of the apps that generated the requests and grant access accordingly; at the Virtualization Layer, it has to understand how the information provided by the VOs has to be processed in order to ensure that the resources obtained are trustworthy.

(6) QoE/QoS Manager. Quality management is particularly problematic in IoT scenarios due to heterogeneity and mobility issues. The proposed architecture addresses these issues due to the adoption of the VOs; however, a quality manager is still needed to assure that the overall acceptability of applications and services is guaranteed, considering both end-to-end communication requirements (QoS) and the users' expectations (QoE).

4. Use Case

In this section, we analyse how the IoT platform previously described can be used to implement a Smart City scenario specific for the sustainable movement of tourists in the city of Cagliari. The scenario considered in this paper is that of cruise tourists that arrive in Cagliari and have just a few hours to visit the city. The tourists wish to visit some of the city's POIs, in the limited time available before the cruise leaves. For this purpose, we suppose tourists are provided with a smartphone/tablet app where they are able to specify which POIs they wish to visit. The objective is to visit all the desired POIs in the shortest time possible. We detected two main elements that can cause delays to the tourists: (i) choosing the wrong mode of transport to get to the POI and (ii) getting to the POI at the wrong time, when a longer queue time is expected.

For this specific use case, we assume to have sensors placed at each POI's entrance (for those that have one, excluding, e.g., statues, squares), to measure queue waiting time. These sensors, to be physically implemented in the next phase of this work, are virtualized by the Virtualization Layer, so that they are easily interoperable with all the objects that participate to the proposed architecture. Data gathered by the queue sensors is sent to the Service Layer and processed to infer a queue model that associates times of the day to queue waiting times, for the POI under examination. This association is necessary in order to have an estimate queue waiting time at the beginning of the route planning operations.

In our use case, we consider a model based on the mean value of queue waiting times with reference to each time of the day and POI.

Based on the current and expected queue waiting time and on the time needed to get to the POI either by public transport or walking, the optimal route to visit all the selected POIs will be recommended to the tourist.

(1) The Tourist Route Planning Problem. The optimisation problem solved by the application is a typical Travelling Salesman Problem (TSP) [12]: the tourist is given an optimal route where he/she visits each POI only once, minimizing a cost that, in this use case, is considered to be the sum of the time wasted to get to the POI and on the queue.

Given a complete digraph $\mathcal{G}(N, E)$ where N is the set of nodes and E is the set of edges, we associate node $i = 0$ to the tourist starting point and node $i = \{1, \ldots, n\}$ to each POI that the tourist wants to visit. For each edge, we introduce a logical variable $x_{ijk} = \{0, 1\}$ that equals 1 if the tourist goes from POI i to POI j at time step k. Analogously, we introduce a logical variable $y_{ik} = \{0, 1\}$ that equals 1 if the tourist is visiting POI i at time step k.

The cost to get from POI i to POI j at time step k is defined as

$$c_{ijk} = c_{ijk}^{\text{path}} + c_{jk}^{\text{queue}}, \tag{1}$$

where c_{ijk}^{path} is the time needed to go from POI i to POI j at time step k and depends on the mode of transport used; c_{jk}^{queue} is the time spent on the queue at POI j at time step k.

The tourist starts from the starting point $i = 0$. Therefore, it is necessary that

$$y_{00} = 1. \tag{2}$$

Furthermore, the tourist visits each POI exactly once. This is expressed as

$$\sum_{k=1}^{K} y_{ik} = 1 \quad \forall i \in N \setminus \{0\}. \tag{3}$$

Finally, if the tourist visits POI i at time step k, then he/she has to leave it at time step $(k+1)$:

$$\sum_{j\in N} x_{jik} = y_{ik} \quad \forall i \in N \setminus \{0\},\ k \in \{1,\ldots,K\},$$
$$\sum_{j\in N} x_{ij(k+1)} = y_{ik} \quad \forall i \in N,\ k \in \{0,\ldots,K-1\}. \tag{4}$$

Summarizing, the problem to be solved can be formalized as follows:

$$\begin{aligned}
\min \quad & \sum_{i\in N}\sum_{j\in N\setminus\{0\}}\sum_{k=1}^{K} c_{ijk}x_{ijk} \\
\text{s.t.} \quad & y_{00} = 1 \\
& \sum_{k=1}^{K} y_{ik} = 1 \quad \forall i \in N \setminus \{0\} \\
& \sum_{j\in N} x_{jik} = y_{ik} \quad \forall i \in N \setminus \{0\},\ k \in \{1,\ldots,K\} \\
& \sum_{j\in N} x_{ij(k+1)} = y_{ik} \quad \forall i \in N,\ k \in \{0,\ldots,K-1\}.
\end{aligned} \tag{5}$$

The problem described by (5) is a binary programming problem, which can be solved using a branch-and-bound algorithm. This type of problem is NP-hard; that is, its complexity scales exponentially with the number of variables [14]. In our case, the number of variables is given by $|x_{ijk}| + |y_{ik}|$, with $i, j \in N$ and $k = \{1,\ldots,K\}$. Considering that at each time step the tourist visits one of the POIs, $K = N$ time steps are needed to visit all the POIs. Therefore, problem (5) complexity is proportional to $2^{(N^3+N^2)}$. Although it can appear high demanding for the computational resources involved, it has to be noted that this is just an upper bound that is rarely reached. Furthermore, it needs to be considered that the tourist has a limited amount of time to spend in the city, so he/she can only visit a low number of POIs. Nevertheless, should the problem still be considered too complex, heuristic techniques such as genetic algorithms or swarm intelligence may be used for the subsequent implementation of the entire system, planned in the next phase of this work.

The pseudocode of the Tourist Route Planning Problem described in this subsection is provided by Algorithm 1.

```
For i in POI_list then
    For j in POI_list then
        For k in range(1, POI_number) then
            For m in transp_mode_list then
                Retrieve path_time_ijk[m]
            End For
            path_cost[i, j, k] = min(path_time_ijk)
            queue_cost[j, k] = retrieve queue_time_jk
            Compute cost[i, j, k] according to eq. (1)
        End For
    End For
End For
Find all opt_route that solve eq. (5)
If there is more than 1 opt_route then
    Assign opt_route to the one that corresponds to
    the shortest path
End If
```

Algorithm 1: Tourist route planning problem.

5. Simulations

The use case described in Section 4 has been tested by means of a Python code implemented in the Service Layer of the previously presented architecture. We opted for our own implementation using Python and not for the free tools offered by Google Maps and Bing Maps APIs because of the usage limitations of these solutions.

We considered 67 of the main POIs belonging to the city of Cagliari, including churches, museums, monuments, gardens, towers, and historic buildings. Each tourist can select from 2 to 8 POIs, which can be reached either by bus or walking: each transportation mode represents a different cost, expressed in number of minutes.

To test the behaviour of the application, the POIs have been selected randomly from the map. The starting point is set to the port of Cagliari. In order to evaluate the cost to get from one POI to the other, the Google Maps APIs have been used in an initial phase. The queue waiting times c_{ik}^{queue} have been set randomly from 0 to 30 minutes. To compute the right time step k at which the cost value c_{ijk} has to be computed, we also considered the time spent by users at each POI, supposing that they need an average of 45 minutes to visit each POI. The results obtained using the proposed application, called TRAPP hereinafter, have been compared to the following approaches:

(i) Minimization of the time spent to reach the POIs (called MIN PATH), without taking into consideration the time spent on the queue: this is the typical behaviour of common route planning apps that are not enhanced by an IoT platform.

(ii) Moving to the next closest POI using the bus whenever possible (called ONLY BUS): this is the typical behaviour of a tourist who does not feel like walking.

(iii) Walking to the next closest POI (called ONLY WALK): this is the typical behaviour of a tourist who does not feel like wasting time either waiting for the bus or on the wrong bus line or finding the right bus stop.

Figure 2 shows the mean time values of the overall time spent to get to the POIs and on the queue, the only time spent to get to POIs, and the only time spent on the queue, for the four approaches described above, for different numbers of POIs. Focusing on the overall wasted time, TRAPP outperforms all the compared approaches, enabling to save up to 21% with respect to MIN PATH, up to 44% with respect to ONLY BUS, and up to 55% with respect to ONLY WALK.

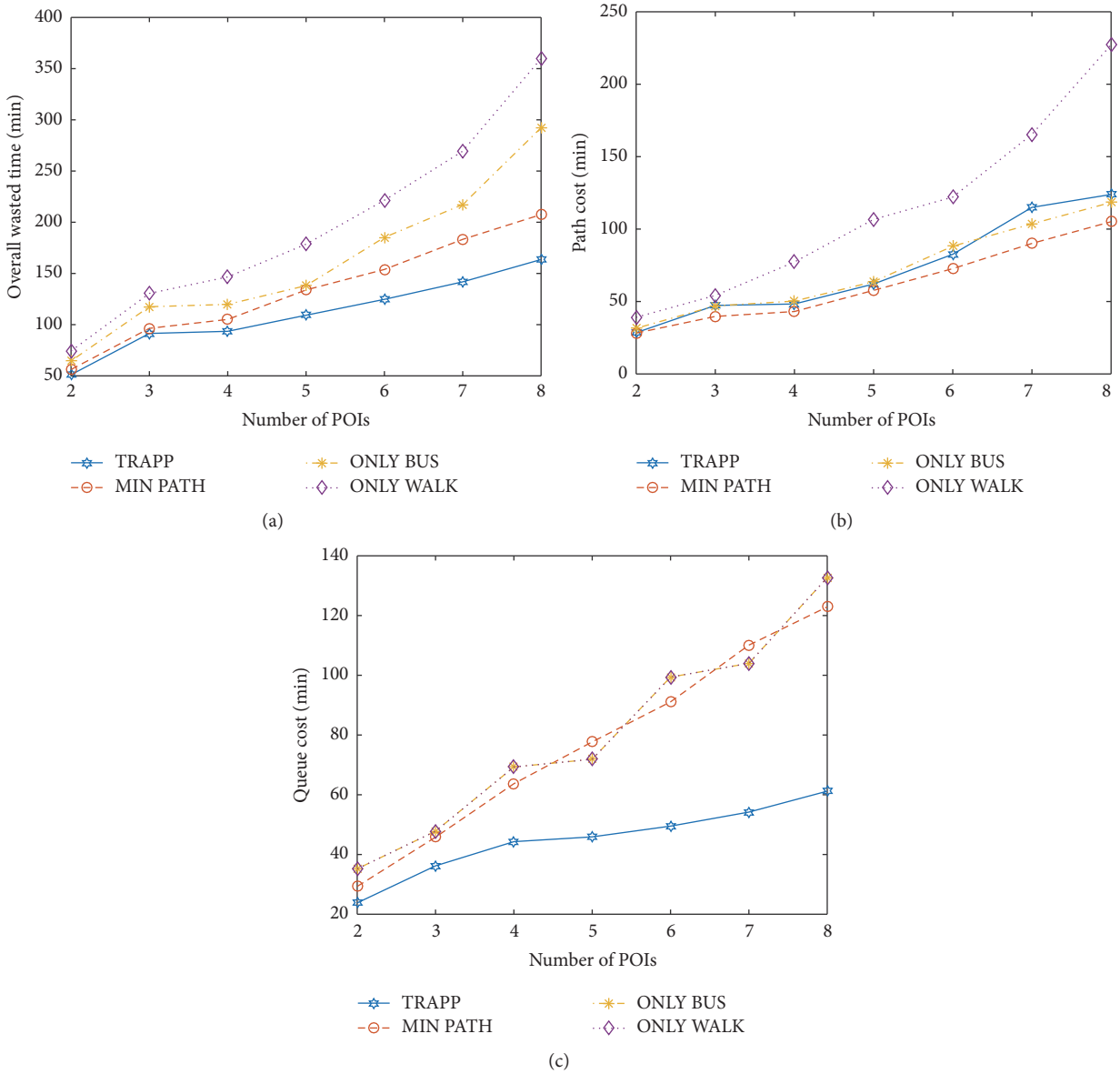

Figure 2: Mean time values for the overall wasted time, path cost, and queue cost for different numbers of POIs.

It has to be noted that, with reference to the queue cost, ONLY BUS and ONLY WALK approaches have the exact same values. This is because in both cases tourists' sequence of visited POIs is the same, as they are moving from one POI to the closest one, but using a different way to reach it.

It is interesting to note that, according to Figure 3, using TRAPP, the tourists' ratio between the distance covered by bus and the total distance is similar to that of MIN PATH when the number of POIs is low, but it gets closer to that of ONLY BUS when the number of POIs considered gets higher. Recall that ONLY BUS ratio is not 100% because it takes into account the distance covered by the tourist to get to bus stops and, in addition to that, sometimes there is no bus line between two POIs. On average, by using TRAPP, the tourist walks 44% of the time.

Cruise tourists usually spend from 6 to 9 hours in Cagliari. Considering this limited amount of time, tourists' satisfaction is higher if they are able to visit all the POIs they have planned to see. Accordingly, we define a satisfaction level given by the ratio between the number of POIs actually visited and the number of POIs that was planned to visit. Figure 4 shows the average tourist satisfaction level of visiting 8 POIs, considering an overall available time of 9 hours, supposing that they spend, on average, 45 minutes to visit each POI.

6. Conclusions

This paper proposes an IoT architecture for a sustainable tourism application in a Smart City scenario. The proposed architecture is tailored for the use case of an application for sustainable tourism. The presented IoT architecture is based

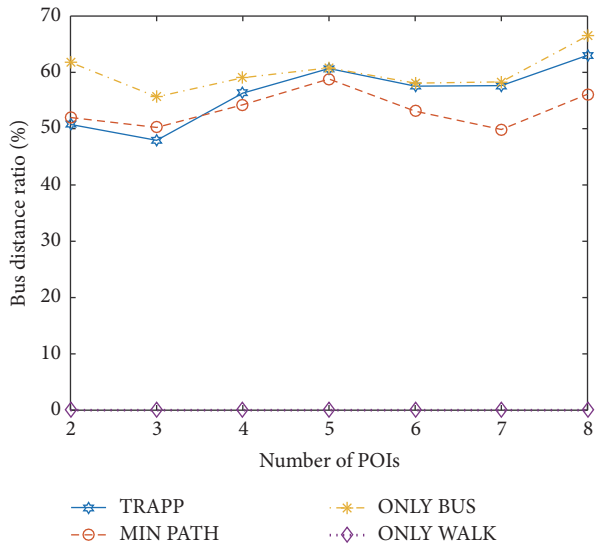

Figure 3: Mean percentage of distance covered by bus, with respect to the overall distance covered either by bus or walking.

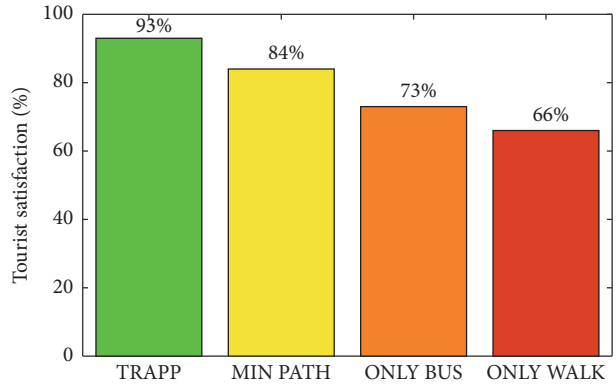

Figure 4: Average tourist satisfaction level to visit 8 POIs, for an overall available time of 9 hours.

on a specific set of requirements and can be implemented using a cloud-based IoT platform. The actual implementation is ongoing and is preceded by the proposed implementation of a specific use case for sustainable movement of cruise tourists that arrive by sea in the city of Cagliari, Italy, and have just a few hours to visit the city. The simulation of the presented use case includes a set of 67 POIs, information about the waiting times at the POIs, and transport information (walking and bus times and distances).

The simulation results evaluate the mean time values of the overall wasted time, the time wasted to get to POIs, the time wasted waiting on the queue, the distances covered by bus or walking, and the average tourist satisfaction level. The results showed that, with reference to other approaches, the proposed application enables tourists to save up to 55% of their time, with a satisfaction level up to 27% higher. Furthermore, the distance covered by bus rather than walking is comparable to that of the approach where tourists chose to use bus whenever possible, particularly for higher number of POIs.

Based on these encouraging first results, we already started the implementation of the proposed IoT architecture on a cloud-based IoT platform. The future work includes the completion of the implementation activities, the deployment of the virtual objects, including the queue and traffic sensors, and the introduction of new constraints to improve the Tourist Route Planning Problem (e.g., considering other modes of transport, user profiling, clustering techniques, and different weights of the cost elements of the problem, based on user preferences).

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research activities described in this paper have been conducted within the Research Project "Cagliari Port 2020" of the Italian Research Ministry Project Call "Quadro Strategico Nazionale 2007–2013, Programma Operativo Nazionale Smart Cities."

References

[1] N. Dlodlo, L. Montsi, and P. Mvelase, "Internet of things platforms in support of smart cities infrastructures," in *Proceedings of the 15th Annual Conference on World Wide Web Applications*, Cape Town, South Africa, September 2013.

[2] P. Peeters and G. Dubois, "Tourism travel under climate change mitigation constraints," *Journal of Transport Geography*, vol. 18, no. 3, pp. 447–457, 2010.

[3] G. Falconer and S. Mitchell, Smart City Framework: A Systematic Process for Enabling Smart + Connected Communities, Executive Report, Cisco Internet Solutions Group, 2012, http://www.cisco.com/web/about/ac79/docs/ps/motm/Smart-City-Framework.pdf.

[4] PROBE-IT Project, "D3.1a Roadmaps for IoT Deployments," November 2012, http://www.probe-it.eu/wp-content/uploads/2014/04/D3-1a_Roadmaps_for_IoT_Deployments_final.pdf.

[5] M. Hausenblas, Key Requirements for an IoT Data Platform, MAPR Blog, https://www.mapr.com/blog/key-requirements-iot-data-platform.

[6] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city internet of things platform with microservice architecture," in *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud '15)*, pp. 25–30, Rome, Italy, August 2015.

[7] D. Bonino, M. T. D. Alizo, A. Alapetite et al., "ALMANAC: internet of things for smart cities," in *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud '15)*, pp. 309–316, IEEE, Rome, Italy, August 2015.

[8] A. Cenedese, A. Zanella, L. Vangelista, and M. Zorzi, "Padova smart city: an urban internet of things experimentation," in *Proceedings of the 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '14)*, Sydney, Australia, June 2014.

[9] F. Paganelli, S. Turchi, and D. Giuli, "A web of things framework for RESTful applications and its experimentation in a smart city," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1412–1423, 2016.

[10] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, "Developing an IoT Smart City framework," in *Proceedings of the 4th International Conference on Information, Intelligence, Systems and Applications (IISA '13)*, pp. 180–185, Piraeus, Greece, July 2013.

[11] White Paper, *How To Select the Right IoT Platform*, Ayla Networks, Santa Clara, Calif, USA, 2014.

[12] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The virtual object as a major element of the internet of things: a survey," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1228–1240, 2016.

[13] C. Musu, V. Popescu, and D. Giusto, "Workplace safety monitoring using RFID sensors," in *Proceedings of the 22nd Telecommunications Forum (TELFOR '14)*, pp. 656–659, IEEE, Belgrade, Serbia, November 2014.

[14] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," in *Encyclopedia of Operations Research and Management Science*, pp. 1573–1578, 2013.