IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 31, 2018, accepted November 28, 2018, date of publication December 4, 2018,
date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2884741

Simplified Agent-Based Resource Sharing Approach for WSN-WSN Interaction in IoT/CPS Projects

GÜNGÖR YILDIRIM AND YETKİN TATAR
Department of Computer Engineering, Faculty of Engineering, Fırat University, 23100 Elazig, Turkey
Corresponding author: Güngör Yıldırım (gyildirim@dsi.gov.tr)

This work was supported by FUBAP of F.U, "Mobile Communication Technologies and Wireless Sensor Networks Systems," under Project MF1420.

ABSTRACT This paper focuses on the problem of interoperability and resource sharing in wireless sensor networks (WSNs) running under the Internet of Things (IoT) and cyber-physical systems (CPSs). Considering the scale of IoT/CPS projects, conventional WSN virtualization techniques remain incapable because of the hardware/software constraints and heterogeneity. To this end, in this paper, an agent-based server system approach, which improves the resource sharing between heterogeneous WSNs in IoT/CPS providers, is proposed. In line with this approach, a software agent framework is introduced. With the help of the framework, called Firat Virtual WSN framework (FVWSN), the clients can move the commands/queries and data fusion/aggregation algorithms, which use on their local networks, to the provider side and run them remotely or automatically. This process is carried out by logical agent entities, called virtual nodes, which are created with the help of FVWSN. In this way, since the client evaluation mechanism is performed at a closer point to the shared resources, a shorter response time can be achieved in time-critical applications. The most important features that differentiate the developed agent framework from other agent-based technologies are that it is semi-autonomous and uses a specific resource selection/allocation algorithm. With the improvement that FVWSN provides for IoT/CPS WSN providers, it is possible to achieve a shorter response time and allow more client applications to share the same limited WSN resources. In this paper, first, the analysis and the necessity of the proposed system are discussed. Then, the system is simulated in the OPNET Modeler platform to make comparisons with well-known conventional WSN resource sharing mechanisms. Finally, the physical comparison tests of the system are carried out on an OpenStack-based cloud system, and the success of the system is shown.

INDEX TERMS WSN, IoT, resource sharing, software agent.

I. INTRODUCTION

With the emergence of the Internet of Things (IOT) and Cyber Physical System (CPS) concepts, WSNs have begun to become more interactive and shareable systems. Resource sharing between heterogeneous WSNs provides significant advantages in terms of installation cost and time, especially for large IoT / CPS systems such as smart city and smart energy. The resource sharing in WSNs is usually carried out in two different ways. Those are WSN virtualization and middleware based server systems (MBSS) usage. The WSN virtualization techniques are classified in two titles; Node-based Virtualization (NoBV) and Network-based Virtualization (NeBV). NoBV is based on the principle in which several applications are executed on a local WSN node in sequential, event-driven or thread-based form [1], [2]. As expected, because of hardware/energy constraints and heterogeneity, NoBV is not preferred in the IoT/CPS scenarios that contain a lot of client applications. However, in time-critical applications, the shortest response time are achieved by the NoBV techniques. NeBV is performed by using specific protocols between different networks. Since these techniques, which also rely on node technology like NoBV, require special protocol designs, they have not been used effectively by all WSN technologies. However, NeBV techniques are successful in concealing the heterogeneity.

In general, considering the heterogeneity problem and today's some WSN technologies that do not run an embedded operating system, NoBV and NeBV may be inadequate in resource sharing in IoT/CPS projects [3]. Because of these factors, MBSSs are more efficient in resource sharing on the IOT / CPS scale. Transparency and synchronization in MBSSs can be successfully achieved by middleware, cloud, and remote procedure calls (RPC) technologies [4]–[6]. The MBSS based projects usually provide the clients with generic services, such as download/upload, filtering and give access permission to the server-side defined procedures. In the case that there are WSNs containing actuators, some MBSSs can offer specific control services [7]. When this is the case, the solutions provided are technology or brand-specific, and these solutions may not be insufficient in the dynamic resource movements in which different heterogeneous resources are added to and removed from the system. In addition, since there is an intermediary system between the clients and the resources shared, the MBSS based systems have a longer response time in time-critical applications. In particular, in the case that applications containing data fusion/aggregation (DF/DA) operations that perform actuator control exist, the response time can be quite important. In IoT/CPS projects, each client-WSN system has usually its own DA/DF operations, which are performed in their local system. Naturally, this makes the concept of response time the first performance factor in the time-critical scenarios. With the help of the arguments developed, this study proposes an improvement for MBSS-based WSN resource sharing. Thanks to the improvement, the heterogeneous WSNs running under an IoT/CPS project can behave as a single system. The improvement considers two performance criteria. The first is to achieve a faster response time in time-critical IoT/CPS projects that contain actuator control mechanisms. The second is to provide an interactive resource sharing for more client WSN applications compared to conventional WSN resource sharing techniques. The basic logic of achieving this is to bring the client evaluation entities a closer location to the resources shared. Thus, since the evaluation process is done at a closer location, the response time that is needed to control actuators or to notify other resources shared will be shorter. The evaluation entities are the commands, queries and DA/DF operations, which the client-WSNs use in their local systems. Considering the number of client applications and the hardware limitations, this location cannot be the target WSN nodes that have the shared resources. In this case, the closer location is the MBSS system. The main problem is how heterogeneous client-WSN evaluation entities developed by different software technologies can be migrated to the MBSS and executed there remotely or automatically. The proposed solution is to bring the client evaluation entities and the MBSS services under the same software framework roof. To this end, a software agent framework, called Firat Virtual WSN framework (FVWSN), has been developed. With the help of FVWSN developed in JAVA, the client-defined commands/queries (C/Q) and DA/DF operations can be executed in the MBSS that has more advanced hardware and software features. This naturally brings a second advantage that more client WSN applications can share the same resources. The proposed system is an improved multilayer IoT/CPS-MBSS that uses a software agent framework that enables client-defined codes to be executed on the provider side. The most active components of the proposed model are Virtual Nodes-VNds. The VNds, defined in the second layer and created with the help of FVWSN, are logical semi-autonomous client agents in the provider system. The coding of these agents is performed on the client side and then they are uploaded to the MBSS server. VNds can operate automatically, or be controlled remotely through the client-defined commands/queries which are carried by M2M-RPC (Remote Procedure Call) protocols. In this way, a physical node in the client WSN can communicate with the VNd on MBSS with its own commands/queries, which this provides a significant flexible interaction for clients. The most important features that distinguish the developed framework from other agent-based technologies such as JADE [8] are semi-autonomous, and use a specific resource selection/allocation algorithm. The main purpose of the study is to provide an improvement towards the conventional MBSS through the advantages of today's software agent concept. Therefore, the study will focus on the proposed improvement.

Before going into details, in order to explain the advantages of the proposed system, related works, the background and the analytic details will be presented in Section II. Section III will introduce the multilayer architecture, FVWSN framework and VNd execution. Applicability of the proposed system is shown in a simulation environment in Section IV. The physical environment test results of the system will be given in Section V. Finally, the future studies and the conclusion will be presented in Section VI and VII.

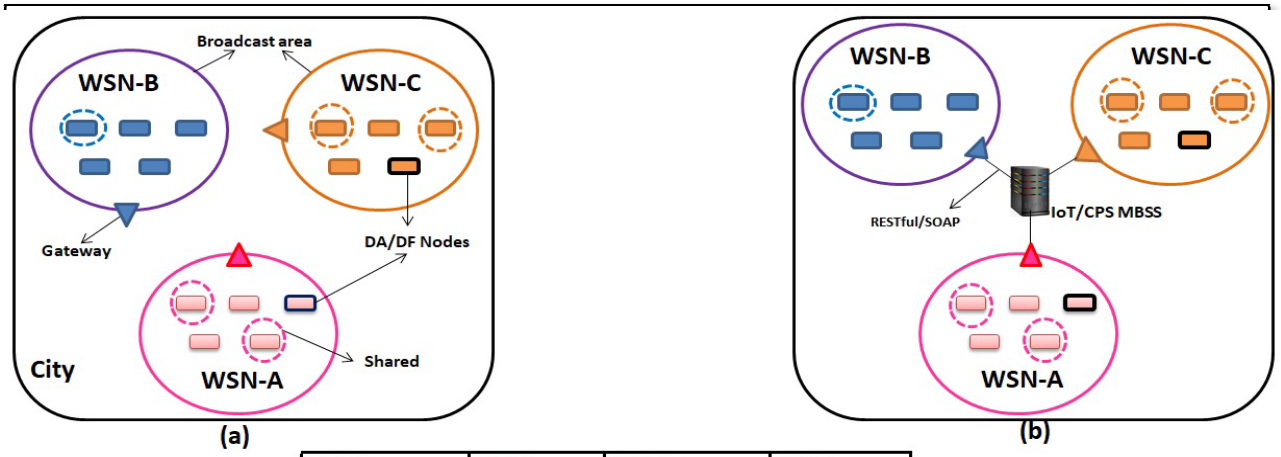
II. MOTIVATION

In order to make the proposed improvement more understandable, related works in the literature and background information will be presented. Also the contributions of the study will be listed in this section.

A. RELATED WORKS

The NoBV aims to run multiple applications on a single node. Depending on the node features, application tasks can be run as sequential, event-driven or multi-tasking. Also, in NoBV, multiple applications can be executed by using small virtual machines. In other words, the node virtualization completely depends on the specifications of the WSN nodes and the embedded operating system used.

The embedded operation systems (EOSs) such as TinyOS [36], Contiki [37], RIOT [38], LiteOS [39] are quite capable in this regard. In the literature, there are many NoBV studies carried out by using the EOSs. Agent-based Agilla framework [21] and Melete [22] are some of them. Although NoBV has the shortest response time, the method is not too suitable for the IoT systems serving many clients because of



Feature	WSN-A	WSN-B	WSN-C
EOS	Yes (TinyOS)	No	Yes (Contiki)
IP support	Yes	No	Yes
Node Technology	TelosB	XBee Standalone	NXP-JN Series
Application Layer	T-SQL	ZigBee Profile	User-defined
Location	different	different	different

FIGURE 1. An example scenario for heterogeneous WSN integration in an MBSS based IoT/CPS project.

the hardware restrictions. On the other hand, the principle in the NeBV is to construct virtual networks with the selected nodes. These types of virtual networks may be created by the nodes which are in a single WSN or different WSNs. It is possible to develop overlay and cluster structures through the NeBV. Vitro [23], Meno [24], IDRA [25] offer some improvements towards NeBV. Since NeBV techniques usually require special protocol designs, they have not been used effectively by all WSN technologies. On the other hand, IP-based WSN technologies such as 6LowPAN [26] have recently attracted the attention of researchers. However, these technologies cannot be used on WSN technologies that do not contain an IP layer like ZigBee version 2007. The other solution for WSN resource sharing is to use intermediate MBSSs. MBSSs allow clients/client-WSNs to share heterogeneous shared resources with different Internet or M2M technologies, such as cloud, pub-sub, and RPC technologies. These types of systems can be seen in both academic and industrial areas. Among these, Digi Device Cloud [7], Sentillo [4], Libelium [27], IoTSense [6] and Sensor Rush [28] are well-known examples. While some serve as sensor cloud systems only, others offer the WSN services to clients as sub-services. MBSS based systems have three basic delay times. Those are the time taken to transfer data from the shared source to the MBSS, the time spent in the MBSS, and the time spent to deliver the relevant data to the client. In the case of a customer request for a specific shared resource, the total delay time will be longer. Therefore, the MBSS based WSN resource sharing solutions are slower compared to NoBV and NeBV.

Agent-based solutions in the WSNs are generally used to solve the problems arising from the WSN's basic characteristics. These solutions are usually for data aggregation [32], [33], best route estimation [31], programming paradigm [30], and reconfiguration problems [34]. The most prominent resource-sharing agent-based solutions in the literature are Agilla and Sensorware [35]. On the other hand, agent-based solutions in WSNs are based on node technology, and the general constraints in NoBV are available in these solutions.

In this study, an improvement towards the MBSSs is proposed. The improvement is agent-based and centralized. The improvement also enables to execute the evaluation entities used in WSNs on the server systems and managing shared resources. To the best of our knowledge, such an agent-based resource sharing method has not been previously proposed in the literature.

B. BACKGROUND AND THE CONTRIBUTIONS OF THE STUDY

In order to make the proposed system more understandable, a scenario illustrated in Fig. I is discussed. In Fig.1a, WSN-A monitors the toxic gas levels in a city and wants to expand its broadcast area. In order to relieve from the new installation costs, it prefers resource sharing with other WSNs that can operate in the overlapped scheduled durations [29]. It is assumed that all of the sub-systems can run in both periodic and query/answer modes. Also, the WSNs want to use their local evaluation entities (command/queries, DA/DF algorithms, etc.) for the all shared resources including actuator resources. In this case, three solutions are proposed. The first one is NoBV.

However, as seen in their features in Fig. 1, the WSNs are heterogeneous. Therefore, the command/queries and the DA/DF algorithms in a WSN cannot be run directly in other systems. Furthermore, the node constraints may not allow uploading a lot of different client applications.

TABLE 1. Some general notations used in the study.

Notation	Remark	Notation	Remark
W_k	"k"*th* sub-WSN under the IoT/CPS system	$\dot{D}_{tot}^{MBSS}$ and $\dot{D}_{tot}^{MBSS^{prp}}$	Total delay in conventional and the proposed model
$N_i^{W_k}$	"i"*th* node in the sub-WSN "k"	φ	The number of a link or route usage
r	Shared resource tuple	OT(.)	Execution duration of a function operation
P_i^{Tr} and P_i^{Rc}	Transmit and receiving power of node "i"	T_{se}	Service time out for an agent VNd
$\grave{E}$	Residual energy level of a node	T_{thr}	Threshold time out for a sub-WSN-MBSS interaction
$\acute{P}$	Processing rate of a node	T_{ro}	Optimized threshold time out
$\acute{M}$	Memory size of a node	C'_{sr}	Reading limitation for an agent VNd
$\dot{D}_{WM}^{k}$	Delay between client "k" and the MBSS	Rs_{max}	Resource limitation for an agent VNd
$\dot{D}_{MW}^{l}$	Delay between the MBSS and sub-WSN	$B_{vt}{}^{queue}$	Waiting time of a demand task in the queue
$\dot{D}_{get}^{l}$ and $\dot{D}_{set}^{l}$	Delay in sub-WSN "l" for sensor reading and actuator setting	$B_{vt}{}^{s}$	Average waiting time of a demand task

The second solution is NeBV. This solution requires a specific protocol design and is not possible most of the time. Since WSN-B has no IP support, the internet based solutions cannot be used directly between the nodes in the WSNs. Instead, the WSNs may prefer to communicate with each other via their gateways that can run RPC technologies. In this case, the complexity of the infrastructure will increase as the number of WSNs increases. Furthermore, since the evaluation entities of the WSNs cannot still be used directly, the evaluation processes will continue to be performed in the local systems.

The third solution is to use intermediate MBSSs. It is clear that this solution will be more practical and useful in the big scale IoT / CPS projects. In this solution, all communication and procedure calling processes will be performed over the MBSS. This centralized solution also provides a good transparency and easy to use system model. However, since there is an intermediary structure between the clients and the shared resources, the delay will be longer. In addition, since all evaluations are still performed in the local WSNs, some disadvantages will emerge. For example, if the control of an actuator on other WSNs is required after an evaluation, the communication process will again be carried out via MBSS. In this case, four disadvantages will emerge; longer response time, an increase in the traffic between the MBSS and the WSNs, a higher local traffic in the WSNs, and an increase in the average energy consumption in the WSNs.

This study proposes an improvement towards the MBSS based solutions to eliminate the disadvantages mentioned above. The basic idea behind the proposed improvement is to bring the client evaluation entities closer the shared resources. Since the NoBV techniques cannot be used, the closest location is the MBSS. Moving and executing client algorithms on the MBSS may be possible with the MBSS and the clients under the same software framework. In addition, if the framework provides the advantages of agent-based technologies, then a multi-directional interaction between the resources and the clients can be achieved. With the help of client-defined agents, the data from the shared resources is evaluated before coming to the local DA/DF node. Thus, control of the shared actuators or other data management operations (filtering, etc.) can be done in less duration. For this improvement, agent-based FVWSN framework has been developed. With the help of FVWSN, the following contributions can be obtained;

- The client evaluation entities can automatically or remotely be executed by the client agents on the MBSS side. Thus, shorter response time can be achieved.
- A node in a client WSN can interact with the shared resources managed by the client agent-VNd via an M2M middleware without any human involvement.
- The traffic between the MBSS and the clients can be minimized by performing client-defined filtering and analysis operations on the MBSS side.
- Lower local traffic and energy consumption in the WSNs
- A maximum coding simplicity and transparency
- Both publish-subscribe and M2M application supports

C. THE ANALYTIC DETAILS OF THE PROPOSED SYSTEM

In this section, the analytic details will be presented. Here, two purposes are aimed. The first is to explain the fundamental operational restrictions that make the proposed agent-based framework semi-autonomous. The second is to provide an insight into the mathematical implementation of the models developed in the simulation platform.

Before the analytic details, it will be useful that some fundamental explanations and equations are given. The heterogeneous WSNs in an IoT/CPS project are defined in the set $= \{\boldsymbol{W}_1, \boldsymbol{W}_2, .., \boldsymbol{W}_n\}$. Similarly, the shared resources are defined in the set of "R" and R has subsets. These subsets are sensor resources ($\boldsymbol{R_s}$), actuator resources ($\boldsymbol{R_a}$) and other resources ($\boldsymbol{R_o}$), $\boldsymbol{R_s}, \boldsymbol{R_a}, \boldsymbol{R_o} \subseteq \boldsymbol{R}$. The node "i" in $\boldsymbol{W_k}$ is expressed by a 3-tuple notation "$\boldsymbol{N_i^{W_k}}$";

$$N_i^{W_k} = <\acute{P}, \grave{E}, \acute{M}> \tag{1}$$

Here, $\acute{P}$, $\grave{E}$, $\acute{M}$ are the processing rate (MIPS), the available energy level (J) and memory (kB), respectively. For example, for TelosB nodes [9], which are used in the laboratory in which the proposed system has been tested, these values are $\acute{P} = 8\ MIPS$, $\grave{E} = 32400\ J$, $\acute{M} = 10\ kB$. A general notation for a shared resource in the system is;

$$r = <id, a, N_i^{W_k}, r_{pos}>, \quad r \in R \tag{2}$$

There, "id" is a unique identification number given by the system for the resource. "a" is the number of common uses for an actuator resource. The value of "a" is assigned by the system according to the number of the VNds that want to use the resource, and this value is decreased by one when the resource is used. Finally, "$\boldsymbol{r_{pos}}$" gives the coordinates of the resource.

In an MBSS based IoT/CPS project, the average delay between the MBSS system and a client $\boldsymbol{W_k}$ that sends demands for the shared resources is $\dot{D}^{k}_{\boldsymbol{WM}}$. The average delay between sub-WSN $\boldsymbol{W_l}$ and the system is $\overline{\dot{D}^{l}_{MW}}$,, and the average data acquisition time in $\boldsymbol{W_l}$, $\boldsymbol{OT}(\boldsymbol{f^{l}_{get}}(.)) = \overline{\dot{D}^{l}_{get}}$. In $\boldsymbol{W_k}$, the processing time of the data of interest is indicated by $\boldsymbol{OT}(\boldsymbol{f_{yrt}}(.)) = \dot{D}^{k}_{\boldsymbol{DADF}}$, and the average time spent on the MBSS for total queuing and other operations is expressed by $\overline{\dot{D}_S}$. Here, **OT(.)** gives the operation time of the execution function of interest. In this case, for a conventional (Conv for notations) MBSS, $\dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{tot}}$, the total service operation time for "n" shared resources, is;

$$\dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{tot}} = \dot{D}^{k}_{\boldsymbol{DADF}} + \varphi^{Conv}_{WM}\overline{\dot{D}^{k}_{WM}} + \overline{\dot{D}_S} + max(\sum_{l=1}^{n} \varphi^{l}_{MW}\overline{\dot{D}^{l}_{MW}} + \overline{\dot{D}^{l}_{get}}) \quad (3)$$

where "φ^{Conv}_{WM}" and "φ^{l}_{MW}" is the number of the link usage used to sending/receiving demand. These two values depend on retransmission between provider and client, and the post-evaluation actuator control. In addition, if there is a control process for "m" actuator resources at the end of the DA/DF evaluation, the response time, $\dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{tot}}$, will be,

$$\dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{tot}} = \dot{D}^{k}_{\boldsymbol{DADF}} + (\varphi^{Conv}_{WM})\overline{\dot{D}^{k}_{WM}} + \overline{\dot{D}^{l}_{S}} + max(\sum_{l=1}^{n} \varphi^{l}_{MW}\overline{\dot{D}^{l}_{MW}} + \overline{\dot{D}^{l}_{get}}) + max(\sum_{j=1}^{m} \varphi^{j}_{sw}\overline{\dot{D}^{j}_{MW}} + \overline{\dot{D}^{j}_{set}}) \quad (4)$$

In NoBV, the DA/DF operation is run in the local WSN where the shared resources exist, and in this case, $\overline{\dot{D}^{k}_{WM}} = \overline{\dot{D}^{\boldsymbol{l}}_{\boldsymbol{MW}}} = \overline{\dot{D}_S} = 0$. Thus, the response time in NoBV $\dot{D}^{\boldsymbol{NoBV}}_{\boldsymbol{tot}}$;

$$\dot{D}^{\boldsymbol{NoBV}}_{\boldsymbol{tot}} = \overline{\dot{D}^{l}_{DADF}} + \overline{\dot{D}^{l}_{get}} + \overline{\dot{D}^{j}_{set}} \quad (5)$$

$\dot{D}^{l}_{\boldsymbol{DADF}}$ is the time that the DA/DF operation in W_k is run in W_l. Here, It assumed that $\dot{D}^{l}_{\boldsymbol{DADF}} \approx \dot{D}^{k}_{\boldsymbol{DADF}}$. On the other hand, in NeBV, Since $\overline{\dot{D}_S}$*is* 0 and the average access time is $\overline{\dot{D}^{l}_{a}} \approx \overline{\dot{D}^{k}_{WM}} + \overline{\dot{D}^{l}_{MW}}$, the response time is,

$$\dot{D}^{\boldsymbol{NeBV}}_{\boldsymbol{tot}} = \dot{D}^{k}_{\boldsymbol{DADF}} + max(\sum_{l=1}^{n} \varphi^{l}_{a}\overline{\dot{D}^{l}_{a}} + \overline{\dot{D}^{l}_{get}}) + max(\sum_{j=1}^{m} \varphi^{j}_{a}\overline{\dot{D}^{j}_{a}} + \overline{\dot{D}^{j}_{set}}) \quad (6)$$

Consequently, the response time comparison in conventional WSN resource sharing techniques will be,

$$\dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{tot}} > \dot{D}^{\boldsymbol{NeBV}}_{\boldsymbol{tot}} > \dot{D}^{\boldsymbol{NoBV}}_{\boldsymbol{tot}} \quad (7)$$

In the proposed MBSS model, since the DA/DF operation in the client WSN is moved to the MBSS, $\dot{D}_{\boldsymbol{DADF}}$ is included in $\overline{\dot{D}_S}$, and the new expression will be $\dot{D}'_{\boldsymbol{S}} = \dot{D}_S + \dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{DADF}}$. Although the processing rate comparison is $N_i^{W_k}.\acute{P} <$ MBSS.$\acute{P}$, here it is assumed that the DA/DF execution time relation is $\dot{D}^{k}_{\boldsymbol{DADF}} \approx \dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{DADF}}$. In the proposed system, the response time expression will be,

$$\dot{D}^{\boldsymbol{MBSS^{prp}}}_{\boldsymbol{tot}} = \overline{\varphi^{prop}_{WM}\dot{D}^{k}_{WM}} + \overline{\dot{D}'_{S}} + max(\sum_{l=1}^{n} \varphi^{l}_{MBSS}\overline{\dot{D}^{l}_{MW}} + \overline{\dot{D}^{l}_{get}}) + max(\sum_{j=1}^{m} \varphi^{j}_{MBSS}\overline{\dot{D}^{j}_{MW}} + \overline{\dot{D}^{j}_{set}}) \quad (8)$$

Where, since the DA/DF operation is done on the MBSS, the relation of link usage numbers between conventional MBSS system and the proposed system will be $\varphi^{prop}_{WM} < \varphi^{Conv}_{WM}$. In addition, in the proposed system, since the resource data will be brought to the server instead of the client WSN, the access time to the resource will be shorter than that of NeBV ($\dot{D}^{\boldsymbol{j}}_{\boldsymbol{MW}} < \dot{D}^{j}_{a}$). In this case, the comparison between the response times is,

$$\dot{D}^{\boldsymbol{MBSS}}_{\boldsymbol{tot}} > \dot{D}^{\boldsymbol{NeBV}}_{\boldsymbol{tot}} > \dot{D}^{\boldsymbol{MBSS^{prp}}}_{\boldsymbol{tot}} > \dot{D}^{\boldsymbol{NoBV}}_{\boldsymbol{tot}} \quad (9)$$

Moreover, if the DA/DF operation, which is carried out by a demand taken from the client WSN, is performed on the system at a pre-scheduled time, then $\dot{D}^{\boldsymbol{MBSS^{prp}}}_{\boldsymbol{tot}}$ gets close to $\dot{D}^{\boldsymbol{NoBV}}_{\boldsymbol{tot}}$, because $\overline{\dot{D}^{k}_{WM}}$ will be zero.

The main goal of the proposed approach is to execute each client evaluation code individually on the provider side, and for this, all client commands/queries and algorithms have to be resolved automatically on the provider side. In addition, a client WSN may have several DA/DF operations in MBSS, and it could run these operations with different commands/queries that are used in the local client WSN. Therefore, the commands/queries (C/Q) must be interpreted according to the client, MBSS and the other sub-WSN definitions. In the proposed model, while the C/Qs used in clients WSNs are defined in $\boldsymbol{Q} = \{\boldsymbol{Q^{W_1}} \cup \boldsymbol{Q^{W_2}} \cup \ldots \cup \boldsymbol{Q^{W_n}}\}$, the C/Qs used by MBSS are defined in $\boldsymbol{Q^{S}} = \{\boldsymbol{q^{S}_1}, \boldsymbol{q^{S}_2}, ..\}$. Thus, if a client sub WSN "$\boldsymbol{W_k}$" wants to reach a shared resource in the other sub WSN "$\boldsymbol{W_l}$" by a function execution "$\boldsymbol{f_R}$ **(r.id)**", then a consecutive C/Q conversion will be performed in MBSS. To achieve this, the client-defined demand is converted into an appropriate MBSS C/Q with the help of the function $\boldsymbol{g_{ks}}$ **(.)**, $\boldsymbol{g_{ks}} : \boldsymbol{Q^{W_k}} \longrightarrow \boldsymbol{Q^{S}}$". Then, the MBSS command set is converted to sub-WSN C/Q by $\boldsymbol{g_{Sl}}$**(.)** ($\boldsymbol{g_{Sl}} : \boldsymbol{Q^{S}} \longrightarrow \boldsymbol{Q^{W_l}}$).

$$f_R(r) = g_{sk}(g_{ks}(\text{r.id}))$$
$$f_R(r) = g_{sk}\left(q^{S}_{i}, |q^{S}_{i} \in Q^{S}\right) \quad (10)$$
$$f_R(r) = q^{W_l}_{j}(q^{W_l}_{j} \in Q^{W_l}) \quad (11)$$

In the OPNET simulation, for command transitions and conversions, "memory sharing" and "child process calling" are performed according to (11). This process is carried out

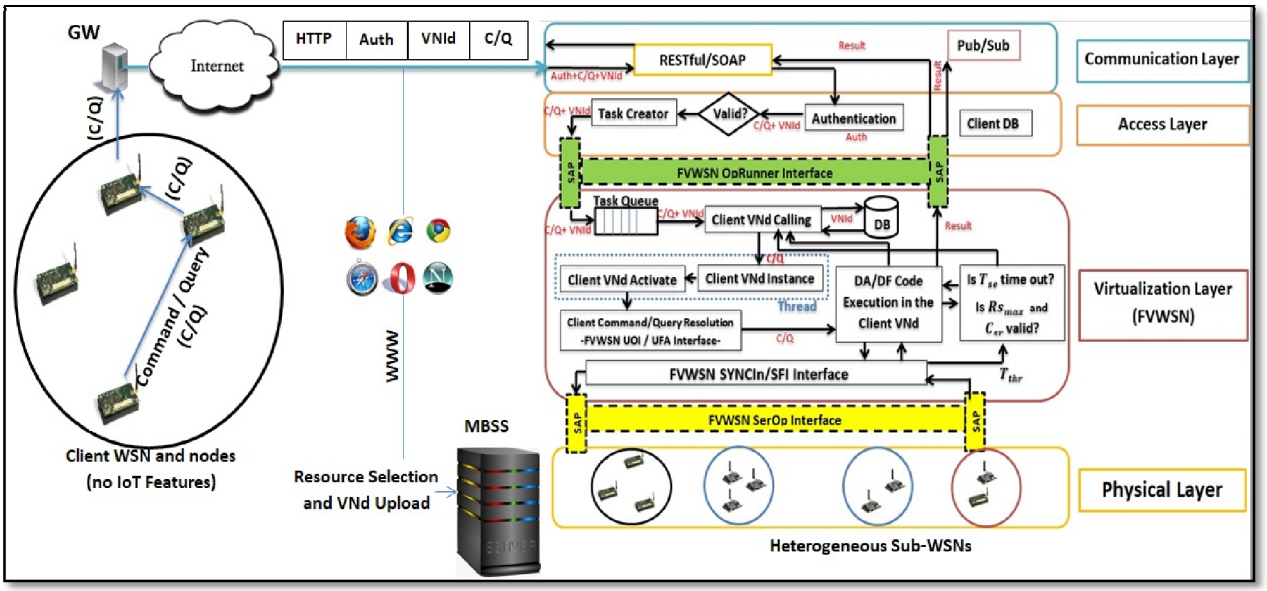


FIGURE 2. The proposed multilayer MBSS model.

automatically, on demand and without pre-set. The main condition here is that the client-WSN is able to introduce its own C/Q set to the MBSS, and to interpret "q_i^S" when the algorithms are executed at run-time. This can be easily achieved by the features that today's Object Oriented Programming Languages offer.

The second improvement is that more client applications, making a resource-sharing, are allowed since the two basic restrictions have been overcome. Those are the node energy ($N_i^{W_k}$.Ṕ) and memory ($N_i^{W_k}$.Ḿ) restrictions. In NoBV, the application footprint is important, and the (12) must be considered;

$$N_i^{W_k}.\acute{\mathrm{M}} > Op_{\acute{\mathrm{M}}} + \sum_{i=1}^{m} A_{\acute{\mathrm{M}}}^{i} \tag{12}$$

In this equation, $Op_{\acute{\mathrm{M}}}$ and $A_{\acute{\mathrm{M}}}^{i}$ are the memory footprint of the embedded operating system and "m" applications, respectively. This is one of the most important factors in NoBV. On the other hand, in the MBSS, these restrictions will be limited by the physical system resources.

In order to monitor the energy levels of the sub-WSNs in the simulations, the energy consumption models of the nodes are important. The energy level of a router/generator node ($N_i^{W_k}$.È) depends on three modes. Those are the power spent in transmitting mode (P_i^{Tr}), the power consumed in receive mode (P_i^{Rc}) and the power spent in execution mode (P_i^{Ex}). Since P_i^{Ex} is very small compared to P_i^{Tr} and P_i^{Rc}, it can be neglected. The general expressions as to the power consumptions are given in (13)-(15). In these equations, B_{ih} represents the data rate (bps) carried out between nodes "i" and "h". In literature, the constants used are $\beta_1 = 50$ nj/bit, $\beta_2 = 0.0013$ pJ/bit/m^4, $\rho = 50$nj/bit ve $\gamma = 4$ [10]. d_{ih} is the distance of interest. Finally, φ_{ih} is the number of the use of route per a flow, which may change according to traffic rate and MAC technique used ($\varphi_{ih} \in Z^+$).

$$P_i^{Tr} = \sum_{h \in W_k, i \neq h} \left(\beta_1 + \beta_2 d_{ih}^{\gamma}\right) B_{ih} \quad \forall \mathrm{i}, \quad \mathrm{h} \in \mathrm{W_k} \tag{13}$$

$$P_i^{Rc} = \rho \sum_{h \in W_k, i \neq h} B_{ih} \quad \forall \mathrm{i}, \mathrm{h} \in \mathrm{W}_k \tag{14}$$

$$P_i^{tot} = \varphi_{ih}(P_i^{Tx} + P_i^{Rx}) \tag{15}$$

As a result, in the case of a non-homogenous resource sharing, the power consumption balance in the entire sub-WSN will also be adversely affected. On the other hand, in the proposed MBSS system, since the data taken from the shared resources are cached for a specific time and the DA/DF operation is performed in the system, the number of the use of routes ($\boldsymbol{\varphi}'$) will be less. Thus, more efficient energy consumption and data traffic goal will be achieved.

III. THE PROPOSED MBSS ARCHITECTURE AND FVWSN FRAMEWORK

In this section, the proposed MBSS architecture and the basic principles of FVWSN will be explained.

A. THE ARCHITECTURE OF THE PROPOSED MBSS AND FVWSN IMPLEMENTATIONS

The multi-layer structure of the proposed system is shown in Fig.2. The structure consists of 4 layers. The heterogeneous WSNs with the shared resources are in the physical (PHY) layer. Virtualization layer is the layer in which the virtualization is carried out and FVWSN is located. The purpose of this layer is to isolate the clients from the complexity due to heterogeneity, and to execute agent VNd operations. The working principle of the system is composed of two fundamental phases. The first one is the registration and resource allocation phase. This phase is carried out via the web. The resource selection/allocation algorithm is presented in [11].

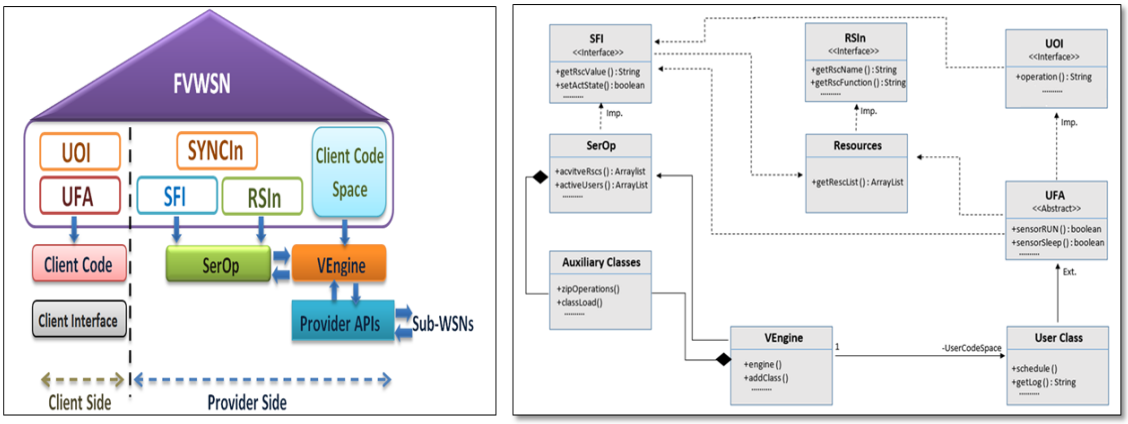
TABLE 2. A Pseudo Code of a Client agent VNd.

```
1 public class virtualNode implements UFA
2 ZigBee Code Definition Class zkt
3 owner ← Client ID
4 node ← VNd Id
5 k← Max recourse reading number
6 Constructor virtualNode()
7         zkt.ProfileID ← "2323", zkt.ClusterID← "0101"
8         setSensorName(node)
8 Override getSensorOwner()
9         return owner
10 Override operation (SFI sf, String CQ)
11        profil_ID ← StringParsing(CQ),  command_ID←StringParsing(CQ)
12    if (zkt.ProfileID ==profil_ID  && zkt.ClusterID  ==command_ID)
13                ArrayList<Double> humidity= sf.getSensorRecords("humidity resource ID", k)
14                ArrayList<Double> temp = sf.getSensorRecords("temperature  resource ID", k)
15                concordant, discordant←0
15                for i=0 : k-1
16                        for j=i+1 : k
17                          if (humidity (i)- humidity (j) and temp(i)-temp(j)  the same direction )
18                                  concordant ++
19                          else
20                                  discordant ++
21
22                KTau Factor←Divide ((concordant-discordant), k(k-1)/2)
23                sf.finished(KTau Factor)
```

During the first phase, the clients get the requisite authorization parameters (username, password, etc.,) and unique IDs for the agent VNds that will be created and uploaded. At the end of the first phase, the client WSN administrator downloads FVWSN and imports it into the IDE platform. The client WSN administrator or designer creates the agent VNds with the help of FVWSN. When doing this, the client WSN designer implements the User Operation Interface (***UOI***) and the User Function Abstract (***UFA***) class methods. These are the abstract methods in which the client WSN commands/queries and DA/DF algorithms are written. An example of a client VNd coded by a client-WSN administrator in the local platform is given in Table 2. In the example, the client WSN runs a ZigBee Application with specific profile ID (profile ID = 2323) and cluster ID in its own network. The aim is to create a VNd that reads the shared resources through the local ZigBee temp/hum command used in the local network (Cluster ID = 0101, Command ID is ignored) and runs a DA/DF operation. The DA / DF operation using the Kendall-Tau method makes an analysis based on the current and a certain number of archive values of shared resources.

After the VNd has been coded and uploaded to the MBSS, any node in the client-WSN or the client administrator can reach it through the local commands. When a VNd is coded, the VNd programmer first implements ***UFA*** abstract class of FVWSN (1). For convenience, the programmer can add a user class with local properties into the VNd (2). The local application properties and the maximum number of archive reading are stored in the appropriate variables of VNd (3-5). The local network properties and the client ID are introduced in the constructor and the methods that are overridden (6-9). The basic client codes or the DA/DF algorithms are written to the overridden method "**operation(SFI,String)**". The method takes two parameters. The first one is the ***SFI*** (Service Functions Interface) parameter. Since this parameter provides the system-defined function interfaces to the programmers, the SFI functions are black boxes for the programmer. The second parameter is the local command/query (C/Q), which is sent from the client WSN through the M2M protocols. The parsing and evaluation of the C/Q are also made by the programmer in "**operation(SFI,String)**". Thus, according to the received command, the programmer can determine which system services will be called or what type of evaluation is to be done (12). The shared resource data is obtained by the method "**getSensorRecords()"**(13-14). The method is overloaded and can take two parameters. The first parameter is the unique ID of the shared resource. The second one is an optional parameter, which is used to get "k" archive data of the interested resource. After the obtained data is evaluated by Kendall's Tau method (15-22), the result is returned (22). Then, the client-defined VNd is uploaded to the provider MBSS via the web. The uploaded VNd can be run in two ways. The first one is to send commands to the VNd in the provider MBSS through M2M-RPC protocols, the second one is to make a pre-scheduled execution on the provider MBSS. In this study, the method of using M2M-RPC will be detailed. A C/Q demand, sent by the administrator or a node in the local WSN, is delivered to the provider-MBSS via SOAP/RESTful based web service in the communication layer.

The demand also includes authentication parameters, the id of the interested VNd. Access layer first checks the



a) FVWSN Interface-Abstract Class Structure

b) The UML Diagram of FVWSN

FIGURE 3. The General Interface-Abstract Class Structure and UML diagram of FVWSN framework [13].

authentication/authorization parameters. A new task that contains the sent C/Q and VNd id is created for the demand passing this checking process. The created task is sent to Virtualization Layer through the instance that implements the OperationRunner Interface of FVWSN. When a task in the queue is executed, firstly VNMonitor checks the VNd Id by looking up the VNd Database. Then, the client-VNd is converted into the instance by using **ClassLoader** and the necessary auxiliary libraries. After activated, the VNd is ready to execute the sent C/Q as a delegated Thread. While the C/Q is parsed and evaluated, SFI and SYNCIn (Synchronization Interface) implementations are utilized. The client operation reaches the shared resources with the help of an implementation of **SerOp** (Service Operations) interface of FVWSN. SerOp is a provider-side implementation and responsible for interacting with the interested sub-WSN APIs. Thanks to this interface instance, transparency is successfully obtained, and the client application is isolated from the infrastructure complexity. The data obtained from the shared resources are evaluated in "***operation(SFI,C/Q)***" of the VNd. After a VNd execution is completed, the result is returned in three ways. The first of these is to use the web service protocols, the second one is Publish/Subscribe middleware, and the third one is to store it in a database. One of the fundamental interfaces is SYNCIn. The instance of this interface meets multiple demands for the same resource in a given period. It achieves this job with a Memcached [12] based caching mechanism. Thus, the local traffic and energy consumption efficiency on the sub-WSN will be provided. On the other hand, while RSIn interface instance is in charge of getting the properties of the shared resources, VEngine is responsible for executing the all VNds that receive a request. The general interface/abstract class structure and UML diagram of FVWSN is given in Fig.3.

B. THE BASIC CONDITIONS FOR VNd EXECUTION

When the system is working, some constraints that make the developed agent framework semi-autonomous are of paramount importance. As mentioned above, the fundamental principle of the system is that the function "$f_{op_i}(q^{I_P}), (q^{I_P} \in Q^{I_P})$" that is implemented by the client is executed by the system. This operation is performed by the function "$V_{op}(.)$" provided by VEngine. Thus, an agent VNd execution is expressed as "$V_{op}(f_{op_i}(q^{I_P}))$". On the other hand, there are some restrictions on the system execution. The first of the restrictions is the pre-defined "T_{se}", which shows the service time out duration for a VNd execution. Thus, the execution duration for an agent VNd is;

$$OT(V_{op}(f_{op_i}(q^{I_P}))) < T_{se} \tag{16}$$

In addition, another important restriction is the data acquisition duration from the sub-WSN. For a VNd that uses different shared resources in different sub-WSNs, the data acquisition duration will directly affect $OT(V_{op}(f_{op_i}(q^{I_P})))$. Some reasons, such as the traffic load in sub-WSNs, lead to exceeding "T_{se}" and an unsuccessful VNd execution. For this, the system uses a threshold variable "T_{thr}" for a sub-WSN. The threshold is used to monitor the statistical status of access/read durations of a resource, and to determine whether re-sending is required. It is not necessary that this value is the same for each resource, but it cannot be less than "ε", a predefined system constant. In terms of execution performance, this value can be optimized for each resource and called "T_{ro}". T_{ro} cannot be less than T_{thr}. In other words, $T_{ro} \geq T_{thr} \geq \varepsilon$ is always valid. If an optimized value is not assigned to T_{ro}, then T_{ro} is equal to T_{thr} ($T_{ro} = T_{thr}$). Accordingly, in an agent VNd execution, it can be calculated how many readings can be done by,

$$C'_{sr} = \frac{T_{ro}}{T_{thr}} \tag{17}$$

However, this value must always be controlled by the system because if the value is high, the traffic in sub-WSNs and the system performance will be adversely affected.

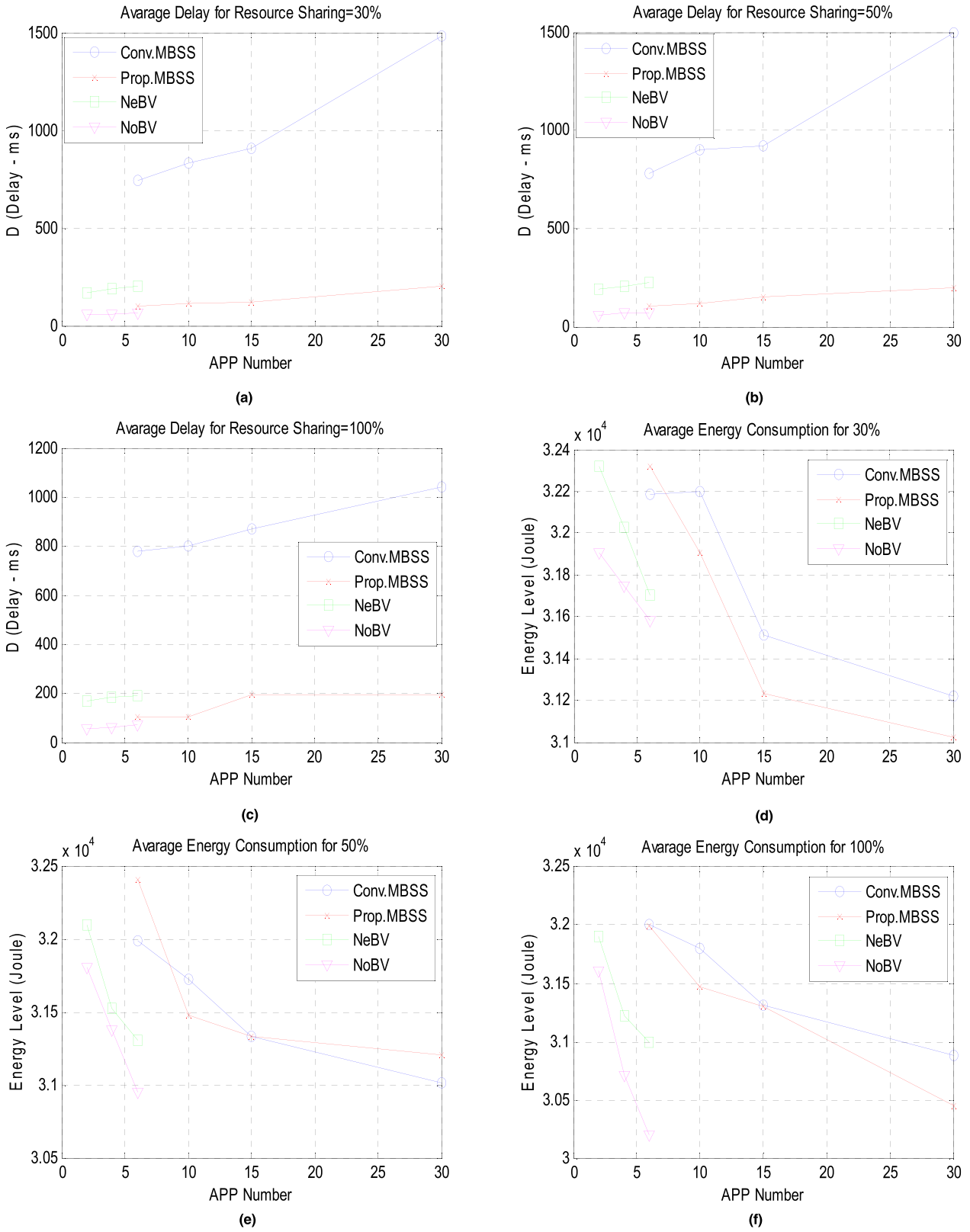


FIGURE 4. NoBV, NeBV, Conventional MBSS and the proposed system comparison in terms of delay and energy level. (a) Delay performance for resource sharing 30%(b) Delay performance for resource sharing 50% (c) Delay performance for resource sharing 100%. (d) Average residual energy for res.shar. 30% (e) Average residual energy for res.shar. 50% (f) Average residual energy for res.shar. 100%.

On the other hand, in terms of execution efficiency, it is not recommended that a VNd has a number of shared resources. Therefore, two restrictions are used in a VNd execution. The first one is the re-reading number of a resource. The second one is the maximum resources that an agent VNd can use. Let $\boldsymbol{\psi}(.)$ and (.) give the reading number of a resource and the number of a resource the agent VNd has read, respectively, in this case, the two conditions are,

$$\psi\left(\mathrm{V_{op}}\left(\mathrm{f_{op_i}}\left(\mathrm{q^{I_p}}\right)\right)\right) \leq \mathrm{C'_{sr}} \leq \mathrm{C_{sr_{max}}} \tag{18}$$

$$(\mathrm{V_{op}}\left(\mathrm{f_{op_i}}\left(\mathrm{q^{I_p}}\right)\right)) \leq Rs_{max} \tag{19}$$

Another important situation is the system congestion. This situation depends on the relation between $\boldsymbol{T_{ro}}$ and the service duration each task receives from SerOp. As proved in [13], the prototype of the system uses an M/M/1/N queue structure. TaskQueue in the virtualization layer prefers a time window to control the congestion of the system. The tasks, which are outside of the window, are removed from the queue and a resending is wanted from the client. According to the basic queue theorem, if the rate between the average arrival rate "λ" and service rate " μ" is bigger 1, then the delay in the system will increase. Considering the general system-defined restrictions, the maximum input number "$\boldsymbol{N_{max}}$"" will determine the size of the time window in TaskQueue. With the assigned "$\boldsymbol{N_{max}}$" ($\boldsymbol{N_{max}}$ = N), the average agent task number (O^s_{vt}) can be found by,

$$O^s_{vt} = \left[\frac{\lambda}{\mu}\right]\left[\frac{1-(N+1)\left[\frac{\lambda}{\mu}\right]^N + N\left[\frac{\lambda}{\mu}\right]^{N+1}}{(1-\left[\frac{\lambda}{\mu}\right])(1-\left[\frac{\lambda}{\mu}\right]^{N+1})}\right]\left(\left[\frac{\lambda}{\mu}\right] \neq 1\right) \tag{20}$$

In addition, the agent VNd task number in only the queue (O^{queue}_{vt}) will be,

$$O^{queue}_{vt} = O^s_{vt} - \left(1 - \frac{1-\left[\frac{\lambda}{\mu}\right]}{1-\left[\frac{\lambda}{\mu}\right]^{N+1}}\right) \tag{21}$$

The average waiting duration of each task and the waiting duration in the queue can be calculated according to (22) and (23).

$$B^s_{vt} = \frac{O^{queue}_{vt}}{\lambda(1-\mathrm{Sg_N})} + \frac{1}{\mu} \tag{22}$$

$$B^{queue}_{vt} = B^s_{vt} - \frac{1}{\mu} \tag{23}$$

Here, $\mathrm{Sg_N} = \left[\frac{\lambda}{\mu}\right]^{\mathrm{N}} Sg_0$ and $Sg_0 = \frac{1-\left[\frac{\lambda}{\mu}\right]}{1-\left[\frac{\lambda}{\mu}\right]^{N+1}}$. Consequently, the system always control the condition "$\boldsymbol{T_{ro}} \geq \boldsymbol{B^{queue}_{vt}}$" to avoid a congestion.

IV. THE SIMULATION OF THE PROPOSED SYSTEM

The advantages and success of the proposed system are firstly tested on the OPNET Modeller simulator platform [14], [15].

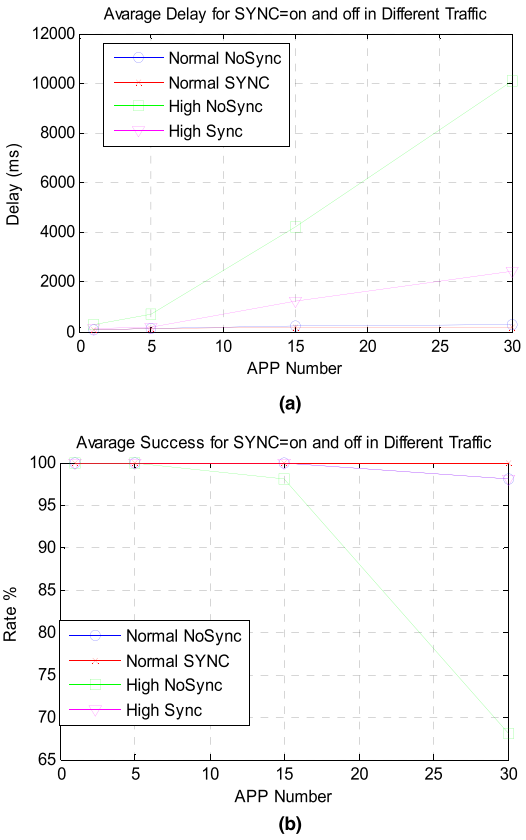


FIGURE 5. Average delay and success for SYNC = on and off in different traffic. (a) Delay for SYNC = on and off in different traffic (b) The success of SYNC = on and off in different traffic.

For a clear evaluation, the performance of the proposed MBSS system has been compared with the performances of NoBV, NeBV and conventional MBSS methods. In the simulation scenario, a client WSN uses different shared resources in different sub WSNs, and performs a DA/DF operation with the data from the shared resources. After the DA/DF, the client application controls another actuator resource shared by a sub-WSN. The basic performance criteria are response time and the number of client applications that utilize the shared resources. In addition to these, to provide an insight into the goal, the average energy consumption of each sub-WSN is also discussed. The simulations have been done for both a normal traffic and high traffic. The some simulation screenshots are given in Appendix. The other simulation parameters are presented in Table 3 and 4. For NoBV simulations, the maximum different applications and node features have been determined as 6 and TelosB, respectively. The simulation scenarios have been separately run according to different shared resource ratios. Thus, it has been tried to address the results for different demanding situations. The results are firstly examined in terms of delay in the response time and the number of client applications. Subsequently, for different shared resource ratios, the average residual energy amount in the sub-WSNs is discussed. Finally, to get an idea

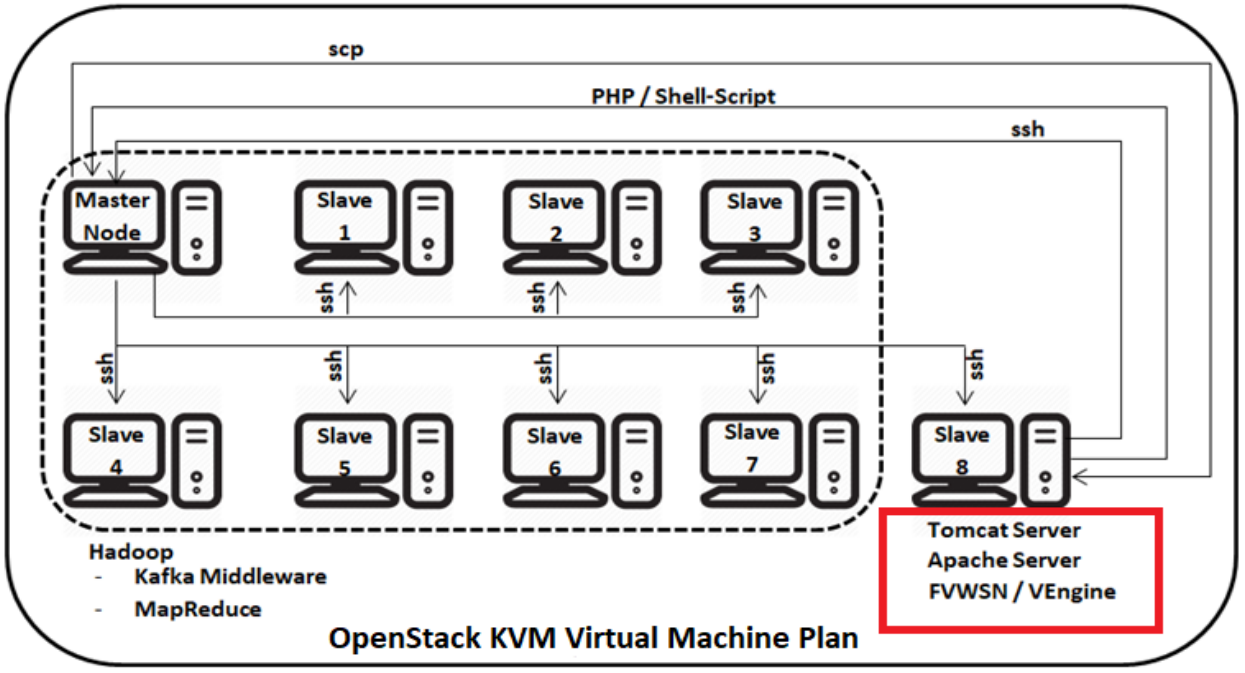


FIGURE 6. The Virtual Machines plan on OpenStack and FVWSN Framework integration.

TABLE 3. Simulation parameters.

Parameters	Value
PHY frequency of sub-WSNs	868/915/2400 MHz
ACK	yes
CSMA/CA	yes
Min.Backoff Exp.	3
Max.Backoff	4
Bandwidth	800-2000 kHz
Modulation	DSSS/FHSS

TABLE 4. Operational parameters.

Parameters	Value
APP number for NoBV	2-6
The type of DA/DF code	*O(n²) Scalar correlation*
APP type	ZigBee, SQL-like
È / Ḿ	32400 joule /10kB
a	3
Sub-WSN number	3 (150mx150m)
The rate resource sharing	30-50-100 %
$C_{sr}/R_{max}/T_{se}$ /T_{thr}	3/2/10s/4s

about the success of the proposed system, the number of successful executions done in the system is analyzed.

In the simulation, the first focus point is whether (8) is valid or not. As mentioned earlier, the main goal of the proposed system is to get close to the NoBV response time. In addition, since the agent VNds reduces the delay between the shared resource and the client evaluation entities ($\dot{D}^{j}_{MW} < \overline{\dot{D}^{j}_{a}}$), the response time in the proposed system will be less than that of NeBV. In order to able to make a general comparison, NoBV and NeBV techniques are included into simulations. In Fig. 4.a-c, it can be seen that the state, $\dot{D}^{MBSS}_{tot} > \dot{D}^{NoBV}_{tot} > \dot{D}^{MBSS^{prp}}_{tot} > \dot{D}^{NoBV}_{tot}$, can be provable. In Fig. 4.a-c, the response times obtained are given for 30-50 and 100% resource sharing. As is seen, while the best response time has been achieved in NoBV (purple), the worst value has been taken from the conventional MBSS system (blue). The result of the proposed system (red) is between NoBV and NeBV (green), and it has been achieved an improvement of 84.67% compared to conventional MBSS. Besides, as the number of client application increases, the response time has not changed (Mean Square Error = 4.2%). In terms of energy consumption, the proposed MBSS has 14.23% less performance compared to the conventional MBSS. The reason for this is that the waiting periods of the nodes in the sub-WSNs with shared resources are longer that of the conventional MBSS. This means that the proposed system takes a shorter response time by waiving a small amount of energy consumption.

On the other hand, NoBV and NeBV have more energy consumption value for more client applications. As mentioned earlier, the proposed system has two operating modes.

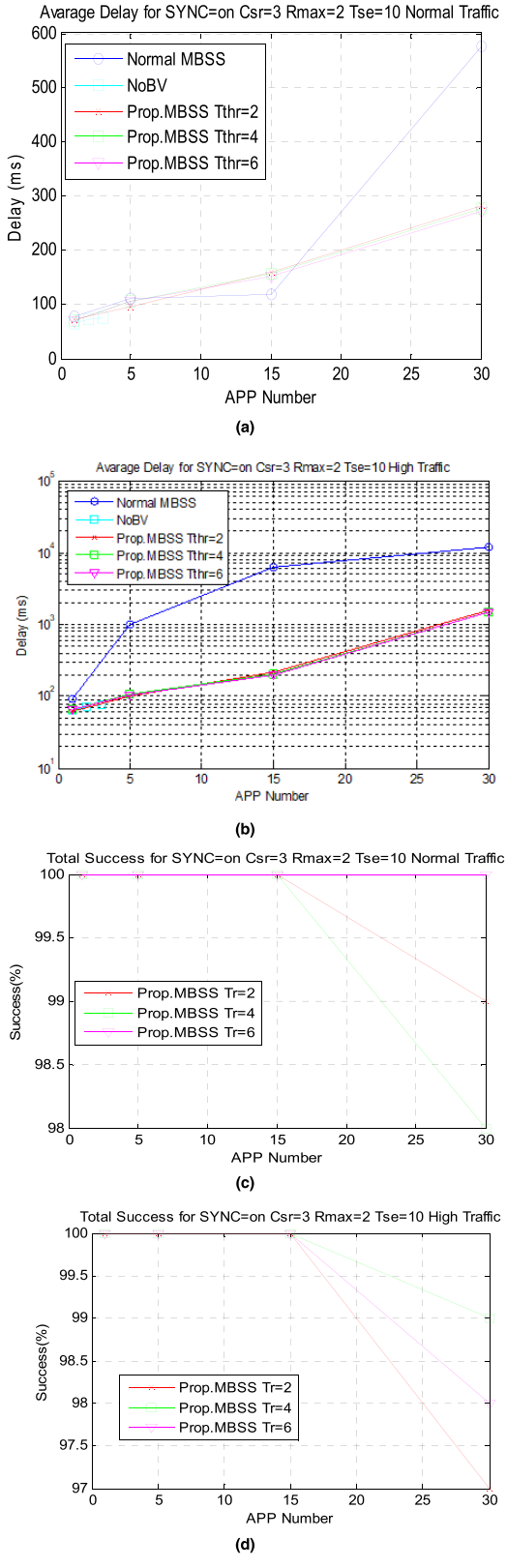


FIGURE 7. The test results of the proposed system on physical cloud system. (a) Average Delay for different Tthr value in normal traffic. (b) Average Delay for different Tthr value in high traffic. (c) Total success for different Tthr value in normal traffic. (d) Total success for different Tthr value in high traffic.

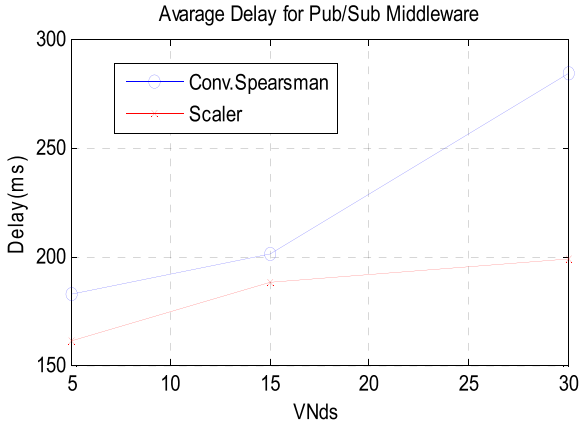


FIGURE 8. Average Pub/Sub delay for the topics opened by different numbers VNds running DF algorithms.

The first is the SYNC mode in which the data from sub-WSNs are cached for a calculated duration. In this mode, cached data is shared with other demands made within this period. However, SYNC mode can be turned off in some situations such as security and priority tasks. This mode is called NoSYNC (SYNC = off). Both modes have been tested and compared in the simulations. The first scenario is the normal traffic condition in which periodical demands are sent to the system. The second scenario is the high traffic condition in which all clients send nonstop demands to the system. As shown in Fig.5a, in NoSYNC mode, while the proposed system has a response time of 286ms in normal traffic condition, it reaches "T_{se}" for 30 APPs in the high traffic condition. In NoSYNC mode and high traffic, some improvements can be made by using different queue structure and different communication techniques between sub-WSNs and the system. These improvements are under development. Besides, in high traffic and SYNC mode, the proposed system reaches 24% of T_{se} at most. In Fig.5b, the successful responses received without exceeding T_{se} are evaluated. Here, the DA/DF operations that do not exceed T_{se} are considered as SUCCESSFUL. In both normal and high traffic SYNC mode, the success of the system is 99% in average.

V. THE IMPLEMENTATION OF THE PROPOSED SYSTEM ON A CLOUD SYSTEM

Finally, the proposed system has been tested on a real cloud platform [13] that contains real sub-WSNs. The general parameters of the test platform are given in Table 5 and 6. The cloud platform runs on three Lenovo servers running OpenStack [15], [16], a cloud operating system. The cloud system also uses Apache Kafka middleware [18] and SOAP based web service. While some screenshots about the cloud system are presented in Appendix, the virtual machine structure of the test platform, created by KVM on OpenStack, is given in Fig.6.

There are 8 virtual machines (VMs) on the cloud platform. While 7 VMs run Hadoop Ecosystem [19], one VM is used

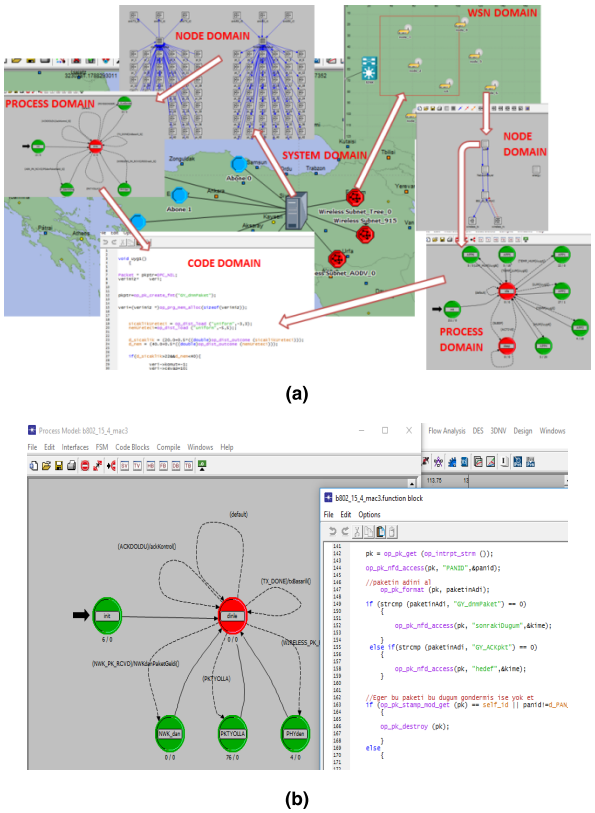


FIGURE 9. Some screenshots from OPNET models of the system [13]. (a) General OPNET simulation structures. (b) An example layer FSM implementation.

TABLE 5. Server parameters.

Parameters	Value
Servers	3 x Lenovo x3650
Processor	2x Xenon 2.4GHz
Memory	48/32/16 GB
OS	Ubuntu 16.04
Cloud OS	OpenStack Newton
Compute/Controller	2/1
Comm. Layer Protocol	SOAP Web Service

to provide the proposed multilayer model. The inner interactions between VMs are made by virtual network, ssh and shell scripts. In the tests conducted on the cloud platform, the recommended default parameters were used. They are respectively 3/2/10s for $C_{sr}/R_{max}/T_{se}$. The access restriction to the resources, T_{thr}, was individually optimized and assigned to each shared resource by the system. Therefore, in normal and high traffics, the tests were performed for different T_r values. In order to make a comparison, TelosB nodes in the laboratory, running TinyOS, were used for NoBV-based tests. Due to memory restrictions, up to three scalar applications could be installed on the nodes. In addition, conventional MBSS using RPC was run over the same system. As is shown in Fig.7.a-b, in normal traffic, the

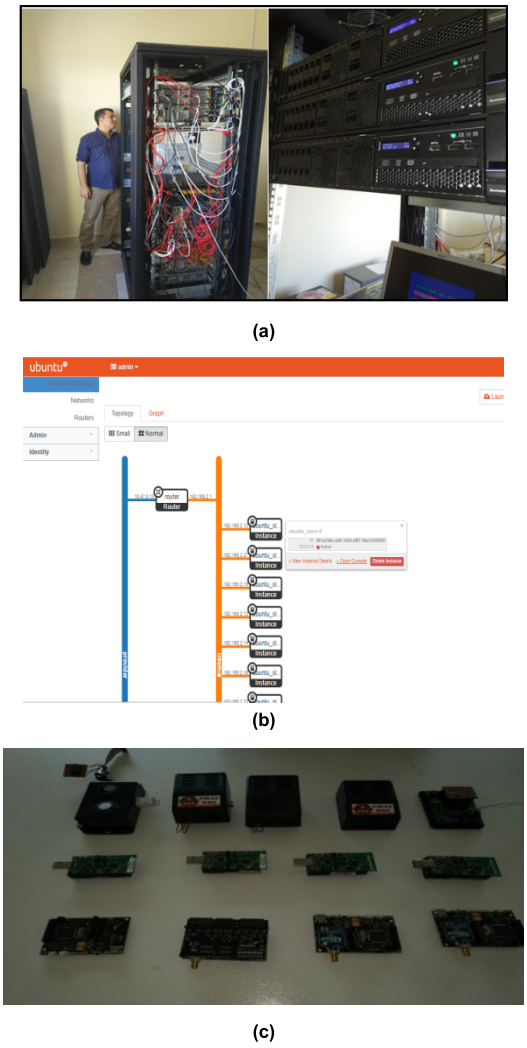


FIGURE 10. Some modules in the physical test platform [13] (a) Physical Servers. (b) The Virtual Machines and network created in OpenStack. (c) The WSN nodes used.

TABLE 6. Sub-WSN parameters.

Parameters	Value
Total Resource	64
Nodes	*TelosB, FiratNode, Libelium Waspmote,, Digi XBee*
Resource Sharing Rate	100 %
Sensors	Temp, Hum, Light, O2
Actuators	Relays
a	3
$C_{sr}/R_{max}/T_{se}/T_{thr}$	3/2/10/2-4-6

conventional MBSS obtained a response time of 577ms at most, while the proposed MBSS could achieve a response time of 283ms at most. NoBV achieved a response time of 77ms. On the other hand, in the high traffic tests, while the longest delay is 11.98s for the conventional MBSS,

the response time is 1.59 at most for the proposed system. Besides, Fig.7 c-d shows that the proposed system has achieved a success rate of 98.5 in average in normal and high traffic.

FVWSN framework also gives a pub/sub middleware interaction support like other agent based technologies [20]. With this feature, agent VNds can open topics on the pub/sub middleware and serve other registered clients. The feature, especially enables meaningful data to be shared with the interested clients. In the test platform, Apache Kafka was used as a middleware. Fig.8 shows the delay results obtained from the topics opened by of the VNds running a scalar and Spearsman Rank Correlation.

VI. WEAKNESSES OF THE SYSTEM AND FUTURE STUDIES

As with other agent-based technologies, the security is an important handicap in the proposed model. Although the system uses standard security policies and authorization mechanisms, the security issue is a title that needs to be studied in detail. In addition to this, service quality can be improved by changing and optimizing queuing parameters, T_{se}/T_{thr} values, the access time to the services and communication technologies. Future studies are going to focus on the security, multi-agent support for different client funds, and the use of optimized parameters.

VII. CONCLUSION

In this study, we focused on the ability of WSNs working under IoT/CPS projects to make an interactive resource sharing with each other. Because of characteristic WSN restrictions and heterogeneity, conventional NoBV and NeBV methods are not too efficient for the resource sharing scenarios in IoT/CPS scale. On the other hand, long response delays are a significant disadvantage in resource sharing mechanisms with RPC-based MBSS, which especially contain actuator controls. In the study, we introduced a semi-autonomous agent based WSN resource sharing approach, which can serve more client applications and has a shorter response time. Here, the basic logic is that the algorithms of client applications are moved to the nearest point to shared resources. Considering heterogeneity, a large number of clients and WSN node limitations, this point is MBSS provider. However, this may be possible with a design in which MBSS and all clients are under the same software framework. For this, a software agent framework, called FVWSN, has been developed. With the help of FVWSN, clients can create the logical agents, executing the interested DA/DF algorithms, on the MBSS provider. These logical agents, called VNds, can be remotely controlled and run by the same commands/queries used in the local client WSNs. Depending on the commands/queries from the client, a VNd can execute different control and evaluation mechanisms. Thus, a more flexible platform, which can run client-defined services instead of provider-defined services, has been created. In this way, the evaluation of the data obtained from the shared sources and, if necessary, the control of the actuators can be provided in a shorter time. After making the analytical analysis of the proposed approach, general performance comparisons with other technics are shown on the OPNET Model simulation platform. The comparative success of the approach has been also proven on an OpenStack based Cloud system.

APPENDIX

The simulation of the system was carried out in OPNET platform. For heterogeneous sub-WSNs, different WSN nodes with different multilayer structure were created first (Fig.AppI.a). Then gateway and MBSS systems were created. For VNd executions, child processes that perform the memory allocation were used. The FSM diagram of some components and other simulation outputs are presented Fig.8.a-b.

Some screenshots of the physical test platform are shown Fig 9. a-d.

ACKNOWLEDGMENT

The authors would like to thank to the department of FUBAP of F.U. for their supports in the project, ''Mobile Communication Technologies and Wireless Sensor Networks Systems''.

REFERENCES

[1] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, ''Wireless sensor network virtualization: A survey,'' *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 553–576, 1st Quart., 2016, doi: 10.1109/COMST.2015.2412971.

[2] M. Sudip, S. Chatterjee, and M. S. Obaidat, ''On theoretical modeling of sensor cloud: A paradigm shift from wireless sensor network,'' *IEEE Syst. J.*, vol. 11, no. 2, pp. 1084–1093, Jun. 2017, doi: 10.1109/JSYST.2014.2362617.

[3] G. Yıldırım and Y. Tatar, ''On WSN heterogeneity in IoT and CPSs,'' in *Proc. Conf. UBMK*, Antalya, Turkey, Oct. 2017, pp. 1020–1024, doi: 10.1109/UBMK.2017.8093421.

[4] *Sentilo Project*. Accessed: Dec. 5, 2018. [Online]. Available: http://www.sentilo.io

[5] *Thingspeak Project*. Accessed: Dec. 5, 2018. [Online]. Available: https://thingspeak.com

[6] *IoTSense Project*. Accessed: Dec. 5, 2018. [Online]. Available: http://www.iotsens.com/en

[7] *Digi Device Cloud Project*. Accessed: Dec. 5, 2018. [Online]. Available: https://www.digi.com/products/cloud/digi-remote-manager

[8] F. Bellifeminea, G. Caire, A. Poggi, and G. Rimassa, ''JADE: A software framework for developing multi-agent applications. Lessons learned,'' *Inf. Softw. Technol.*, vol. 50, nos. 1–2, pp. 10–21, Jan. 2008, doi: 10.1016/j.infsof.2007.10.008.

[9] *TelosB WSN Node*. Accesses: Dec. 5, 2018. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

[10] Y. Shi, Y. T. Hou, J. Liu, and S. Kompella, ''Bridging the gap between protocol and physical models for wireless networks,'' *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1404–1416, Jul. 2013.

[11] G. Yıldırım and Y. Tatar, ''Alternative resource allocation model for dynamic resource sharing WSN systems,'' in *Proc. 5th Int. Conf. Adv. Comput., Electron. Commun. (ACEC)*, Rome, Italy, May 2017, pp. 15–19, doi: 10.15224/978-1-63248-121-4-04.

[12] *Memory Object Caching System*. Accessed: Dec. 5, 2018. [Online]. Available: https://memcached.org/

[13] G. Yıldırım, ''Design of a distributed-parallel cyber physical system based on virtual wireless network,'' Ph.D. dissertation, Dept. Comput. Eng., Firat Unv., Elazig, Turkey, 2017.

[14] J. Gao, W. Tong, X. Jin, Z. Li, and L. Lu, ''Study on communication service strategy for congestion issue in smart substation communication network,'' *IEEE Access*, vol. 6, pp. 44934–44943, 2018, doi: 10.1109/ACCESS.2018.2863725.

[15] *Opnet Modeller*. Accessed: Dec. 5, 2018. [Online]. Available: https://www.riverbed.com/sg/products/steelcentral/opnet.html
[16] Y. Yamato, Y. Nishizawa, S. Nagao, and K. Sato, "Fast and reliable restoration method of virtual resources on OpenStack," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 572–583, Apr./Jun. 2018, doi: 10.1109/TCC.2015.2481392.
[17] H. J. Syed, A. Gani, F. H. Nasaruddin, A. Naveed, A. I. A. Ahmed, and M. K. Khan, "CloudProcMon: A non-intrusive cloud monitoring framework," *IEEE Access*, vol. 6, pp. 44591–44606, 2018, doi: 10.1109/ACCESS.2018.2864573.
[18] J. Bang, S. Son, H. Kim, Y. Moon, and M. Choi, "Design and implementation of a load shedding engine for solving starvation problems in Apache Kafka," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Taipei, Taiwan, Apr. 2018, pp. 1–4, doi: 10.1109/NOMS.2018.8406306.
[19] P. Merla and Y. Liang, "Data analysis using hadoop MapReduce environment," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Boston, MA, USA, Dec. 2017, pp. 4783–4785, doi: 10.1109/BigData.2017.8258541.
[20] *Jade Pub/Sub Support*. Accessed: Dec. 5, 2018. [Online]. Available: https://github.com/mihaiparaschiv/university-jade-pubsub
[21] C.-L. Fok, G.-C. Roman, and C. Lu, "Agilla: A mobile agent middleware for self-adaptive wireless sensor networks," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 3, pp. 16:1-16:26, Jul. 2009.
[22] Y. Yu, L. J. Rittle, V. Bhandari, and J. B. LeBrun, "Supporting concurrent applications in wireless sensor networks," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst.*, New York, NY, USA, 2006, pp. 139–152.
[23] M. Navarro, M. Antonucci, L. Sarakis, and T. Zahariadis, "VITRO architecture: Bringing virtualization to WSN world," in *Proc. IEEE 8th Int. Conf. Mobile Ad-Hoc Sensor Syst.*, Oct. 2011, pp. 831–836, doi: 10.1109/MASS.2011.96.
[24] J. Hoebeke, E. De Poorter, S. Bouckaert, I. Moerman, and P. Demeester, "Managed ecosystems of networked objects," *Wireless Pers. Commun.*, vol. 58, no. 1, pp. 125–143, 2011.
[25] E. de Poorter, E. Troubleyn, I. Moerman, and P. Demeester, "IDRA: A flexible system architecture for next generation wireless sensor networks," *Wireless Netw.*, vol. 17, no. 6, pp. 1423–1440, Aug. 2011.
[26] D. Villa, F. Moya, F. J. Villanueva, O. Acena, and J. C. López, "Ubiquitous virtual private network: A solution for WSN seamless integration," *Sensors*, vol. 14, no. 1, pp. 779–794, 2014, doi: 10.3390/s140100779.
[27] *WaspMote Nodes*. Accessed: Dec. 5, 2018. [Online]. Available: http://www.libelium.com
[28] *SensorRush Project*. Accessed: 2017. [Online]. Available: http://sensorrush.com
[29] G. Hong, X. Fang, J. Li, and Y. Li, "Data collection in multi-application sharing wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 403–412, Feb. 2015, doi: 10.1109/TPDS.2013.289.
[30] M. A. B. Brasil, B. Bösch, F. R. Wagner, and E. P. de Freitas, "Performance comparison of multi-agent middleware platforms for wireless sensor networks," *IEEE Sensors J.*, vol. 18, no. 7, pp. 3039–3049, Jan. 2018, doi: 10.1109/JSEN.2018.2791416.
[31] Y. Zhao, Q. Wang, D. Jiang, W. Wu, L. Hao, and K. Wang,"An agent-based routing protocol with mobile sink for WSN in coal mine," in *Proc. 3rd Int. Conf. Pervasive Comput. Appl.*, Alexandria, Egypt, Oct. 2008, pp. 857–862, doi: 10.1109/ICPCA.2008.4783730.
[32] F. Bai, K. S. Munasinghe, and A. Jamalipour, "A novel information acquisition technique for mobile-assisted wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 61, no. 4, pp. 1752–1761, May 2012, doi: 10.1109/TVT.2012.2188657.
[33] S. Sasirekha and S. Swamynathan, "Cluster-chain mobile agent routing algorithm for efficient data aggregation in wireless sensor network," *J. Commun. Netw.*, vol. 19, no. 4, pp. 392–401, 2017, doi: 10.1109/JCN.2017.000063.
[34] H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, "RWiN: New methodology for the development of reconfigurable WSN," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 109–125, Jan. 2017, doi: 10.1109/TASE.2016.2608918.
[35] A. Boulis, C.-C. Han, and M. B. Srivastava, "Design and implementation of a framework for efficient and programmable sensor networks," in *Proc. ACM MobiSys*, May 2003, pp. 187–200.
[36] P. Levis *et al.*, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Springer, 2005, doi: 10.1007/3-540-27139-2_7.
[37] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462, doi: 10.1109/LCN.2004.38.
[38] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 79–80, doi: 10.1109/INFCOMW.2013.6970748.
[39] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The liteos operating system: Towards unix-like abstractions for wireless sensor networks," in *Proc. IPSN*, Apr. 2008, pp. 233–244.

GÜNGÖR YILDIRIM received the B.S. degree in electrical and electronic engineering from Fırat University, Elazig, and the M.S. and Ph.D. degrees in computer engineering from Firat University in 2012 and 2017, respectively. He served as the Head of the Electric Program at Tunceli MYO for three years. His research interests include wireless sensor networks, IoT systems, and electromagnetic propagation. In 2014, he received the Award of Appreciation in DSI.

YETKİN TATAR received the B.Sc. degree from EDMMA in 1974 and the M.Sc. and D.Sc. degrees in electrical and electronic engineering from Fırat University, Turkey, in 1984 and 1994, respectively. He is currently a Professor with the Department of Computer Engineering, Fırat University. His research areas are wireless sensor networks, computer networks, and network security.