A Service-Oriented Middleware for Cloud of Things and Fog Computing Supporting Smart City Applications

Nader Mohamed[1], Jameela Al-Jaroodi[2], Sanja Lazarova-Molnar[3], Imad Jawhar[4], and Sara Mahmoud[5]

[1]Middleware Technologies Lab., Pittsburgh, Pennsylvania, USA
[2]Department of Engineering, Robert Morris University, Moon Township, Pennsylvania, USA
[3]Center for Energy Informatics, University of Southern Denmark, Denmark
[4]Midcomp Research Center, Saida, Lebanon
[5]College of Information Technology, UAE University, Al Ain, UAE
nader@middleware-tech.net, aljaroodi@rmu.edu, slmo@mmmi.sdu.dk, imad@midcomp.net, 201370014@uaeu.ac.ae

***Abstract*— Several cities around the world are seeking to adopt the smart city concept to enhance the living quality of their residents and optimize city resources. Smart cities heavily rely on utilizing a number of technologies to improve the performance and services in healthcare, transportation, energy, education, and many others, while reducing costs and resource consumption. One of the promising technologies to support such efforts is the Cloud of Things (CoT). CoT can provide a platform for linking the cyber parts of the smart city that are executed on the cloud with the physical parts of the smart city including residents, vehicles, power grids, buildings, water networks, hospitals and other physical resources. Another technology that can be utilized for enhancing smart cities services is Fog Computing, which extends the traditional Cloud Computing paradigm to the edge of the network to enable better support for operating enhanced services. However, proper integration and efficient utilization of CoT and Fog Computing for smart cities is not an easy task. The paper discusses how the service-oriented middleware approach can help resolve some of the challenges of developing and operating smart city services using CoT and Fog Computing. In addition, this paper proposes a service-oriented middleware, called SmartCityWare, for effective integration and utilization of CoT and Fog Computing for smart cities.**

Keywords- Smart City, Cloud of Things, Internet of Things, Middleware, Service-Oriented Middleware, Fog Computing

I. INTRODUCTION

Advances of technology in Cyber-Physical Systems (CPS), Internet of Things (IoT), Cloud Computing (CC), Fog Computing and other software technologies have positively contributed to the emerging concept of smart cities. These new technologies offer an unprecedented opportunity to create a wide array of applications that optimize smart cities' services. The technologies are integrated into what is termed as Cloud of Things (CoT) [1]. In CoT, all objects of smart cities like city residents, vehicles, streets, buildings, hospitals, and energy and water plants are interconnected through the IoT, which is integrated with cloud computing systems, running intelligent software to optimize smart cities' services. In addition, Fog Computing can offer extension features for cloud computing systems to better support low latency requirements, location awareness, scalability, and mobility for smart city services [2].

Smart city applications can be developed to effectively utilize available technologies and other emerging technologies in the future to continue enhancing the living quality of their residents, while optimizing utilization of city resources and reducing negative impact on the environment. These technologies include:

1. Cloud Computing Systems, to provide different computation and data storage services to smart cities [3][4][5],
2. Fog Computing Systems, to provide low latency support, location awareness, better mobility support, and streaming and real-time support for smart cities [6][7],
3. Wireless Sensor Networks (WSN), for monitoring of the different resources and situations of smart cities [8][9],
4. The Internet of Things (IoT), to integrate the physical objects in smart cities in a city network [10][11],
5. Cyber-Physical Systems (CPS), to facilitate the interaction between the cyber world and the physical world in smart cities [12],
6. Robotics, to provide different ground actions and physical controls for smart cities [13],
7. Unmanned Aerial Vehicles (UAV), to enhance delivery of services, traffic monitoring, security and safety controls, and telecommunication services in smart cities [14][15], and
8. Big Data Analytics (BDA), to provide smart decisions based on collected data for enhancing smart city services [16][17].

Integrating Cloud Computing with WSN, IoT, CPS, robotics, and UAVs would form Cloud of Things to support operations in smart cities. These operations can be further enhanced by utilizing Fog Computing [18]. The integration of the mentioned systems and others that will be developed in the future can provide a powerful environment for supporting operations of smart city applications. However, developing, implementing, maintaining, and operating these applications in an effective manner are major challenges. This paper investigates utilizing a service-oriented middleware approach to relax the challenges. We propose SmartCityWare, a service-oriented middleware for integrating CoT and Fog Computing to support the development and execution of smart city applications. This approach will provide a meaningful representation and utilities to design and implement such services.

This paper is organized as follows. Section II provides background information about smart city applications, CoT, and Fog Computing. Section III discusses using service-oriented middleware for smart city applications. A conceptual design and the functions of SmartCityWare are provided in Section IV. Section V illustrates an example smart city application of using SmartCityWare. A prototype implementation and experimental evaluations are discussed in Section VI, Section VII is an overview of some related work, while Section VIII concludes the paper.

II. Background

A smart city concept involves monitoring, controlling, and managing the conditions of all of the city's critical infrastructures, like power, communication, water, roads, vehicles, buildings, hospitals, bridges, tunnels, subways, seaports, and airports to optimize operations and use of resources, while providing high quality services to the citizens [19][20]. This requires using Information and Communication Technologies (ICT) to optimize these services. It is often stated that the goal of utilizing ICT is to improve existing services by making them more efficient, more user-friendly or, in general, more citizen-centric. With the recent advances in ICT, all critical infrastructures can be integrated, monitored, and controlled for the benefit of the citizens. One of the emerging technologies that effectively can be used for smart cities is the CoT. The CoT helps connect, monitor, and apply enhancements in all aspects of the cities through the IoT and Cloud services as shown in Figure 1. The CoT provides powerful cloud platforms to run optimization models to make accurate and efficient decisions for smart city services. CC provides a flexible virtual execution system and on-demand services for smart cities. It can process and store huge data sets and offer dynamic computing capabilities that can be scaled in or out based on the varying demands of the smart city services. The CoT can be implemented with a multiple-layer model (see Figure 1). One of the most important layers is the CoT platform as a service. This layer links the IoT and the CC infrastructure and services and provides services to implement and operate optimization applications for smart city services.

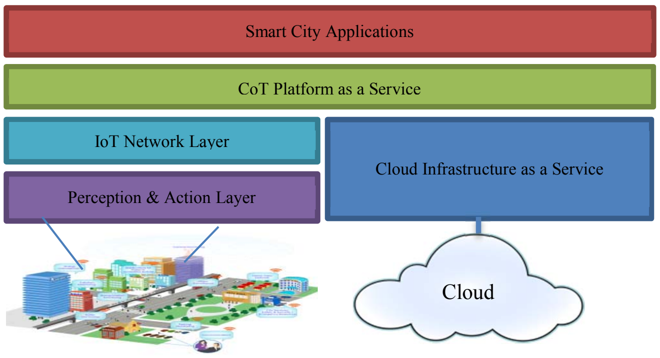
Smart City Applications
CoT Platform as a Service
IoT Network Layer
Perception & Action Layer
Cloud Infrastructure as a Service
Cloud

Figure 1. The CoT Model for Smart Cities.

Fog computing can enhance the CoT paradigm by providing small platforms located at the network edges in smart cities. These platforms can operate cloud-like services to support different IoT operations. The services can be control, storage, communication, processing, configuration, monitoring, measurement, and management services to support a certain IoT smart city application. Using Fog Computing, an application in a certain area in a smart city can utilize an architecture that uses a dedicated computer available locally, or one or more end-user clients or nearby edge devices. The Fog platform will allow executing services geographically close to the IoT applications. This provides a number of advantages for IoT applications including [2]:

- Providing low latency services
- Offering location aware services
- Providing better scalability support for widely geographical distributed applications
- Supporting better mobility and access control
- Offering better Quality of Services (QoS) support
- Providing more efficient communication with other systems

These advantages help create solutions to many challenges that smart city applications face and enable creation of higher quality and more controllable services to perfectly achieve the vision of a smart city. The architecture of integrating Fog computing into CoT for smart city is shown in Figure 2. In this architecture, the fog will provide more localized real-time monitoring, control, and optimization for smart city applications while the cloud will provide global monitoring, control, optimization, and future planning for smart city applications. Table 1 lists smart city applications that can benefit from both CoT and fog computing. More discussion about how CoT and fog computing can support these applications is discussed in [41].

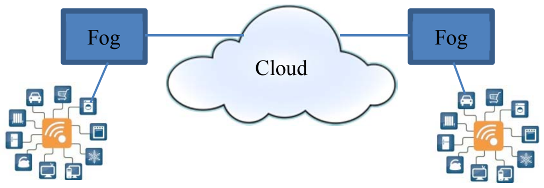

Figure 2. Integrating Fog Computing into CoT for Smart City.

Table 1. Smart city applications that can benefit from CoT and fog computing.

Smart City Application	**Sub-application**
Intelligent transportation	Route planning and congestion avoidance
	Intelligent traffic light controls
	Intelligent parking services
	Accident avoidance
	Self-driving buses/cars
Smart energy	Smart grid
	Smart buildings
	Renewable energy plants
	Smart meters
	Wind farms
	Hydropower plants
Smart water	Leakage detections
	Water leakage reduction
	Water quality monitoring
	Smart water meters
	Smart irrigation
City structure health monitoring	Bridges health monitoring
	Large public buildings health monitoring
	Tunnels health monitoring
	Train and subway rails health monitoring
	Oil and gas pipelines health monitoring
Environmental monitoring	Air quality monitoring
	Noise monitoring
	River monitoring
	Coastal monitoring
Public safety and security	Crowd control for large events (sports games, parades, and outdoor celebrations)
	City crime watching and alerting
	Large-scale emergency response services in disasters (floods, large-scale terrorist attacks, earthquakes, volcanoes, bushfires, and wars)

III. Service-Oriented Middleware

Middleware platform technologies have become a necessary part of any distributed and complex applications and systems [21]. Middleware can provide needed functionalities for facilitating the integration of the subsystems and the operation supports of the whole distributed and heterogeneous applications and systems. It simplifies the implementation, deployment, and execution of distributed applications and hides their complexity. It also provides common services for repeated challenges in the distributed environment. Middleware also connects any set of software and hardware components and other devices in a distributed system to provide improved functionalities. These components could be hardware devices such sensors, actuators, robots, UAVs, communication devices, microcontrollers, cloud servers; or software components including control modules, monitoring applications, analytics services, and application specific software modules. Improved functionalities can be related to communication, operation, reliability, availability, scalability, security, and other value-added purposes.

Smart cities are very large distributed and complex systems that share with other distributed systems their heterogeneity, security, and reliability issues. In addition, they also have their own unique issues such as high scalability, safety, real-time operations and responses, and smartness requirements. These are common issues and challenges facing most smart city services including smart grid, smart water networks, intelligent transportations, and others. Therefore, it is almost impossible to design, implement, execute, and control smart city services without depend on advanced middleware platforms to enable the development and operations processes.

An innovative method in middleware technologies is the use of service-oriented middleware (SOM). This method has been recognized to shorten development and smooth operations of a number of applications in various domains [22]. Similarly, SOM can be used for developing and operating smart city services. SOM logically views smart city software, hardware and physical components as a provider for a set of services for smart city services. Moreover, it can integrate these services with other services offered by Fog Computing and Cloud Computing to allow application developers to view everything as a single large system that provides a number of services to smart city services. Generally, a SOM for smart city services should implement a number of requirements, some of which are shared for any SOC application, while the rest are imposed by the characteristics of the smart city environment and the issues of developing and executing applications on CoT and Fog Computing. These requirements include:

1. As all devices in the smart city including components of supported cloud computing and fog computing are viewed as a set of services available to support the applications, SOM should offer mechanisms to load, deploy, and execute these services as required.
2. Support for different efficient communication methods among service consumers and producers, and brokers that enable reliable local and remote services consumption.
3. SOM should enable client applications to discover and use registered services.
4. SOM should allow client applications to transparently use registered services without exposing services detailed implementation or, in some cases, their locations.
5. Offering appropriate abstractions to hide the heterogeneity details of the smart city's resources. SOM should offer high-level interfaces to utilize cyber and physical smart city resources without requiring application developers to deal directly with the heterogeneity.
6. SOM should provide mechanisms for client applications to configure smart city services to meet specific applications requirements such as QoS, security or reliability.
7. As smart cities can be very dynamic environments where resources can be added, removed, or changed, SOM should provide self-management mechanisms such as self-configuration, self-healing, and self-optimization of service providers.
8. Many smart city applications involve large volumes of data and high communication loads as they operate in data-rich environments and handle a large amount of data generating services. As some of the used systems and devices may have limited resources, SOM should efficiently and carefully deal with that load through trade-offs between smart city applications requirements in terms of accuracy, bandwidth, delay, and energy consumption.
9. As most smart city applications involve sensitive and critical information, SOM should provide mechanisms to secure the utilization and operations of both CoT and Fog Computing services.
10. Most interactive smart city services have specific QoS requirements. Mechanisms are needed to configure and satisfy these requirements in the CoT and Fog computing systems utilized by these applications.
11. As smart city applications usually do not operate in isolation, SOM should enable integration of smart city applications with other systems such as enterprise or web systems.

In addition to these requirements, there are other requirements that are needed for some smart city services. Examples are support for context awareness and location-based services. SOM is usually developed to best suit the underlying environments it will serve. Therefore, many of the existing SOM designs may provide some of the requirements described above, but most do not support all smart city services requirements and do not offer comprehensive solutions for their issues.

IV. SmartCityWare Architecture

This section defines the architecture, components, and main functions of SmartCityWare, a service-oriented middleware that is developed to utilize CoT and Fog Computing to enable and support smart city services. The main drive of SmartCityWare is to offer a virtual environment to be utilized to develop, implement, and deploy smart city services. In SmartCityWare, all functions are seen as a set of services that can be used to build and support the execution of diverse smart city applications. These services are classified into core services and environmental services. Core services are those designed specifically for the core operations of SmartCityWare, such as the broker, security, service invocation, and location aware services. These services offer the main management and control mechanisms for the entire system. Environmental services deliver access to services offered by one or more cloud service providers, services provided by one or multiple distributed fogs, and services provided by multiple IoT devices including sensors, WSN, actuators, cameras, cars, UAVs, robots, etc. Cloud services can be Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software a Service (SaaS), which can define different services for smart city applications including data mining, big data analytics, optimization, and simulation services. Fog services can be storage, caching, streaming, processing, configuration, monitoring, measurement, and management services. The IoT devices services provide interfaces to utilize device functionalities and they can be sensing, action, or other services. The environmental services can either offer direct interfaces to access the original services, provided by a cloud, a fog, or IoT device or they introduce some added-value for the original services such as adding quality of service, reliability, availability, and security features. While some IoT devices can execute on-board code to implement and provide some functions, other IoT devices will be controlled mainly by a fog that will have services that provide interfaces to utilize these devices functionalities. SmartCityWare services can be used by smart city applications available on the cloud, fog, or IoT device such as a car asking for a certain service from a smart city application available on the cloud.

The main functions of SmartCityWare are to enable smooth integration and operations among all these units to effectively support smart city applications. The SmartCityWare services will be distributed among multiple clouds, fogs, and IoT devices as shown in Figure 3. Each service defines a few simple interfaces that make it available to other services. Using SOM concepts for SmartCityWare provides mechanisms to link available services to build new services. A specific subset of the available distributed services in SmartCityWare can be integrated and deployed for the accomplishment of a specific application in smart city. In addition, any service can be a client or consumer of other services. The required services for a certain application are integrated using SmartCityWare that aims to achieve loose coupling among interacting services. SmartCityWare manages service advertisement and discovery, communications, and invocations. In addition, it can be used to implement collaborative services across multiple smart city applications and with other SOC type systems.

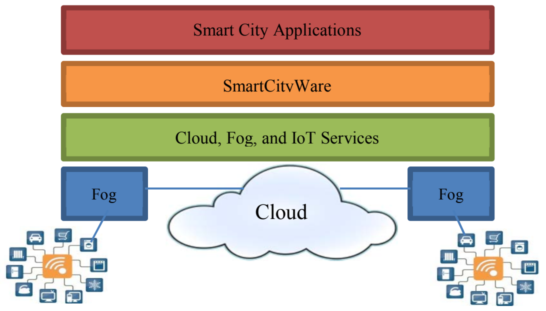

Figure 3. SmartCityWare Supporting Smart City Applications.

Although SmartCityWare can include many core services, the main mandatory services are the broker services, invocation services, location-based service, and security services. These four services are essential to ensure effective utilization of other available services and to facilitate the main functions it offers.

A. Broker Services

Broker services are responsible for CoT, fog computing, and IoT devices services advertisement, discovery, and registration. All services in all participating platforms and devices are advertised, registered, and discovered using the broker services. There are two types of broker services: a global broker that is available on the cloud, and local broker services that are available in each fog in the environment. While the global broker service maintains information about all services in the environment, the local broker service in each fog only maintains information about the current available services within the fog and information about services provided by IoT devices currently associated with the fog. This approach is used to allow applications and services within a fog to utilize available services and resources and provide low latency responses. As a result, the time needed to discover services is minimized to efficiently and accurately utilize the services. The global broker service maintains services information for the whole environment including services available in all fogs. The local fog brokers regularly update the global broker about the availability and status of their services. The fog service brokers maintain description information about local services. The service description information includes the standard information using Web Service Description Language (WSDL) for each defined service. Each service information includes the operations that the service can perform, the specific types and formats of the message it expects between the service provider and service consumer, and where the service can be located on the network.

B. Invocation Services

Invocation Services, local and remote, are invoked by consumers with SmartCityWare support. Remote service invocation can be between a fog service and another fog service, between an IoT device service and a fog service, between a fog service and a cloud service, or between a cloud service and a fog service. SmartCityWare handles message addressing, data marshaling and demarshalling, establishing communication connections between service consumers and producers, delivering requests and responses, and executing services.

C. Location Based Services

SmartCityWare can provide location-based services. Unlike regular service brokers that are used over the Internet, SmartCityWare service brokers for fog maintain additional information about the current positions of currently connected IoT mobile devices. The main reason for maintaining current positions is that some of their services can be considered useful and may be utilized only if the provider device is in a specific location; otherwise, there is no usefulness in utilizing that service. One example is using a sensor service on a robot if the robot is available within a specific location. A service consumer in a fog can look up a certain service within a specific location through its fog service broker. If this service is available within the fog's range then WSDL information about the service is sent to the service consumer to utilize the service using a local service invocation. Otherwise, the fog service broker will forward the lookup request to the global service broker available on the cloud and if the service is available then the service consumer utilizes the service using a remote service invocation.

D. Security Services

Various security mechanisms can be used by several clouds, fogs, and IoT devices in smart city applications. The main functions of security services in SmartCityWare are to integrate and regulate security mechanisms among all these components and ensure that the needed security measures are applied appropriately to protect the smart city applications, provided services, and the physical environments. The security services include authorization and authentication services and access control services for the smart city applications, SmartCityWare services, and the physical environment. These services can be provided with varying levels of protection measures, such that different applications can use the suitable set for their security requirements.

V. Application Example

This section explains how to utilize SmartCityWare for building and operating smart city applications. The example application is an intelligent traffic light control system in a smart city. SmartCityWare can be utilized to develop and support intelligent traffic light controls, which include monitoring devices across multiple locations to correctly forecast traffic patterns and regulate traffic lights to enhance flow. In addition, this type of application can gain from vehicle-to-vehicle communication, vehicle-to-infrastructure communication, and global positioning systems to collect more detailed views of the traffic situations. This additional information can be utilized to achieve global urban traffic control optimizations. This type of smart city application will decrease traffic delays, minimize vehicles travel times, improve vehicles average velocity, and enhance the prioritization for emergency vehicles' movements in urban areas. Learning, data mining, and adaptation algorithms are usually used in such applications to enhance urban traffic control optimization. Examples of such projects are discussed in [23], [24], and [25].

Intelligent traffic light control systems can be designed, implemented, and operated using SmartCityWare. A fog can be used for each traffic light to execute a number of services and to provide local optimization. All participating devices within the area of range of the traffic light such as the vehicles, road sensors, communication devices are viewed as services used to observe the current status of the traffic in that area. The fog will use all this information to make informed optimization decisions about the particular traffic light. However, a set of services on a cloud can observe the global traffic status within the smart city by monitoring the status of the fogs operations and local conditions. The cloud services will use this information to provide global optimizations for reducing traffic delays, enhancing vehicles' flow, and prioritizing emergency traffic on a larger scale covering the whole urban locations. These services can periodically collect the status of traffic in a smart city, evaluate and optimize the traffic, and configure the fog services accordingly. In addition, cloud services can provide coordination mechanisms among multiple fogs to coordinate multiple traffic lights available close to each other or available in an important street in a smart city. As traffic conditions change, all fogs will adjust their operations using the locally collected information and the global view obtained from the cloud services.

VI. Prototype Implementation and Experiments

A SmartCityWare prototype was implemented using a distributed Java agents environment [34][35]. This environment provides a middleware infrastructure that enables implementing different distributed programming models and advanced middleware services for distributed heterogeneous environments. These heterogeneous environments may consist of multiple heterogeneous nodes with different capabilities. In addition, this middleware infrastructure provides scalability solutions to deploy, operate, monitor, and control large-scale distributed applications. Each agent in the environment can provide runtime support for securely executing some code. This makes this middleware infrastructure very suitable to build other advanced middleware services for integrating the cloud with its powerful resources with one or more fogs that usually have limited resources. For the prototype implementation, most of the core services of SmartCityWare mentioned in Section IV were implemented. These include broker services, invocation services, and location-based services. Two types of broker services were implemented: the global broker that will be deployed on the cloud and the local brokers that will be deployed on all fogs. A distributed process to update the global broker, the cloud broker, with new information from the local brokers, fogs, is executed periodically every 30 seconds. This update may include adding a new service, removing a service, or changing the location of a service. A mechanism is also implemented to allow a local broker to forward a service search request to the global broker if it does not have the requested service. Both local and remote service invocations were also implemented and added to the prototype implementation.

For the IoT side, we used the Arduino board [36] which is open source hardware for embedded systems. For this prototype implementation, the Arduino was used as the IoT payload subsystem that is the onboard device requesting services. Some sensors were connected to the Arduino such as DHT11 sensor [37] for temperature and humidity measurements. Additionally, some LEDs and a buzz were installed to represent actuators. We connected an Adafruit CC3000 Wi-Fi board [38] to link the Arduino to a local area network that has a fog. The Arduino code was developed using the Arduino IDE [39] with the Adafruit CC3000 library [40]. Each IoT service was developed with a RESTful API.

At the fog side, there is a service that represents each sensor or actuator attached to the Arduino. The main function of these services is to map and bridge a call from the SOAP APIs to RESTful APIs. All sensor and actuator services are registered with the local broker. In addition, the global broker is periodically updated with these services.

For the experiments, we use three computers; one represents the cloud while the other two represent two fogs. In addition, we used WAN emulators among these machines to introduce the effects of using the Internet to connect the machines. An experiment was conducted for a local IoT service call (Local) within the corresponding fog machine, a remote IoT service call from the cloud machine (RemoteCloud), and a remote IoT service call from another fog (RemoteFog). The experiment was repeated for two types of services. The first service is to get the current temperature (CurTemp) while the second is to turn on the LED (LEDon).

The average results of 10 runs of multiple service calls are shown in Figure 4. These recoded times for these calls do not include the service lookup times. The response time for a service call from fog to fog and from cloud to fog are similar as the service call that is directly done between the client fog and the server fog without involving the cloud. The cloud will be involved only during the service lookup process. After that, the service can be directly called.

The average service lookup time for local services (Local) and remote services (Remote) are shown in Figure 5. As shown, there is a big difference between local and remote service lookup times. The local services can look for and utilize the local fog services and local IoT device services faster. This enables having low latency services supported by the available fogs for large-scale IoT applications. At the same time, local services can utilize services at the cloud or at other fogs including their IoT device services. Any cloud service can also utilize any services available at any fog.

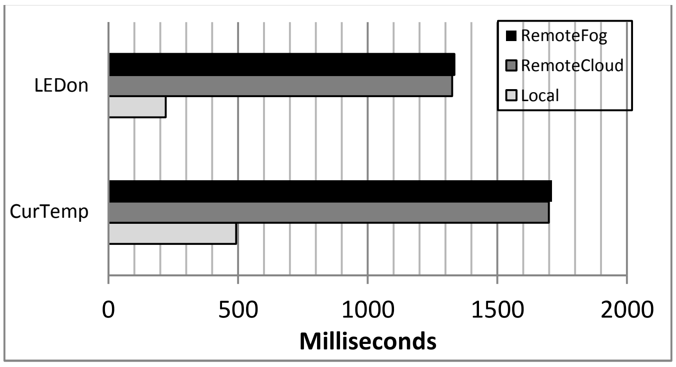

Figure 4. Service call response times.

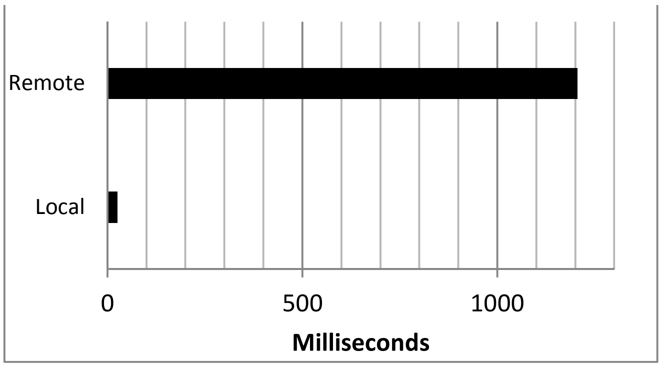

Figure 5. Service lookup times.

VII. Related Work

A number of middleware platforms were proposed and developed to solve different challenges in smart cities. Some examples of these middleware platforms are Civitas [26], SOFIA [8], VITAL [27], SmartUM [28], SMArc [29], GAMBAS [30], and CityHub [31]. One of the main differences between SmartCityWare and other proposed middleware platforms is that SmartCityWare is a completely service-oriented middleware that utilizes both emerging CoT and Fog Computing to provide different services for smart cities. The representation of all components and tools as services allow for a smooth integration of services using a common integration and deployment mechanism. An advantage of using service-oriented middleware is allowing the middleware to be open for unlimited extensions including utilizing services provided by other systems as well as integrating and utilizing future developed technologies as part of the middleware.

Different engineering issues such as developing generic computational models for smart city platforms and the difficulty of developing of a common platform for smart cities were discussed in [32][33]. Our work in this paper utilizes the flexibility and extensibility of service-oriented architecture to add the needed flexibility and extensibility to the SmartCityWare middleware platform. As a result, SmartCityWare can be built using current available technologies, yet it can evolve over time to include more services and utilize state-of-the-art algorithms, tools and models as they are developed.

VIII. Conclusion

This paper discussed the functions and features needed by a service-oriented middleware for building and operating smart city applications. Based on these functions, a service-oriented middleware that integrates and utilizes cloud of things and fog computing and provide a set of services to support smart city applications was proposed. This middleware is named SmartCityWare. In SmartCityWare, all system resources are viewed as a set of services to be used to develop smart city applications. One of the main advantages of this approach is the flexibility feature of extending the middleware itself to include new and more advanced services to support smart city applications as they develop. In addition, it provides the flexibility to add more devices, components, and services as the city grows or more services are needed. In addition, the proposed middleware can be easily extended to utilize new emerging technologies, other than Cloud of Things and Fog Computing, for supporting smart city applications in the future.

References

[1] P. Parwekar, "From internet of things towards cloud of things," In 2nd International Conference on Computer and Communication Technology (ICCCT), pp. 329-333, IEEE, 2011.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," In Proc. of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16, ACM, 2012.

[3] T. Clohessy, T. Acton, and L. Morgan, "Smart City as a Service (SCaaS): a future roadmap for e-government smart city cloud computing initiatives," In proc. of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 836-841, 2014.

[4] S. Yamamoto, S. Matsumoto, and M. Nakamura, "Using cloud technologies for large-scale house data in smart city," In IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 141-148, 2012.

[5] Z. Khan, and S.L. Kiani, "A cloud-based architecture for citizen services in smart cities." In Proceedings of the 2012 IEEE/ACM fifth international conference on utility and cloud computing, pp. 315-320, IEEE, 2012.

[6] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, Q. and Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," In proc. of the ASE BigData & SocialInformatics, 2015.

[7] A. Giordano, G. Spezzano, and A. Vinci, "Smart Agents and Fog Computing for Smart City Applications," In International Conference on Smart Cities, pp. 137-146, Springer International 2016.

[8] 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 4th International Conference on Sensor Technologies and Applications, pp. 281-286, 2010.

[9] T. Watteyne and K.S. Pister, “Smarter cities through standards-based wireless sensor networks,” IBM Journal of Research and Development, 55(1.2), pp.7-1, 2011.

[10] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” IEEE Internet of Things journal, 1(1), pp.22-32, 2014.

[11] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, “An information framework for creating a smart city through internet of things,” IEEE Internet of Things Journal, 1(2), pp.112-121, 2014.

[12] L. Gurgen, O. Gunalp, Y. Benazzouz, and M. Gallissot, “Self-aware cyber-physical systems and applications in smart buildings and cities,” In Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1149-1154, EDA Consortium, 2013.

[13] G. Ermacora, S. Rosa, and B. Bona, “Sliding autonomy in cloud robotics services for smart city applications,” In Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, pp. 155-156, ACM, 2015.

[14] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, I. Jawhar “UAVs for Smart Cities: Opportunities and Challenges,” in proc. Int’l Conference on Unmanned Aircraft Systems (ICUAS'14), IEEE, pp. 267-273, 2014.

[15] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar “Opportunities and Challenges of Using UAVs for Dubai Smart City,” in proc. 1st Int’l Workshop on Architectures and Technologies for Smart Cities (SmartCity’2014), IEEE Communications, 2014.

[16] R. Kitchin, “The real-time city? Big data and smart urbanism,” GeoJournal, 79(1):1–14, 2014.

[17] E. Al Nuaimi, H. Al Neyadi, H., N. Mohamed, and J. Al-Jaroodi, “Applications of big data to smart cities,” Journal of Internet Services and Applications, 6(1), p.1., 2015.

[18] M. Aazam and E.N. Huh, “Fog computing and smart gateway based communication for cloud of things,” In International Conference on Future Internet of Things and Cloud (FiCloud), pp. 464-470, IEEE, 2014.

[19] S. Dirks, C. Gurdgiev, and M. Keeling, "Smarter cities for smarter growth: How cities can optimize their systems for the talent-based economy," IBM Institute for Business Value, 2010.

[20] T. Bhattasali, et al,.”Secure and trusted cloud of things,” In 2013 NDICON, pp. 1-6, IEEE, 2013.

[21] J. Al-Jaroodi and N. Mohamed, “Middleware is STILL Everywhere!!!,” in Concurrency and Computation: Practice and Experience, Wiley, 24(16): 1919-1926, Nov. 2012.

[22] J. Al-Jaroodi and N. Mohamed, “Service-Oriented Middleware: A Survey,” in The Journal of Network and Computer Applications, Elsevier, Vol. 35, No. 1, pp. 211-220, Jan. 2012.

[23] A.A. Salkham, R. Cunningham, A. Garg, A. and V. Cahill, “A collaborative reinforcement learning approach to urban traffic control optimization,” In proc. of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Volume 02, pp. 560-566, , 2008.

[24] I. Arel, C. Liu, T. Urbanik, and A.G. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control,” IET Intelligent Transport Systems, 4(2), pp.128-135, 2010.

[25] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto,” IEEE Transactions on Intelligent Transportation Systems, 14(3), pp.1140-1150, 2013.

[26] F.J. Villanueva, M.J. Santofimia, D. Villa, J. Barba, and J.C. López, “Civitas: the smart city middleware, from sensors to big data,” In 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 445-450, IEEE, 2013.

[27] R. Petrolo, V. Loscrì, and N. Mitton, “Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms,” Transactions on Emerging Telecommunications Technologies, 2015.

[28] H.S. Jung, C.S. Jeong, CY.W. Lee, and P.D. Hong, “An Intelligent Ubiquitous Middleware for U-City: SmartUM,” Journal of Information Science & Engineering, 25(2), 2009.

[29] J. Rodríguez-Molina, J.F. Martínez, P. Castillejo, and R. de Diego, “SMArc: a proposal for a smart, semantic middleware architecture focused on smart city energy management,” International Journal of Distributed Sensor Networks, 9(12), p.560418, 2013.

[30] W. Apolinarski, U. Iqbal, and J.X. Parreira, “The GAMBAS middleware and SDK for smart city applications,” In IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 117-122, IEEE, 2014.

[31] R. Lea and M. Blackstock, “City hub: A cloud-based iot platform for smart cities,” In 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 799-804, IEEE, 2014.

[32] S. Pradhan, A. Dubey, S. Neema, and A. Gokhale, “Towards a generic computation model for smart city platforms,” In 1st International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE), 2016.

[33] M. Lehofer, M. Heiss, S. Rogenhofer, C.W. Weng, M. Sturm, S. Rusitschka, and S. Dippl, “Platforms for Smart Cities–connecting humans, infrastructure and industrial IT,” In 1st International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE), 2016.

[34] J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, “Middleware Infrastructure for Parallel and Distributed Programming Models on Heterogeneous Systems,” in IEEE Transactions on Parallel and Distributed Systems, Special Issue on Middleware Infrastructures, Volume 14, No. 11, pp. 1100-1111, Nov. 2003.

[35] J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, “An Agent-Based Infrastructure for Parallel Java on Heterogeneous Clusters,” in proc. 4th IEEE Int’l Conference on Cluster Computing (CLUSTER 2002), IEEE, pp. 19-27, Sep. 2002.

[36] Arduino website, https://www.arduino.cc/, viewed March 17, 2017.

[37] DHT Sensor Library Website, https://github.com/adafruit/DHT-sensor-library, viewed March 17, 2017.

[38] CC3000 Wi-Fi board Website, https://www.adafruit.com/products/1469, viewed March 17, 2017.

[39] Arduino IDE Website, http://arduino.cc/en/main/software, viewed March 17, 2017.

[40] Adafruit CC3000 library Website, https://github.com/adafruit/Adafruit_CC3000_Library, viewed March 17, 2017.

[41] N. Mohamed, S. Lazarova-Molnar, and J. Al-Jaroodi, “Cloud of Things: Optimizing Smart City Services,” in proc. 7th Int’l Conference on Modeling, Simulation and Applied Optimization, IEEE, April 2017.