Logical Space Composition of IoT for a Scalable and Adaptable Smart Environment

Hyeyoung An, Woojin Park, and Soochang Park
Department of Computer Engineering
Chungbuk National University
Cheongju, Republic of Korea
{elinyoung, woojin415, cewinter}@chungbuk.ac.kr

Euisin Lee
School of Information & Communication Engineering
Chungbuk National University
Cheongju, Republic of Korea
elsee@chungbuk.ac.kr

***Abstract*—The Internet of Things (IoT) represents the integration of smart environments seamlessly woven into the fabric of our daily lives. However, the inherent characteristics of IoT, such as its vast scale and heterogeneity, have posed challenges in the development of practical and smarter applications, primarily due to the lack of an architecture capable of effectively managing large-scale IoT systems. In response to these challenges, this paper introduces a Logical Space Subdivision (LSS) system. This system modularly configures smart devices, taking into account user requirements and adapting to spatiotemporal changes. These modules are interconnected to establish logical spaces that can be combined to represent more extensive physical spaces. The paper delineates five fundamental design principles: modularity, encapsulation, authorization, interoperability, and mobility. The proposed system facilitates scalable and flexible service provisioning within IoT infrastructures, enhancing the adaptability of smart spaces to meet users' evolving needs.**

***Index Terms*—IoT, Modularity, Encapsulation, Logical space**

I. Introduction

In recent decades, the advancements in computing and wireless communication technologies have significantly improved our daily lives by providing a highly convenient channel for information exchange, offering end-users efficient resource utilization and intelligent services [1]. The interconnection of all things paves the way for the Information and Communication Technology (ICT) revolution known as the Internet of Things (IoT), where numerous sensors, actuators, and mobile devices are deployed at the network edge [1], [2]. IoT is a fusion of two fundamental elements "Internet" and "Things." In other words, connected devices, called things, are embedded into our life spaces for carrying out daily life activities and tasks in an easy and natural way using information and intelligence, hidden in the network connecting the things [3], [4]. Such a pervasive vision of IoT might increase the value of the information generated by a number of interconnections between people and things and the transformation of the processed information into knowledge for the benefit of mankind and society.

In accordance with a Statista Research Development report from November 2019, it is anticipated that the global count of interconnected IoT devices will soar to 75.44 billion by the year 2025 [5]. This signifies the involvement of billions of interconnected devices in intelligent services all around us. An end user is able to receive/exploit various smart services

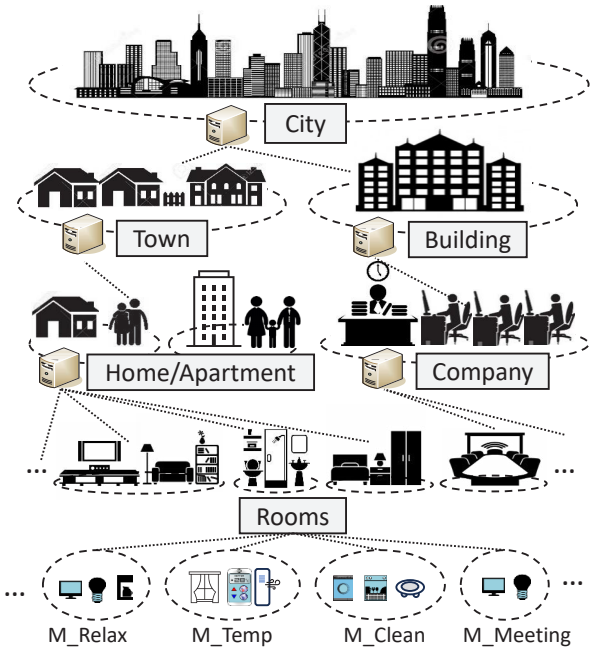

Fig. 1. Hierarchy of logical space

that are provided by a wide variety/range of smart devices in any place at any time. Smart device services need to change dynamically to users' spatiotemporal variations, as Fig. 1 illustrates. In other words, as you wake up in the morning and get ready, users should be able to seamlessly interact with smart devices, such as thermostats or heaters, placed in specific domestic spaces like your bedroom and bathroom. Similarly, when you have breakfast or watch the news, the kitchen and living room should be connected to the relevant smart devices, such as the TV, dishwasher, oven, etc. This does not simply mean a change in space. Each space is composed of different smart devices with heterogeneous properties, such as the type, function, and management service provider of the connected smart device [1], [2], [6]. Today, most service providers are taking a Top-down approach, passively collecting the necessary data from individual devices. Furthermore, each individual device is limited by the monitoring tools provided by the device vendor, which also makes it difficult to extract

useful information [7]. Therefore, it prevents scalable and flexible service provision and interoperation management over broadly deployed IoT infrastructures.

To address the challenges and limitations associated with the heterogeneity of large-scale IoT, including issues related to flexibility, scalability, and interoperability, this paper introduces a Logical Space Subdivision (LSS) system. In the proposed system, smart devices within a user's space are modularly configured based on the user's requirements, taking into account the user's spatiotemporal changes. Modules within the space are interconnected to form a single logical space. Furthermore, these logical spaces can be combined with other logical spaces, ultimately representing larger physical spaces. As a result, our system can effectively manage large and highly diverse IoT infrastructures. It does so with low complexity in control and communication while ensuring high interoperability among the various smart objects involved in smart services.

In the following sections, we discuss the basic design principles to achieve practical goals. In Section III, we explain our proposed LLS System. In Section IV, we present the performance evaluation results along with usage scenarios to validate the proposed approach. The conclusion section of this article highlights the main concept.

II. Basic Design Principles

This section explains the basic design principles to achieve the purpose of the proposed LSS system. The core vision is that smart devices near the user are logically organized by modularizing similar types, and these modules are combined to form a single logical space. The logical spaces are combined to form a larger logical space at the top. This bottom-up approach allows for the dynamic arrangement of smart spaces. The following subsections present the five design principles.

A. Modularity

This system identifies modularity as one of the key design principles to establish independence among interconnected elements. These elements are capable of connecting, interacting, and exchanging data with each other. Modularity, therefore, is defined as the principle that states related elements should remain unaffected by additions, deletions, modifications, or reconstructions. Unlike a tightly coupled system whereby each component is designed to work specifically and exclusively with other components, in the proposed system, things are designed to be loosely coupled. To achieve modularity, it predefines and standardizes interfaces for a general purpose according to the category since things interact with others only through interfaces. Owners or administrators could freely add some new attributes and procedures to their own objects but could not add them to interfaces for other objects. The modularization of the proposed system can be seen in the lower part of Fig. 1. As depicted in the figure, devices situated within a room are organized into modules based on their respective functionalities (M_Relax, M_Temp, and M_Clean). Equipment can be modularized and configured independently of one another, even when they are deployed in the same logical space.

B. Encapsulation

Encapsulation can be used to hide attributes and procedures from view outside of the object for security and privacy. Logical space could prevent access from unauthorized users by hiding all interfaces and adjust access levels for authorized users by hiding partial interfaces according to user profiles. In Fig. 1, the logical space labeled "Rooms" combines several smaller logical spaces, including bedrooms and living rooms, to form a larger logical space referred to as a "House." Additionally, the "House" is incorporated to build the logical space of a "Town," and these logical spaces are finally merged to create the highest-level logical space, which is the "City."

C. Authorization

Authorization can be managed in diverse ways by the owner or administrator of things or spaces. In public facilities, access can give authorization to all users for both things and spaces, while in residential settings, only family members are given access. Offices may have diverse authorization levels, including temporary authorization for guests.

D. Interoperability

The proposed system combines smaller logical spaces (SubLS) to create a larger entity known as the Superordinate Logical Space (SuperLS). SuperLS provides comprehensive information and functionalities, allowing users to interact with various SubLS through SuperLS. For instance, consider an apartment (SuperLS) containing multiple houses (SubLS).

In the event of a fire occurrence within a house, the apartment can gather fire and smoke-related data, trigger fire alarms within each house, guide users on evacuation routes, and issue fire notifications. Conversely, if a fire incident arises within a room of a house, a fire notification can be relayed to the higher-level entity (where the SuperLS of Rooms is the house and the SuperLS of House is the apartment or town).

E. Mobility

Human-centric mobility can be thought of as a shift within a logical space. When a user moves from one logical space to another, it involves the establishment of a new connection, whether explicitly or implicitly, with a new space agent. To ensure seamless and uninterrupted delivery of smart services, the concept of concatenated tunneling can be considered. This allows for the transfer of user profiles, controls, and service profiles to the new agent, in addition to tracking the user's movement trajectories. For instance, when an individual leaves the bedroom and heads to the bathroom for a shower, this transition can be depicted as a move from the bedroom (SubLS) to the bathroom (SubLS). The transmission of this information can be facilitated through a higher-level logical space, known as the House (SuperLS).

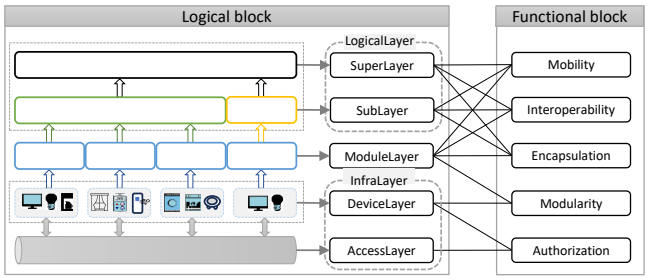

Fig. 2. LSS system architecture based Home

III. LSS System

This section describes our proposed LSS system architecture and the operational mechanism of the system.

A. System architecture

As depicted in Figure 2, the system architecture of LSS comprises two primary components: the functional block and the logical block. The LSS system is integrated into an IoT infrastructure, encompassing smart devices, and is established upon the foundation of the smart device layer (DeviceLayer). The logical blocks can be broadly classified into three principal segments: InfraLayer, ModuleLayer, and LogicalLayer. These logic blocks are closely linked to the functional blocks, which are the basic design principles defined in Section II. Each functional block is connected to the corresponding logical block, providing users with comprehensive and sophisticated information and enabling the expansion of services.

1) InfraLayer: The InfraLayer consists of two key components, namely DeviceLayer, and AccessLayer, each serving distinct roles in the operation of smart devices. The DeviceLayer is responsible for the discovery of devices, the management of device-related information, and the authorization of access control for their interfaces and properties. Each smart device is capable of recognizing its associated agent connected to the cloud system, reporting to the agent, and receiving commands from it. The AccessLayer governs the connections to agents of devices connected to the cloud system through an authentication mechanism.

2) ModuleLayer: The ModuleLayer, on the other hand, offers a method for logically modularizing the devices identified in the InfraLayer based on user requirements. This logical modularization decreases system complexity and minimizes communication overhead. Instead of repeatedly searching for nearby smart devices, users only interact with organized, modular sets of related devices.

3) LogicalLayer: Within the LogicalLayer, the specific location of the user is defined as a SubLayer, which is further composed of multiple ModuleLayers. Additionally, the combination of multiple SubLayers creates a SuperLayer.

B. Operations

The core functionality of the LSS System encompasses logical proximity clustering, which logically organizes nearby smart devices based on the user's current location and required functionalities, as well as device and user access management.

Algorithm 1 Logical proximity clustering

```
1:  function MAKEMODULE(userID, userLoc)
2:      devices ← getSmartDevices(logical_Space)
3:      strModule ← Name of module
4:      listSelModule = [ ]
5:      for device in devices do
6:          if isAuthentication(userID, device) then
7:              if device is selected OR essential device then
8:                  listSelModule.append(device)
9:              end if
10:         end if
11:     end for
12:     NotifyChangedConfiguration()
13: end function

14: function MAKELOGICALSPACE(userID, userLoc, level)
15:     listSpces = [ ]
        // Configuration SubLayer
16:     if level == LEVEL_1 then
17:         if changeSubLayer() then
18:             if isSubLayer(userLoc) then
19:                 if doModify() then
20:                     strSubLayer ← name of SubLayer
21:                     releaseLocSpace(nameOfSubLayer)
22:                 end if
23:             end if
24:             makeModule(userID, userLoc)
25:             makeLogicalSpace(userID,userLoc,LEVEL_2)
26:         else
27:             return NULL
28:         end if
29:     else  // Configuration SuperLayer
30:         strSpace = getNameSuperlayer(userLoc, leve)l
31:         spaceID = getIDSuperlayer(userLoc, level)
32:         listSubLayer =
33:           getSubLayerList(userID, spaceID)
34:         for subLayer in listSubLayer do
35:             if isAuthentication(userID, subLayer) then
36:                 if subLayer is selected then
37:                     listSpces.append(subLayer)
38:                 end if
39:             end if
40:         end for
41:     end if
42:     NotifyChangedConfiguration()
43: end function
```

1) Logical proximity clustering: Logical proximity clustering is implemented through the modularization function, which logically organizes nearby smart devices into one or more modules based on the user's requirements, and through the logical space function, which combines one or more of these modules to create a logical space. The pseudo-code of

the proposed logical proximity clustering method is shown in Algorithm 1.

a) Make module: The makeModule function offers the capability to logically group nearby available devices based on the user's location for modularization. This function takes the user's ID (userID) and location(userLoc) as inputs. It proceeds to search for nearby smart devices using the getSmartDevice function, utilizing the user's location as a reference. The search results are stored in the variable (devices). Subsequently, the user defines the name of the module to be configured in the string variable (strModule). Following this, the isAuthentication function verifies the user's authentication status for each device in the device list. A device can only be added to the module if the user is authenticated for that specific device. Furthermore, if a module necessitates the inclusion of a particular device by default, it is automatically added. Upon completion of the user's selection of smart devices to include in the module, this function automatically notifies the subLayer that the list of configured modules has been updated.

b) Make logical space: The MakeLogicalSpace function, responsible for establishing the logical space, restructures the space based on the user's location and the current level of the logical space. Initially, "LEVEL_1" designates the level where smart devices are interconnected to form the foundational SubLayer. When there's a need to modify the SubLayer configuration, the function verifies the presence of an existing SubLayer and adjusts its actions accordingly—whether it's to amend the SubLayer or add a new one. When configuring a SubLayer, the function calls MakeModule to add modules and build the SubLayer. In cases of modifications, the function first releases the current module configuration before proceeding to construct a new module through the MakeModule function. To configure a SuperLayer by combining SubLayers, you first obtain the list of SubLayers (which can include both SubLayers and SuperLayers). Next, you check user authentication for space access. After confirming user authentication, configure the logical space according to the administrator's selection.

2) Management: When devices are deployed, each device is registered with the device agent's cloud system. Subsequently, user access management for the device is established.

a) User management: User management is employed within the infrastructure and internal systems for credentialing, verification, and authorization in accordance with access control policies for smart devices and logical space access. The user information is transmitted to the service administrator of the corresponding logical space, and access to the respective device and logical space is granted after approval.

b) Device management: When smart devices are deployed, in common commercial practices, they are typically registered under the service provider's account, and personal data is usually transmitted to the provider's cloud (e.g., Amazon or Google ecosystem) [8].

3) Event notification: Our introduced system is structured hierarchically, following a bottom-up approach, and it utilizes notifications to communicate information regarding changes and events. Consequently, when there is a configuration alteration or an event takes place in the lower layer, a notification event is triggered in the upper layer.

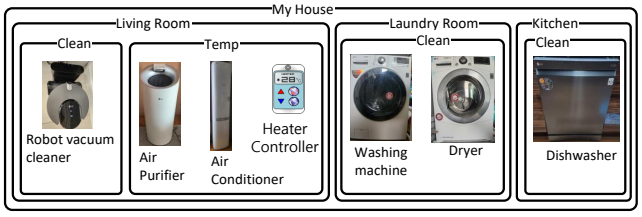

Fig. 3. Experiment Configuration

IV. Proof-of-Concept Prototype

This section first describes the methodology used in the evaluation of our prototype and then discusses the evaluation results. In our experiments, we consider a smart home scenario comprising smart home appliances that are controllable by their respective manufacturers, as shown in Fig. 3. These smart home devices can connect to IPv4 via a WiFi network. The manufacturers of these home devices include Samsung and LG, both of which offer APIs for controlling these devices [9], [10]. Furthermore, individual profiles and lists of smart devices are managed through dedicated vendor applications. Additionally, the discovery process updates the list of smart devices that can currently be configured by the administrator.

A. Scenario

To facilitate this setup, a control server capable of managing the aforementioned home appliances can be implemented on either a PC or a mobile device, and each module can be configured accordingly. Depending on the user's needs, the scenario includes a "Clean" module for cleaning and a "Temp" module for household temperature control, as shown in Fig. 3. The "Clean" module and "Temp" module are enclosed within a SubLayer referred to as "Living Room," and this SubLayer, in turn, is encompassed within a SuperLayer designated as "My House". User1 is granted control rights over these household appliances, thereby gaining access to interfaces for the "Living Room" SubLayer and the SuperLayer named "My House". Essentially, User1 has to notify the SubLayer and SuperLayer about events related to the configured home appliances.

As previously outlined, when establishing a scenario based on the Algorithm 1, the initial step involves configuring the "Clean" and "Temp" modules using the MakeModule function. Employing the MakeLogicalSpace function, we then establish a SubLayer called "Living Room," which encompasses the configured "Clean" and "Temp" modules. Subsequently, we create a SuperLayer referred to as "My House" and integrate the SubLayers of the living room, laundry room, and kitchen, into this superLayer. As an example, we will describe the process of setting up the "Temp" module when User1 is presently located in their living room. When User1 configures the "Temp" module in this location and includes household appliances such as an air purifier, air conditioner, and heater controller in the module, any changes made to the module

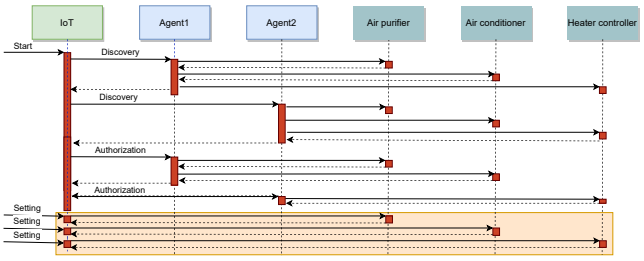

Fig. 4. Sequence diagram of legacy manners

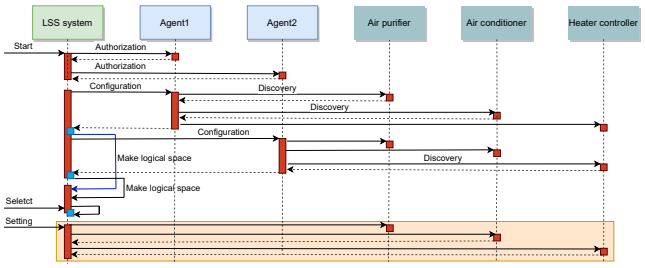

Fig. 5. Sequence diagram of LSS system

TABLE I
COMPARISON OF LEGACY MANNERS AND LSS SYSTEM

System	Action information	Click	Time
LSS System	Input User Authentication	1	t_{ua}
	Discovering and Authorizing through Vendor Services	1	t_{da}
	Make logical space and module	nD + nL	$t_l \times (nD + nL)$
	Set	1	t_s
Legacy Manners	Authorizing through Vendor Services	nV	$t_a \times nV$
	Discovering and configuration through Vendor Services	nV + nD	$t_{dc} \times (nV + nD)$
	Set	nD	$t_s \times nD$

Legend:nD-Num of Devices, nV-Num of Venders, nL-Num of Logical space

are notified to the SubLayer, which is the living room. The SubLayer subsequently forwards this event to the SuperLayer that represents "My House".

B. Experiment Results

As depicted in Fig. 5, when controlling devices within the legacy manners framework, one must initially conduct a search for devices across various access technologies within the cloud. Subsequently, connections must be established to each device via their respective access technology once a sensor is discovered. Each device can be managed through authentication by an agent designated by the device manufacturer. Any necessary device configurations must be executed separately for each device. On the other hand, in the LSS system illustrated in Fig. 4, authentication of the agent is the first step. Following this, the device is automatically detected through different access technologies. Users can then configure the module for the discovered device, and the SubLayer and SuperLayer are configured for the logical space authenticated to the user. Consequently, device settings can be accomplished in a single operation, eliminating the need for individual configurations for each device.

The primary difference between traditional methods and LSS system is their capacity to establish reusable, logical module configurations. In the scenario involving the setup of three smart devices related to temperature within a smart home service, the legacy system requires three times as many setting steps as the LSS system. As the number of smart devices increases, the legacy method's setting steps also increase in direct proportion to the number of devices that need setting.

Table I displays the count and timing of user interactions between the legacy method and the LSS system based on the aforementioned sequence diagram. In a scenario featuring 3 devices and 2 agents, where each action consumes 1*ms*, the total click count for the traditional method amounts to 10. On the other hand, the LSS system entails 8 clicks, encompassing the creation of the logical space (1 sublayer and 1 superlayer), making it more simple. In terms of setup, the legacy method demands as many actions as there are devices, while the LSS system requires only one interaction. When we compare the time taken, the LSS system completes the task in 6*ms*, whereas the legacy approach takes 8*ms*. This indicates that the LSS system offers a more streamlined and quicker setup process.

V. CONCLUSION

This article proposes a novel smart device management paradigm considering the end-user perspective. The LSS system efficiently divides the user's space into logical spaces taking into account spatiotemporal changes and user preferences. The experimental results demonstrate how the LSS system can improve the performance of IoT-based service operations.

Furthermore, future work will involve the creation of larger-scale testbeds incorporating state-of-the-art IoT communication standards and a diverse range of proximal objects.

REFERENCES

[1] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, oct 2019.

[2] P. Bellini, P. Nesi, and G. Pantaleo, "Iot-enabled smart cities: A review of concepts, frameworks and key technologies," *Applied Sciences*, vol. 12, no. 3, 2022.

[3] A. Solanki and A. Nayyar, "Green internet of things (g-iot): Ict technologies, principles, applications, projects, and challenges," in *Handbook of research on big data and the IoT*. IGI Global, 2019, pp. 379–405.

[4] C. Stolojescu-Crisan, C. Crisan, and B.-P. Butunoi, "An iot-based smart home automation system," *Sensors*, vol. 21, no. 11, p. 3784, 2021.

[5] K. Fizza, A. Banerjee, K. Mitra, P. P. Jayaraman, R. Ranjan, P. Patel, and D. Georgakopoulos, "Qoe in iot: a vision, survey and future directions," *Discover Internet of Things*, vol. 1, pp. 1–14, 2021.

[6] C. G. García, L. Zhao, and V. García-Díaz, "A user-oriented language for specifying interconnections between heterogeneous objects in the internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3806–3819, 2019.

[7] M. Yu, "Network telemetry: Towards a top-down approach," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, p. 11–17, feb 2019.

[8] C. Meurisch, B. Bayrak, and M. Muhlhauser, "Edgebox: Confidential ad-hoc personalization of nearby iot applications," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[9] Samsung, "SmartThings," https://developer.samsung.com/smartthings, 2023, [Online; Samsung Developers].

[10] LG, "LG ThinQ Platform," https://thinq.developer.lge.com/ko/cloud/, 2023, [Online; ThinQ Connect Service].