SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services

Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, Sanja Lazarova-Molnar, and Sara Mahmoud

***Abstract*—Smart cities are becoming a reality. Various aspects of modern cities are being automated and integrated with information and communication technologies (ICT) to achieve higher functionality, optimized resources utilization and management and improved quality of life for the residents. Smart cities rely heavily on utilizing various software, hardware and communication technologies to improve the operations in areas ike healthcare, transportation, energy, education, logistics and many others, while reducing costs and resources consumption. One of the promising technologies to support such efforts is the Cloud of Things (CoT). CoT provides a platform for linking the cyber parts of a 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 resources. Another useful technology is Fog Computing, which extends the traditional Cloud Computing paradigm to the edge of the network to enable localized and real time support for operating enhanced smart city services. However, proper integration and efficient utilization of CoT and Fog Computing is not an easy task. The paper discusses how the service-oriented middleware (SOM) approach can help resolve some of the challenges of developing and operating smart city services using CoT and Fog Computing. We propose a SOM called SmartCityWare for effective integration and utilization of CoT and Fog Computing. SmartCityWare abstracts services and components involved in smart city applications as services accessible through the service-oriented model. This enhances integration and allows for flexible inclusion and utilization of the various services needed in a smart city application. In addition, we discuss the implementation and experimental issues of SmartCityWare and demonstrate its use through examples of smart city applications.**

***Index Terms*— Smart City, Cloud of Things, Internet of Things, Cyber Physical Systems, Middleware, Service-Oriented Middleware, Cloud Computing, Fog Computing**

I. INTRODUCTION

SMART cities are the promising future of high quality living for the increasing population of cities in the world. Urban population increased from 746 million in 1950 to almost 4 billion in 2014 and the projections show further increases in these numbers reaching around 6 billion by 2050. Mega cities, accommodating 10 or more million people are increasing in numbers and large cities are also growing rapidly. To achieve high quality living and manage and operate these large cities, innovative solutions are needed, leading to the development of the smart city concept.

In a smart city, various aspects of living, operations, and management are automated and streamlined through effective and usually intelligent computing systems. The base unit in a smart city system is the sensors. Sensors of various types, capabilities and functionalities are deployed to monitor and record city parameters. These sensors need to be integrated with other devices and computing facilities to achieve their goals for monitoring and control of the smart city functions.

Various technologies and computational approaches provide the basic capabilities to integrate the sensors, actuators and other devices in a city's physical environment to create a smart city. Advances in Cyber-Physical Systems (CPS), Internet of Things (IoT), Cloud Computing (CC), Fog Computing and other software technologies have positively contributed to this goal. These new technologies offer an unprecedented opportunity to create a wide array of applications that optimize smart cities' services. These technologies are integrated into the Cloud of Things (CoT) [1]. In CoT, all objects of a smart city like the residents, vehicles, streets, buildings, hospitals, and energy and water plants are interconnected through the IoT, which is integrated with CC systems, running intelligent software to optimize the smart city's services. In addition, Fog Computing can offer extension features for CC systems to better support low latency requirements, location awareness, scalability, and mobility for these services [2].

Smart city applications can be developed to effectively and efficiently use the available and emerging technologies to continue enhancing the living quality of the residents, while optimizing the utilization of the city's resources and reducing the negative impact on the environment. One of these is Cloud Computing Systems, which provide large scale computational and data storage services to smart cities [3][4][5]. Another technology is Fog Computing Systems, which augment the functions of cloud services by providing services closer to the physical city environment. As a result, it can support the low latency, location awareness, mobility, streaming, and real-time requirements of the smart city applications [6][7]. Another very useful technology is the Wireless Sensor Networks (WSN), which are used to connect sensors for monitoring the different resources, components, residents and operations of a

Nader Mohamed is with Middleware Technologies Lab., Pittsburgh, Pennsylvania, USA (e-mail: nader@middleware-tech.net).

Jameela Al-Jaroodi is with Department of Engineering, Robert Morris University, Moon, Pennsylvania, USA (e-mail: aljaroodi@rmu.edu).

Imad Jawhar is with Midcomp Research Center, Saida, Lebanon (e-mail: imad@midcomp.net).

Sanja Lazarova-Molnar is with Center for Energy Informatics, University of Southern Denmark, Denmark (e-mail: slmo@mmmi.sdu.dk).

Sara Mahmoud is with College of Information Technology, UAE University, UAE (e-mail: 201370014@uaeu.ac.ae).

smart city [8][9]. The Internet of Things (IoT) is another technology being developed for various applications; however, it is very useful for integrating the physical objects of a smart city in a well-defined network [10][11]. Cyber Physical systems further extend the IoT concept to facilitate the interaction between the cyber world and the physical world in a smart city [12]. Other relevant technologies for a smart city include robotics, to provide different ground actions and physical controls [13]; Unmanned Aerial Vehicles (UAV), to enhance delivery of services, traffic monitoring, security and safety controls, and telecommunication services [14][15]; and Big Data Analytics (BDA), to provide smart decisions based on collected data [16][17].

Integrating technologies like WSN, IoT, CPS, robotics, and UAVs in addition to other technologies that will be available in the future with Cloud Computing will create the Cloud of Things (CoT). CoT can support the operations in a smart city, which can also be further enhanced by utilizing Fog Computing [18]. CoT and Fog computing will provide a powerful environment for supporting the operations of smart city applications. However, developing, implementing, maintaining, and operating these applications in an effective manner are major challenges. This paper introduces a service-oriented middleware approach to relax these challenges. We propose SmartCityWare, a SOM 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 and services of SmartCityWare are discussed in Section IV while SmartCityWare runtime environment is discussed in Section V. Section VI illustrates some examples for smart city applications using SmartCityWare. Some experimental evaluations are discussed in Section VII, Section VIII is an overview of some related work, and Section IX concludes the paper.

II. BACKGROUND

Creating and sustaining a smart city requires the integration of various technologies and the collaboration of many entities. City administration, city officials, emergency response teams, workers and residents all need to be involved in the process. In addition, the infrastructure, buildings, transportation systems, spaces and all physical aspects of the city are involved. To tie all of this together a sophisticated network of sensor devices, actuators, computing facilities and smart devices must be put in place. The general smart city concept involves monitoring, controlling, and managing the conditions of all of the city's infrastructures and physical components to optimize operations and use of resources, while providing high quality services to the citizens [19][20]. These include critical components like hospitals, power and water plants, communication networks, airports, seaports and transportation infrastructures. In addition, there are residential and commercial buildings, parks, recreational facilities, vehicles, and all types of electronic and mechanical devices used by people.

The major contributor to the emergence of smart cities is the development in sensor technologies. These include specialized sensors devised for specific purposes and also the use of smart devices to provide sensing and data collection features like smart phones capable of sensing the location, temperature and other aspects of an environment. However, sensing devices also require networks and computing facilities that allow for accurate data collection, aggregation and dissemination. This requires using Information and Communication Technologies (ICT) to facilitate and optimize the services provided in a smart city. 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 city components and 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. Having an IoT in place that includes sensing and actuating devices within a smart city can help provide specific enhancements. However, integrating this IoT with the cloud opens up a larger set of capabilities to facilitate large scale computation and decision making for the smart city. It will also allow for the integration of multiple IoTs and physical environments to create a larger view of the smart city's operations, leading to enhanced decisions and optimizations. The CoT helps connect, monitor, and apply enhancements in all aspects of a smart city through the IoT and Cloud services as shown in Figure 1. Cloud computing provides a flexible virtual execution system and on-demand services for a smart city. 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. 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 provide services to implement and operate optimization applications for smart city services.

Another emerging technology that could be of great benefit to smart city applications is Fog computing as it can enhance the CoT paradigm by providing small platforms located at the network edges in a smart city. These fog platforms can operate localized cloud-like services to support 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 devices or nearby edge devices. The Fog platform will allow executing services geographically close to the IoT applications. This offers several advantages for IoT applications including [2]:

- Providing low latency services, as fog devices are located closer to the actual IoT components and can react faster than the cloud.
- Offering location aware services based on the location of the IoT components in use and the connected fog nodes.
- Providing better scalability support for widely geographical distributed applications. This is enabled due

to the availability of multiple fog nodes within the different geographic locations, thus the need to centralize the tasks is minimized as the fog nodes can handle many requirements locally.

- Supporting better mobility and access control for different types of mobile devices as they travel around the city. As a result, these devices can have access to the required services through the nearest fog nodes.
- Offering better Quality of Services (QoS) support. Some services have strict QoS requirements and the fog nodes will be able to help support these requirements locally.
- Providing more efficient communication with other systems. Fog nodes are structured and designed like cloud nodes thus they can communicate with different system through the cloud or other fogs to achieve certain goals.

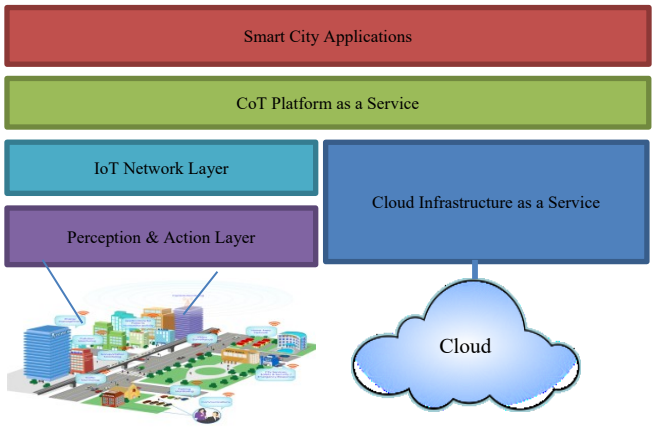

Figure 1. The CoT Model for Smart Cities.

These advantages help create solutions to many challenges that smart city applications face and enable the 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 a smart city is shown in Figure 2. In this architecture, the fogs will provide more localized real-time monitoring, control, and optimization for the smart city applications while the cloud will provide global monitoring, control, optimization, and future planning for these applications.

Various types of smart city applications can be designed and implemented with the support of COT and Fog computing. These include applications for intelligent transportation systems, smart energy systems, infrastructure and environment monitoring, and public safety applications. Table 1 lists some examples of smart city applications and how they can benefit from both the CoT and fog computing. More discussion on how CoT and fog computing can support these applications is available in [41]. These applications have specific requirements and may pose several challenges for their developers. Some of these challenges include:

- Support for real time operations and responses.
- The ability to seamlessly handle heterogeneous devices and components.
- The ability to accommodate for devices with limited resources and operational capabilities.
- The ability to support highly distributed systems spanning large geographic areas.
- Support for security and privacy measures.
- Support for reliability and fault tolerance.
- Support for device mobility.
- The ability to integrate and interoperate with other systems.

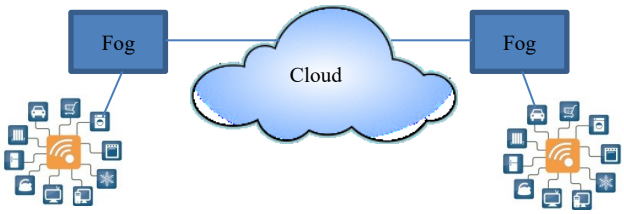

Figure 2. Integrating Fog Computing and CoT for Smart City applications.

The integration of CoT and Fog computing along with the right software architectures can leverage many of these challenges leading to more effective and efficient smart city applications.

III. SERVICE-ORIENTED MIDDLEWARE

Middleware technologies have become a necessary part of any distributed environment [21]. Middleware offers essential enabling features and functionalities for facilitating the integration of the distributed environment components and the operations of the whole distributed and heterogeneous applications. It simplifies the development and execution of distributed applications and hides their complexity. It also provides common services for recurring challenges in the distributed environment. Middleware also connects any set of components in a distributed environment to provide better 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. Better functionalities can be defined in terms of communication, integration, operations, reliability, availability, scalability, security, and other value-added functions.

Smart cities are complex and very large distributed systems that share with other distributed environments their heterogeneity, security, and reliability challenges. In addition, they also have their own unique challenges to provide and support high scalability, efficiency, safety, real-time responses, and smartness (intelligence) requirements. These are common challenges facing most smart city applications including smart grids, smart water networks, intelligent transportation systems, infrastructures monitoring and protection, and several others. Designing and building applications meeting all these challenges is extremely complex. As a result, it is almost impossible to develop and operate smart city applications without relying on advanced middleware technologies to simplify and facilitate the development and operations processes.

Table 1. Examples of smart city applications that can benefit from CoT and Fog computing.

Smart City Application	Sub-applications	Fog Roles	Cloud Roles
Intelligent transportation	• Route planning and congestion avoidance • Intelligent traffic light controls • Intelligent parking services • Accident avoidance • Self-driving buses/cars	Fogs in the form of Road Side Units (RSUs) or other computerized units provide low-cost relays among vehicles', roads' and parks' sensors, traffic lights, and the cloud. They provide fast response and control services.	Cloud collects, filters, and stores traffic information. It helps in coordinating city traffic and parking optimizations. It also helps in planning for enhancing traffic systems.
Smart energy	• Smart grid • Smart buildings • Renewable energy plants • Smart meters • Wind farms • Hydropower plants	Fogs provide local controls for energy systems, distribution units, and consumer locations. They also enable smooth integration of different energy systems.	Cloud collects, filters, and stores energy information. It supports decision making for utilizing smart grids and renewable energy features based on collected and analyzed data for consumers' needs and renewable energy productions.
Smart water	• Leakage detections • Water leakage reduction • Water quality monitoring • Smart water meters • Smart irrigation	Fogs provide better and faster local monitoring and controls for smart water networks. They also offer real-time monitoring for faults and leakages and support repair and maintenance operations.	Smart water networks information is collected, stored, and utilized by cloud services to enhance the water networks, production, and quality and to reduce water losses.
City structure health monitoring	Health monitoring for • Bridges • Large public buildings • Tunnels • Train and subway rails • Oil and gas pipelines	Fogs help reduce data traffic between the sensors monitoring the structures and their main control stations. In addition, they provide fast safety controls for some applications.	Cloud collects, filters, and stores structure health information. The cloud can help analyze collected data to enhance the maintenance processes and improve the health of the city structures.
Environmental monitoring	• Air quality monitoring • Noise monitoring • River monitoring • Coastal monitoring	Fogs help enhance environmental monitoring processes by providing smart environmental monitoring closer to the monitored areas.	Cloud provides processes to collectively analyze city environmental and health status.
Public safety and security	• Crowd control for large events (sports games, parades, and outdoor celebrations) • City crime watch and alerts • Large-scale emergency response services (e.g. floods, earthquakes, terrorist attacks, volcanoes, and wars)	Fogs help reduce the communication traffic between these places and the main security monitoring stations.	Cloud provides a powerful platform for analyzing the collected data about the current situation to help in providing possible actions for better controls and emergency relief.

A new and advanced approach in middleware technologies is the use of service-oriented middleware (SOM). This approach has been proven to simplify the implementation and operations of many applications in diverse industrial domains [22]. The approach was used to reduce the effort and cost of development, testing, and operations. Similarly, SOM can play an important role for developing, operating, and supporting smart city applications. Accordingly, we anticipate a successful migration of the SOM model to utilize the concept of IoT to support smart city applications and provide a generic middleware platform that will highly increase productivity and widen the range of applications that can be designed and built for smart cities.

SOM extends the capabilities of middleware and provides high flexibility for adding new and advanced functions to smart city applications. SOM logically views smart city cyber and physical components as providers of services for smart city applications. With SOM, all hardware devices such as sensors, actuators, storage devices, communication devices, and processors can be viewed and utilized as services. The implementation of this SOM is usually achieved through web services standards, wrapper technologies to map different devices' interfaces to web service interfaces, and middleware services to enable the integration among both services clients and services providers [22]. Furthermore, it can integrate these services with other services provided by Cloud Computing and Fog Computing Thus, allowing application developers to view everything as a single large system that provides basic and advanced services to smart city applications. Advanced services, such data aggregation, adaptation, security, self-organization, reliability, and management, can be designed, implemented, and integrated in a SOM framework through a more flexible and easy to use development and execution environment. SOM for smart cities is necessary to support several, otherwise hard to incorporate, functionalities in the service-oriented computing (SOC) model. These functionalities include the functional and non-functional requirements that different services may need. For any service-oriented application, there are common functionalities such as service registry, discovery, communication, reliability and security that are needed by any of these applications. These can be easily

generalized and made available via the SOM platform to be used by the applications' developers to easily implement smart city applications. Generally, SOM for smart cities should support several requirements, some of which (e.g. the first three in the list) are common for any SOC application, while the rest are enforced by the characteristics of the smart city environment and the challenges of implementing and operating applications on CoT and Fog Computing. These requirements include:

1. Runtime support for services deployment and execution: as all devices in a 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 provide mechanisms to link, load, deploy, and execute these services as needed.
2. Support for different communication methods among service consumers, services, service registries and brokers that enable reliable and efficient local and remote services utilization.
3. Support for consumers to discover and use registered services: SOM should enable client applications to discover and use registered services when needed. SOM should also support run-time integration between applications and registered services.
4. Support for service transparency to client applications: SOM should allow client applications to transparently use available services without exposing services implementation details or, in some situations, their detailed components locations.
5. Suitable abstractions to hide the heterogeneity of the underlying environments: all heterogeneity details of a smart city's physical devices and networks should be hidden from the applications. SOM should provide high-level interfaces to utilize cyber and physical smart city resources without requiring application developers to deal directly with the heterogeneity.
6. Support for configurable services: SOM should provide mechanisms for client applications to configure smart city services to meet specific applications' requirements such as QoS, security or reliability. Configuring smart city services usually requires dealing with details and parameters of some hardware, network components, cloud and fog configurations. Doing this for configuring QoS requirements, for example, will be a complex task for regular users. However, SOM can provide mechanisms to be used easily by client applications to configure smart city resources for their specific requirements.
7. Support for self-organization mechanisms: this includes self-x properties such self-management, self-healing, self-configuration, auto-discovery, self-adaptation, and self-optimization of service providers. Smart cities are dynamic distributed environments where resources can be added, changed, or removed anytime. In addition, some resources may be mobile like the UAVs. The availability of services for such devices is also dynamic. Therefore, SOM should support self-management, auto-discovery, self-optimization and auto-change mechanisms for efficient utilization of all available services. For example, when a sensor provides a service for sensing a certain attribute, the SOM should discover that service and allow the client applications to use it when they need it. When that senor fails, SOM should automatically switch the application to a similar service currently available in the area. Furthermore, the SOM should be able to notify the application if nothing matching their needs is currently available. This helps solve the scalability, heterogeneity, and network organization challenges.
8. Support for interoperability with a variety of devices: This requirement helps solve the heterogeneity challenge in smart cities by supporting different interoperability mechanisms to match available devices. Some smart city applications require a variety of devices to be operated. SOM for smart cities can be designed to be interoperable with different devices such as devices with different types of sensors, RFID, vehicles, UAVs, cloud and fog services to enable easy application development and operations.
9. Efficient handling of large volumes of data and high communication loads: 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. SOM should also be capable of reallocating data as necessary while maintaining proper access to the applications using it.
10. Support for secure communication and execution: as most smart city applications involve sensitive and critical information, secure communication and execution becomes a very important aspect in SOM for smart cities. SOM should provide mechanisms to secure the utilization and operations of both CoT and Fog Computing services. All communications and execution for supporting these services should be also secured.
11. Support for QoS requirements: some smart city applications have specific QoS requirements. Mechanisms are needed to configure and satisfy these requirements in the CoT and Fog computing systems utilized by these applications. An example of QoS requirements in a smart city can be observing and reporting a current traffic situation in a certain road intersect within a given time frame and within a specific error margin. In some situations, the QoS requirements can come from multiple applications such as safety and collaborative sensing. SOM for smart cities should provide uniform interfaces to configure the QoS requirements for these applications and ensure achieving these requirements.
12. Support for integration with other systems: smart city applications usually do not operate in isolation thus the SOM should enable the integration of smart city applications with other systems such as enterprise or web

systems. For example, some web applications rely on smart city applications for their current information such as information on traffic conditions. In this case, SOM should enable that integration such that these applications can fulfill their goals.

In addition to these requirements, there are other advanced and specialized requirements that are needed for some smart city applications. Examples are support for context awareness and location-based services. Generally, SOM is designed 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 applications requirements and do not provide complete solutions for their challenges.

IV. SMARTCITYWARE

SmartCityWare is a service-oriented middleware platform designed specifically to utilize CoT and Fog Computing to support developing and operating smart city applications. The main purpose of SmartCityWare is to provide a virtual environment to be used to develop and deploy smart city applications. SmartCityWare consists of a set of services and a multi-agent runtime environment. In this section, we discuss the services of SmartCityWare while in Section V we will discuss the multi-agent runtime environment of SmartCityWare.

In SmartCityWare, all functions are viewed as a set of services that can be used to build and support the execution of different smart city applications. These services are classified into core services and environmental services. Core services are those developed specifically for the core operations of SmartCityWare, such as the broker, security, service invocation, and location aware services. These services provide overall control for the whole system. Environmental services provide access to services provided by one or more cloud service provider, services provided by 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 control, processing, storage, communication, streaming, configuration, monitoring, measurement, and management services. The IoT devices services provide interfaces to utilize device functionalities like sensing, action, or other services. The environmental services can either provide 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 reliability and security features. While some IoT devices can execute on-board code to implement and provide some services, 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 devices 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 a 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.

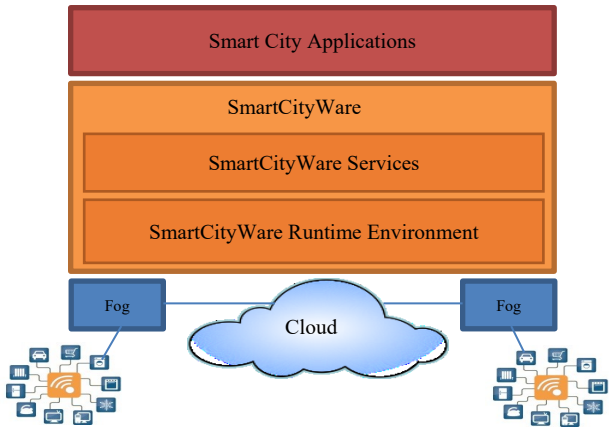

Figure 3. SmartCityWare Supporting Smart City Applications.

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 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

When an application discovers a service it needs, it will require that service to be invoked with the proper interfaces. The invocation services, local and remote, are used by consumers with SmartCityWare support to start executing the required services. Local invocation services are limited within a single system where services needed exist in the same location. 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, establishing communication connections between service consumers and producers, data marshaling and demarshalling, delivering requests and responses, and executing services. All these steps in the invocation process are achieved based on web services standards.

C. Location Based Services

SmartCityWare can provide location-based services. Unlike regular service brokers that are used over the Internet, SmartCityWare service brokers for fogs 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 these services. 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 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 various 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 required 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.

E. Other Core Services

SmartCityWare is comprised of a collection of services, thus it becomes possible to introduce other specialized services that implement common solutions for smart city applications to the SmartCityWare platform. These specialized services can cover additional requirements from those discussed in Section III. One example is adding reliability and fault tolerance features. For example, Alho and Mattila [42] proposed, developed, and evaluated a service-oriented approach to address reliability and fault tolerance in cyber-physical systems. This approach can be employed as a collection of services and added to SmartCityWare. Furthermore, additional core features and services can be added to further customize SmartCityWare for specific smart city applications like the ones listed in Section II. Some examples can be services to support high mobility for vehicular applications or distributed resource management services for large-scale smart applications such as smart grids. In addition, SmartCityWare services can use advanced functions provided by the network used for connecting the smart city's distributed components. One example is to utilize Software Defined Networking (SDN) features which provide abstractions for the underlying networks and systems to programmatically control and configure the networks to achieve the required network performance [43]. This requires integrating SmartCityWare with SDN controllers such as OpenDaylight [44]. The details of this option are left for future investigation. Generally, SmartCityWare offers a middleware infrastructure where services and resources provided by Cloud Computing and Fog Computing can be added to help in solving the diverse issues of smart city applications.

V. SmartCityWare Multi-Agent Runtime Environment

The main function of SmartCityWare multi-agent runtime environment is to manage the Smart city's IoT, fogs and cloud distributed services. It provides a distributed run-time environment to securely execute these services as shown in Figure 3 earlier. This runtime environment is based on our earlier implementation of a multi-agent middleware infrastructure environment developed for heterogeneous systems [34][35]. We customized that middleware to support the SOC model of SmartCityWare. This SmartCityWare multi-agent runtime environment is a pure Java infrastructure based on a distributed memory model. This makes it portable, secure, and capable of handling different heterogonous environments that consist of heterogeneous fogs, clouds, and IoT devices. The agents are deployed in the participating fogs and cloud machines to support SmartCityWare services runtime

requirements. The infrastructure has various components that collectively provide the runtime support for both the core and environmental services of SmartCityWare.

A. Agents

Software agents' technology has been used in many systems to enhance the performance and quality of their services [47]. Our middleware infrastructure utilizes software agents to provide flexible and expandable middleware services for high-performance distributed service-oriented environments. The main functions of the agents are to deploy, schedule, and support the execution of the service codes in different fogs, in addition to managing, controlling, monitoring, and scheduling the available resources on a single fog or on a set of related distributed heterogeneous fogs. When a smart city application is executed, an agent performs the following tasks:

1. Examine available fog resources and schedule the services for execution.
2. Convert scheduled user services into threads, then remotely upload and execute them directly from the main memories on the remote fog machines.
3. Monitor and control resources and provide monitoring and control functions to the user.

For high throughput, the agents are designed to be multithreaded, where each thread serves a client's service request. Once user threads are deployed, they directly communicate with one another to perform their distributed tasks, thus freeing the agents and reducing the overhead on the user programs. Agents' communication mechanisms are employed using sockets and each agent consists of several components that implement different functions:

1. The Request Manager handles user job requests such as deploying services' classes, starting/stopping a service, and checking agents/services/threads status. Requests come from client services or from agents of other fogs or clouds.
2. The Resource Manager provides methods to manage, schedule, and maintain the resources of the machine where the agent resides. It keeps records of executing threads, machine and communication resources' utilization, and performance information. In addition, it is responsible for reclaiming system resources after each service's completion or termination.
3. The Security Manager provides security measures for the system (see next subsection for details).
4. The Service Class Loader remotely loads user service classes on the remote machines preparing for execution.
5. The Scheduler selects the fogs/cloud machines to execute a user service based on the its requirements.

B. Multiuser and Security Issues

The multi-agent middleware infrastructure allows multiple users to execute multiple services simultaneously. To properly manage these services, each service has multiple levels of identification, starting with a unique service ID assigned by the system. The user ID and the program name further distinguish different services. Within each service, thread IDs are used to identify the remote threads of the service. Executing user threads on remote fog machines exposes these fogs to many "alien" threats, raising security and integrity worries. Therefore, these machines must be protected to ensure safe execution. Java's default security manager offers some security mechanisms of protection by checking executions against certain defined security policies before execution. However, the security manager in Java has some limitations, thus many features were updated for our middleware infrastructure. More specifically, two modes of operation are used to offer a secure and reliable environment:

1. The Agent Mode, in which no limitations are enforced. A thread running in this mode has full access and control of all the operations, services, and resources in the corresponding fog.
2. The User Mode, in which limitations are enforced to limit users' access to the fog operations, services, and resources. Some operations, such as removing files, initiating a process, using system calls, and changing system properties are inactive in this mode.

With the security modes in place, the user services have full access to operations and resources on their local machines (where the user job was initiated), but limited and controlled access to all remote machines' resources (since they are running in user mode). Nevertheless, this policy can be adapted to offer other levels of access control, when needed, on the available fog machines. For example, a user can be given to access to certain fogs while he/she is disabled on other fogs.

VI. APPLICATION EXAMPLES

SmartCityWare can be utilized for building and operating smart city applications. Here we present two potential and relevant smart city applications for the SmartCityWare: an intelligent traffic light control system; and smart buildings collaborative data analytics.

A. Intelligent Traffic Light Control System

Intelligent traffic light controls can be facilitated by SmartCityWare. This implies that the traffic light controls will feature monitoring devices at numerous locations to accurately capture and model traffic patterns and utilize this information to adjust traffic lights to optimize flow. This type of application can, furthermore, utilize global positioning, vehicle-to-vehicle and vehicle-to-infrastructure communication systems to enhance the granularity of data collection. For example, individual vehicles can be observed from various perspectives (road sensors, cameras, on vehicle sensors, neighboring vehicles, etc.) to create a more accurate models. This enhanced granularity of data can be utilized to achieve global urban traffic control optimizations. We expect this application to yield shorter traffic delays, higher throughput, shorter vehicles travel times, higher vehicles average velocity, and improved prioritization of emergency vehicles' movements in urban areas. Typical algorithms that can be utilized to enhance urban traffic control optimization are learning and adaptation algorithms. Projects that exemplify this are discussed in [23], [24], and [25].

SmartCityWare can support the development and operations of intelligent traffic light control systems. All involved devices within the reaching area of the traffic light, such as the vehicles, road sensors, and communication devices, are considered as services that observe the status of the traffic in that area. These processes and elements are illustrated in high-level in Figure 4. In each traffic light, a fog can be used to perform various services, as well as provide local optimization. The information gathered through the relevant devices will be utilized by the fog to make informed optimizing decisions about specific traffic lights. A collection of services on a cloud can also monitor the global traffic status within the smart city by observing the status of the fogs operations and local conditions. This information will be utilized by the cloud services to provide global optimizations for decreasing traffic delays, enhancing vehicles' flow, and prioritizing emergency traffic on a larger scale, covering urban locations more thoroughly. The status of the traffic in a smart city can be gathered through these services in regular periods. This status can then be used to analyze, assess and enhance the traffic, as well as configure the fog services correspondingly. Furthermore, cloud services can facilitate synchronization mechanisms among multiple fogs to organize multiple traffic lights available within a short distance of each other or positioned at a significant street in a smart city. With the change in traffic conditions, all fogs will fine-tune their operations using the locally collected information along with the global view delivered by the cloud services.

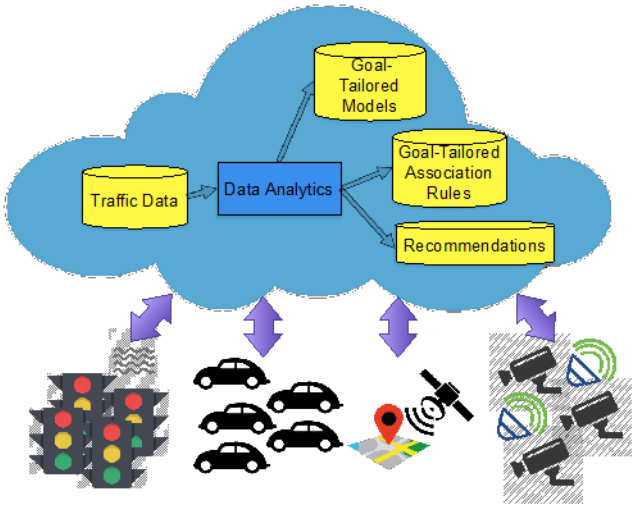

Figure 4. SmartCityWare application for intelligent traffic light control systems.

B. *Smart Buildings Data Analytics*

Another potential application of SmartCityWare is enabling collaborative data analytics for optimizing smart buildings energy performance. As it has been stated repeatedly, buildings in urban areas account for around 40% of the overall energy consumption [45]. Thus, their energy performance optimization represents a significant opportunity to reduce the overall energy consumption. A large part of commercial buildings is run by Building Management Systems (BMS), and the utilization of Information and Communication Technologies (ICT) enables for utilization of sophisticated algorithms and methods to enhance buildings' performance. Furthermore, smart buildings encompass many sensing and monitoring devices, as well as various types of actuators. As it has been demonstrated in [46], that collaborative data analytics of smart buildings can positively impact the accuracy and quality of results, decisions and actions, as compared to isolated buildings data analytics, in which cases it would need long times of data collection before their data becomes useful. SmartCityWare can be utilized to enhance the collaborative data analytics among large numbers of smart buildings.

The availability and quality of security services would be highly relevant for participating buildings' owners, as we are talking about commonly confidential data. Therefore, SmartCityWare will be responsible for providing adequate mechanisms for anonymizing and protecting data. It will also provide mechanisms to transfer data to the cloud, as well as machine learning algorithms for development of models. The cloud will be providing these learning algorithms and it will perform global optimization, also taking into consideration information from the smart grid power generation. Furthermore, SmartCityWare will offer mechanisms for transferring decisions and recommendations to smart buildings, which can then act upon them to enhance their performances. These processes and main participants are illustrated in Figure 5. Here, as shown in the figure, a fog can be used for each building to execute many recommendations and to provide local optimization. SmartCityWare will enable development of much more accurate models, as yielded by the collaborative data analytics. These models can serve different purposes, such as: timely and accurate fault detection and diagnosis (cloud services), generation of (nearly) optimal preventive maintenance schedules (fog services), optimization of matching of demand and response (cloud services), generation of optimal set points for each building (fog services), etc.

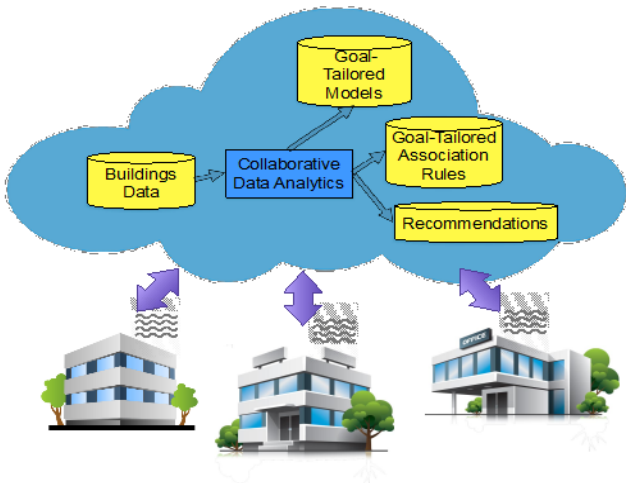

Figure 5. SmartCityWare application for collaborative data analytics for smart buildings.

Using SmartCityWare to enable utilizing cloud computing for collaborative data analytics for smart building can provide faster mechanisms to discover faults. As shown in [46], energy

savings can be achieved with connecting two or more smart buildings to the cloud. The savings can be increased by connecting more buildings as faults can be discovered faster and energy losses will be discovered and avoided earlier. Although, some overhead and extra cost will be involved in this connection, energy savings from experiences collected from multiple buildings can significantly exceed the extra costs specially with multiple buildings. More analysis and information on this approach can be found in [46].

VII. Implementation and Experiments

A SmartCityWare prototype was implemented including the multi-agent runtime environment discussed in Section V. In addition, most of the core services of SmartCityWare mentioned in Section IV were implemented. These include the 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, with new information from the local brokers, 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 lookup 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. Furthermore, some LEDs and a buzz were installed to represent actuators. In addition, we installed an Adafruit CC3000 Wi-Fi board [38] to connect 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 implemented 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 used three computers; one represents the cloud and two represent two fogs. In addition, we used WAN emulators among the machines to introduce the effects of using long distances and/or the Internet to connect them. Experiments were conducted with different configurations:

- LSC: a local IoT service call within the corresponding fog.
- RSCCF: a remote IoT service call from the cloud.
- RSCFF: a remote IoT service call to another fog where both fogs are connected using a WAN and not involving the cloud.
- RSCFCF: a remote IoT service call to another fog through the cloud.

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 6. These recoded times for these calls do not include the service lookup times. The response time for a service call from fog to another fog and from cloud to fog is similar as the service call that is directly done between the client fog and the server fog without involving the cloud.

The average service lookup time for local services (Local), remote services between a fog and the cloud or between a fog and another fog where they are connected by a WAN (Remote-direct), and remote services between a fog and another fog through the cloud (Remote-cloud) as there is no direct network between both fogs are shown in Figure 7. 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 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.

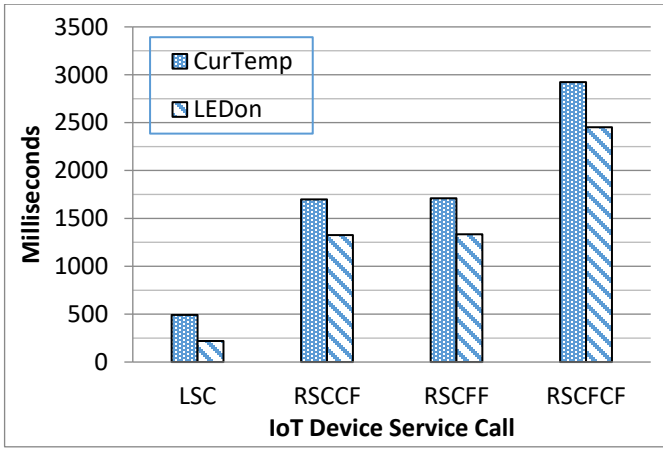

Figure 6. IoT device service calls response times.

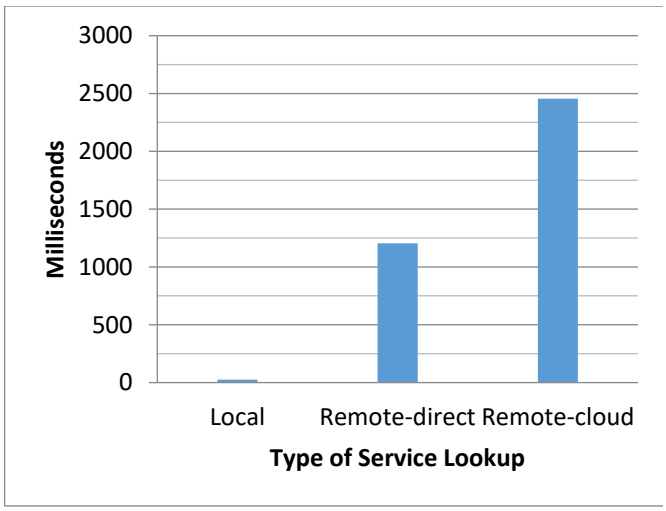

Figure 7. Service lookup times.

VIII. RELATED WORK

Several 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 SOM 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 a common platform for smart cities were discussed in [32][33]. Our work in this paper utilizes the flexibility and extensibility of the 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.

IX. CONCLUSION

Smart city applications provide numerous enhancements to the smart city features and capabilities leading to enhanced operations, optimized resources utilization and ultimately a better quality of life for the residents. These applications often require using multiple ICT components and the integration of various systems and services. IoT, Fog and Cloud computing can be integrated to support these applications, yet this imposes multiple challenges on the development and operations of these applications. Middleware support is essential for such applications to meet the challenges imposed such as heterogeneity, mobility, and real-time support. In this paper, we outlined the functions and features needed in a middleware infrastructure to support smart city applications. Based on these functions, a service-oriented middleware that integrates and utilizes the cloud of things (CoT) and fog computing and provides a set of services to support smart city applications was proposed. This middleware is named SmartCityWare, where 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 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 emerging technologies, other than Cloud of Things and Fog Computing, for supporting smart city applications in the future. In addition, we discussed the design and architecture of SmartCityWare and its runtime environment. We also offered some implementation details and experimental results showing the validity of the approach. In the future, we plan to enhance the implementation of SmartCityWare, include more features and services common to many smart city applications and demonstrate its features through actual implementations of specific smart city applications.

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 Proceedings 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, A. and Vinci, "Smart Agents and Fog Computing for Smart City Applications," In International Conference on Smart Cities, pp. 137-146, Springer International Publishing, 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 (SENSORCOMM), 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, B., "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 4-6, 2017.

[42] P. Alho and J. Mattila, "Service-oriented approach to fault tolerance in CPSs," Journal of Systems and Software, 105, pp.1-17, 2015.

[43] D. Kreutz, et al. "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, 103(1), pp.14-76, 2015.

[44] J. Medved, et al. "Opendaylight: Towards a model-driven sdn controller architecture," IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014.

[45] C. A. Balaras, A. G. Gaglia, E. Georgopoulou, S. Mirasgedis, Y. Sarafidis, and D. P. Lalas, "European residential buildings and empirical assessment of the Hellenic building stock, energy consumption, emissions and potential energy savings," Building and Environment, vol. 42, pp. 1298-1314, 2007.

[46] S. Lazarova-Molnar and N. Mohamed, "Towards Collaborative Data Analytics for Smart Buildings," in International Conference on Information Science and Applications, pp. 459-466, 2017.

[47] F.D. Ahmed, M.A. Majid, M. Sharifuddin, and A.N. Jaber, A.N., "Software Agent and Cloud Computing: A Brief Review," International Journal of Software Engineering and Computer Systems, 2(1), 2016.