

Design and Implementation of Autonomous Collaboration System of Smart Things using accumulated Experience knowledge

Chang-Su LEE*, Sweung-Won CHEUNG**, Seong-Soon JOO***, Hyun-Kook KAHNG*

*Department of Electronics & Information Engineering, Korea University, Korea

**Hanwha Techwin Co, Ltd., Korea

***ETRI (Electronics and Telecommunications Research Institute), Korea

lcs3708@korea.ac.kr, kahng@korea.ac.kr, ssjoo@etri.re.kr, sw.cheung@hanwha.com

Abstract— With the advent of many IoTs, users need a common use environment to easily connect, search, and share various IoTs. In addition, as IoTs become increasingly intelligent, there is a growing demand for services that work together. However, a large number of IoTs can cause serious traffic to the network when they collaborate with each other in a common usage environment. In this paper, we have designed the structure of IoT to construct a distributed social network based on semantic web to solve this problem. We also designed and implemented an autonomous collaborative service of IoT that can find collaborative partners through social network and accumulate experience knowledge after collaboration. We defined IoT with this structure as Smart Thing and a unique Basic Identity Information (BII) that allows Smart Thing to identify each other. In addition, we defined Basic Response Rules (DRRFS) for Smart Thing to follow in the Query & Answer process for Collaboration. Smart Thing with basic identity information is an independent, autonomous IoT device. Smart Thing with a distributed social network finds collaborative partners through recursive queries and provides services. In addition, we have confirmed in the autonomous collaborative system that Smart Thing who has experienced collaborations can work faster and more accurately using the accumulated experience knowledge when the same thing is done again. If all IoTs have basic identity information and use semantic technology, the Internet will be a huge space where Smart Thing can collaborate.

Keywords— Autonomous Collaboration System, Semantic Web, WebID, Distributed social network, IoT device

I. INTRODUCTION

As many IoTs come into existence, users need a common usage environment, which easily connects, searches, and shares various IoT's.[1] In addition, as IoTs become increasingly intelligent, there is a growing demand for services that work together. Figure 1 shows that IoTs create collaborative groups for tasks and accumulate their collaborative experience knowledge. IoT 1, 2, 6, and 7 are group to complete task A. IoT 3, 4, and 10 were grouped for task B, and IoT 5, 8, 9, and 11 were grouped for task C. IoTs accumulate experience knowledge after completion of work. However, many IoTs collectively collaborate in a common

environment, causing a lot of data traffic to the network. To solve this problem, we designed the structure of IoT, which constitutes a distributed social network based on Semantic Web. We also designed and implemented an autonomous collaborative service of IoT that can find collaborative partners through social network and accumulate experience knowledge after collaboration. We defined the IoT device with this structure as Smart Thing. In this paper, we use the Web as an environment for autonomous collaboration of Smart Things. In addition, Smart Thing expands its social network with accumulated experience.

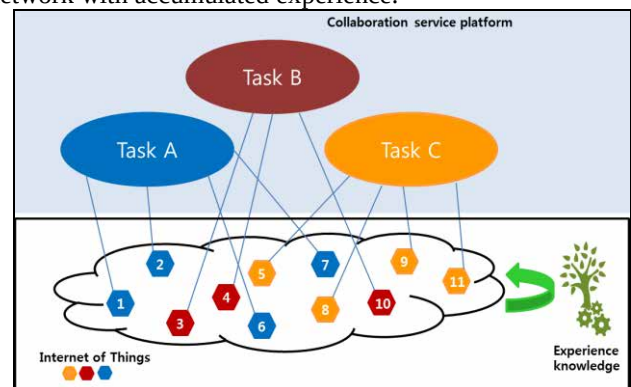


Figure 1 IoTs create collaboration groups for tasks

In this design, all Smart Things have Basic Identify Information (BII). BII is WebID, Web server address, and Endpoint for SPARQL. Smart Things identify each other's basic identity information through an Identification Process (IDP). We discussed BII in detail in Section III. Originally, WebID is a technology that defines user's data as ontology and notifies the location of defined data through URL. It is defined for identification of users in a global distributed social web. We use this technology to allow Smart Things to identify each other. Smart Thing also has Apache Jena Fuseki. Fuseki provides SPARQL and Triple Database (TDB) to Smart Thing. In addition, Smart Things accumulate experience knowledge to TDB after collaboration.

Note 1: Solid is a new project led by Prof. Tim Berners-Lee, inventor of the World Wide Web. It provides a set of rules and tools that we use in this paper

In this paper, using these technologies, we implemented one case of autonomous collaboration systems: a mailing and delivery collaboration system among drones as a specific task of autonomous collaboration. A delivery drone takes the mail to the recipient's company and finds an in-house drone that can deliver the mail to the recipient, and then delivers the mail to the in-house drones. In-house drones are responsible for in-house delivery.

All Smart Things in the system were implemented in PCs and assumed that physical movement and mailing were replaced by separate applications using GPS and mechanical devices. In section II, we describe WebID, Solid server (WebID server), SPARQL and Apache Jena Fuseki in the background of the related technology. And in section III we show an example model including definitions and procedures of the structure of Smart Thing, the Basic Identity Information (BII), Identification process (IDP), and Default Response Rule for SPARQL (DRRFS). In section IV, we describe the implementation in detail, and finally present conclusions and future work in section V.

II. TECHNOLOGY TRENDS AND BACKGROUND

A. WebID

WebID is an HTTP URI referring to an agent (an individual, an organization, a group, a device, and the like) with a technology defined according to a standard proposed by the W3C. WebID provides a globally unique identifier by URI. [2] The WebID profile is a document that describes the owner of the WebID in the RDF data format. The data in this document is defined through an ontology that describes each relationship. Ontology consists of a URI, which provides a globally unique identifier for the defined data, and its definition can represent the service. Figure 2 shows the relationship between WebID URI and WebID Profile document.

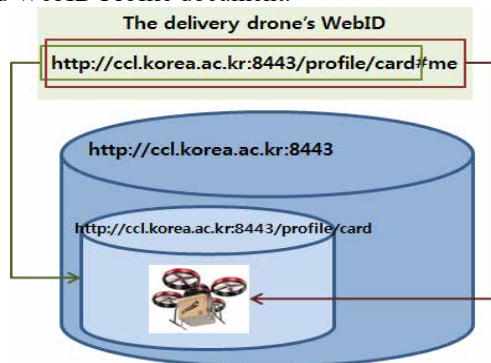


Figure 2 Relationship between WebID and WebID profile

The WebID URI is created by adding the hostname of the WebID Server to the HTTPS protocol and appending the location of the profile document. The reason for using the HTTPS protocol is that WebID authentication and security are performed using TLS. In the WebID URI, the location of the profile document can be specified using a hashtag. Figure 2 shows the WebID URI using the WebID of the delivery drones as an example. <http://ccl.korea.ac.kr:8443/profile/card>,

which is a WebID profile URI that does not include hash tag #me, describes the owner's profile document. That is, if <me> is included, it indicates information describing the WebID owner, and if not, it indicates information describing the WebID profile document. Agents and applications use WebIDs to authenticate WebIDs to Web sites that support the WebID protocol and act on behalf of the owner. The WebID protocol integrates authentication and trust into the concept of WebID. The basic concept of the WebID protocol is to connect the SSL client certificate to the WebID profile so that the WebID owner can authenticate against other Web sites that support the WebID protocol. WebID authentication and security uses TLS, and this protocol is called WebID-TLS protocol. [3]

B. WebID Server (Solid Server)

Solid Server is a server program provided by Solid¹. It has functions of WebID creation and management. [4] In this paper, we use a Solid Server for the WebID server that generates and manages WebID and profile in the mailing and delivery collaboration system among drones. Solid Server has Single-mode and Multi-mode. Single-mode, used in this paper, creates a single WebID and profile on the server, thus indicating the location of the profile in the domain of the Server. Single-mode also requires the use of SSL certificates because Server can authenticate WebID.

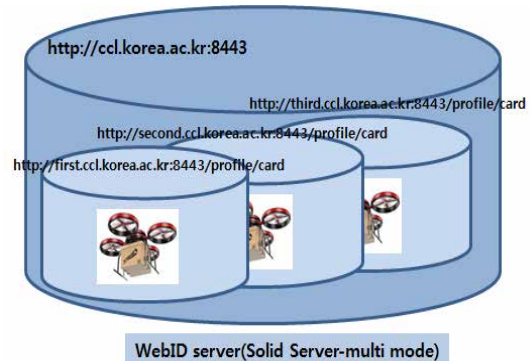


Figure 3 Multi-mode in Solid Server

WebID creation and management uses the LDP (Linked Data Platform) standard. The LDP standard defines a set of rules for HTTP operations on RDF-based Web resources to provide architecture for reading and writing linked data on the Web. [5] WebID profile can be created and modified using the HTTP method defined by LDP in the WebID URI.

C. SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is a query language for processing structured data in RDF format or RDF format and has been recommended by W3C. [6] The main purpose of SPARQL is to provide a formal language for expressing questions with semantic information. [7] SPARQL Endpoint is also used. SPARQL Endpoint is a Web-accessible query service that provides the ability to process SPARQL query language. Normally, when an HTTP GET request is sent to a SPARQL Endpoint, the server that received the

Note 1: Solid is a new project led by Prof. Tim Berners-Lee, inventor of the World Wide Web. It provides a set of rules and tools that we use in this paper

Table 1 BII (Basic Identity Information)

BII	Web server address
	WebID
	Endpoint

6) As same token Smart Thing B logs in to the Web server, which confirmed in BII of Smart Thing A. 7) Smart Thing A sends Smart Thing B's WebID to its Fuseki. 8) Fuseki in Smart Thing A do SPARQL to the WebID profile of Smart Thing B. 9) LOAD the profile of Smart Thing B. 10) Smart Thing A stores it in TDB <Temporary knowledge>. We defined the process from (1) to (10) as IDP as a procedure in which two Smart Things identify each other. Every Smart Thing always identifies each other through an IDP. 11) After the IDP, Smart Thing A does SPARQL to the endpoint of Smart Thing B for collaboration. The endpoint is the TDB <Work knowledge> address of Smart Thing. We designed the TDB with three directories, as shown in Table 2.

Table 2 TDB structure in Smart Thing

TDB & directory	Role
TDB <Temporary knowledge> (ex: http://ccl.kora.ac.kr:3030/temp)	Storage Temporary data
TDB <Work knowledge> (ex: http://ccl.kora.ac.kr:3030/ds)	Storage Work knowledge data
TDB <Experience knowledge> (ex: http://ccl.kora.ac.kr:3030/sn)	Storage Experience knowledge data

The TDB <Temporary knowledge> of the delivery drones in the mailing and delivery collaboration system among drones is <http://ccl.korea.ac.kr:3030/temp>, a place to store data temporarily. TDB<Work knowledge> is <http://ccl.korea.ac.kr:3030/ds>, a place to store working data. In addition, TDB <Experience knowledge> is <http://ccl.korea.ac.kr:3030/ds>, a place to store experience knowledge data after collaboration. In this system, the delivery drone has two query statements for collaboration. The first is 'Can you carry 0.8kg of mail?' The second is 'Can you deliver mail to Chang-Su Lee?' That is, the delivery drones are looking for in-house drones that can deliver 0.8 kg of mail to Chang-Su Lee. 12) And confirm the answer of Smart Thing B. We assume that Smart Thing A's TDB-Work knowledge has mailing information, and Smart Thing B's TDB-Work knowledge has Smart Thing B's limit weight and some employee's information. If the answer is yes, B output the Web server address. 13) Smart Thing A enters its endpoint into Smart Thing B's Web server. 14) Web server sends the endpoint to Fuseki. 15) 16) Smart Thing B do LOAD the mail information at the endpoint of Smart Thing A. 17) Smart Thing B save it in TDB <Temporary knowledge>. 18) Smart Thing A moves and saves the Smart Thing B's profile stored in TDB<Temporary knowledge> to TDB<Experience knowledge>. Smart Thing B also saves the profile of Smart Thing A that was stored in TDB<Temporary knowledge> to TDB<Experience knowledge>. At 12) if Smart Thing B's answer is 'No', Smart Thing A finds the other Smart Thing again and starts again from 1). The information stored in Smart Thing's TDB <Experience knowledge> is

collaborative experience knowledge and it is social network information.

If Smart Thing A is going to do the same thing again after completing the collaboration, it will complete the collaboration faster than the first by SPARQL directly to Smart Thing B's Endpoint, which is accumulated in the experience knowledge.

C. Default Response Rules for SPARQL (DRRFS)

Smart Thing follows the Default Response Rules for SPARQL (DRRFS) in the Query & Answer process for collaboration (see Table 3).

Table 3 Default Response Rules for SPARQL (DRRFS)

DRRFS 1 – do SPARQL	
Smart Thing will simultaneously SPARQL the endpoints of all Smart Things registered in TDB <Temporary knowledge> or TDB <Experience knowledge>.	
1	After IDP,
2	do (query) {
3	SPARQL to the Endpoint which is registered in TDB<Experience knowledge> or TDB<Temporary knowledge>
4	}
DRRFS 2 - Structure of Questions	
At the end of the question, attach the endpoint address of all Smart Thing receiving the question.	
1	Question+ ' http://gate1.korea.ac.kr:3030/ds/query '
DRRFS 3 - Structure of Answer	
1) If the answer is No, Smart Thing will answer 'undefined'. If the answer is yes, Smart Thing answers with his web address.	
DRRFS 4 - If answer is No	
1) Smart Thing, whose answer is No, asks the same question to the endpoints of all social networks registered in TDB <Experience Knowledge> before answering. The structure of the question is as shown in DRRFS 2.	
1	Question+ ' http://gate1.korea.ac.kr:3030/ds/query ' + ' http://gate1.korea.ac.kr:3030/ds/query ' + ' http://dd3.korea.ac.kr:3030/ds/query '
2) If any of the added endpoint addresses have duplicate addresses, do not ask questions to the duplicate Smart Thing. 3) If the answer of all Smart Things is No, output 'No'.	
1	if (result value of SPARQL == 'undefined') {
2	do (query) {
3	all Smart Things which are in TDB-Experience knowledge
4	if (result value of SPARQL == 'undefined') {
5	message == 'No'
6	}
7	}
DRRFS 5 - If answer is Yes	
1) If the answer is yes, Smart Thing answers with his Web address.	
2) Smart Thing, whose answer is No, asks the same question to the endpoints of all social networks registered in TDB <Experience Knowledge> before answering.	
- If Smart Thing A receives a web server address from Smart Thing B registered in TDB<Experience knowledge>, Smart Thing A attaches its web server address after B's web server address and output it. (ex. "B's web server address + A's web server address")	
- If Smart Thing A received web server address from one or more Smart Things registered in TDB<Experience knowledge>, select	

Note 1: Solid is a new project led by Prof. Tim Berners-Lee, inventor of the World Wide Web. It provides a set of rules and tools that we use in this paper

only a Web server address of Smart Thing, which answered first.	
1	if (result value of SPARQL == 'undefine') {
2	do (query) {
3	all Smart Things which are registered of social network
4	if (result value of SPARQL == 'undefine') {
5	message == 'No'
6	} else {
7	Select the Web server address which
8	answered first
9	Message==Web server address
10	}
11	}
12	}
DRRFS 6 – Collaboration & Finish	
After the question, Smart Thing, which receives the web server address, confirms that its TDB has the same web server address.	
1) If it is, log in as an Endpoint at that address and collaborate.	
2) If not, proceed to 1) after the IDP procedure.	
1	if (result value == 'undefine') {
2	finish the SPARQL process
3	} else if {
4	select first web server address
5	fws address == first web server address
6	if (fws address is in TDB-Experience knowledge or
7	TDB-Temporary knowledge){
8	logs in to the fws address with Endpoint
9	} else {
10	logs in to the fws address with WebID
11	}
12	}
13	}

IV. IMPLEMENTATION

A. The mailing and delivery collaboration system among drones

Table 4 shows the details of Smart Things in the mailing and delivery collaboration system among drones. It describes the BII and TDB information of each Smart Things.

Table 4 BII and TDB information of Smart Thing in the System

IoT	BII	TDB information
Gate1	<https://gate1.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://gate1.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://gate1.korea.ac.kr:3000>.	TDB-Experience knowledge 1) <https://dd1.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd1.korea.ac.kr:3030/ds/query>; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd1.korea.ac.kr:3000>. 2) <https://dd2.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd2.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd2.korea.ac.kr:3000>. TDB-Work knowledge 1) Dong-Hee Kim/010 3077 4727/Gate1 2) U-Suk Ji/010 2236 2531/Gate3 3) Seung-Won Jung/010 2345 2345/Gate1
Delivery drone	<https://ccl.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint>	TDB-Work knowledge Delivery information : Chang-Su Lee/010 3537 3708 Digital Camera/0.8kg/

	oint> <http://ccl.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://ccl.korea.ac.kr:3000>.	
In-house drone #1	<https://dd1.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd1.korea.ac.kr:3030/ds/query>; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd1.korea.ac.kr:3000>.	TDB-Experience knowledge 1) <https://gate1.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://gate1.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://gate1.korea.ac.kr:3000>. TDB-Work knowledge 1) Dong-Hee Kim/010 3077 4727/Gate1 2) Chang-Su Lee/010 3537 3708/Gate3 less than 0.7kg
In-house drone #2	<https://dd2.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd2.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd2.korea.ac.kr:3000>.	TDB-Experience knowledge 1) <https://dd3.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd3.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd3.korea.ac.kr:3000> . 2) <https://gate1.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://gate1.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://gate1.korea.ac.kr:3000>. TDB-Work knowledge 1) Dong-Hee Kim/010 3077 4727/Gate1 2) Nak-Hyun Kim/ 010 3030 5904/Gate1 less than 0.5kg
In-house drone #3	<https://dd3.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd3.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd3.korea.ac.kr:3000>.	TDB-Experience knowledge 1) <https://dd2.korea.ac.kr/profile/card#me> <http://sac.korea.ac.kr:3003/ontology#Endpoint> <http://dd2.korea.ac.kr:3030/ds/query> ; <http://sac.korea.ac.kr:3003/ontology#Server> <http://dd2.korea.ac.kr:3000>. TDB-Work knowledge 1) Chang-Su Lee/010 3537 3708/Gate3 2) U-Suk Ji/010 2236 2531/Gate3 less than 1kg

In this system, the delivery drones are looking for in-house drones capable of delivering 0.8 kg of mail to Chang-Su Lee. According to Table 4, the delivery drones must find in-house drones # 3 and pass the mail information.

B. Delivery drone vs. In-house drone #3

The delivery drone that brought the mail to the recipient's company will hand over the mail to the in-house drone #3. A delivery drone knows the location of the recipient's company and information about the in-house drone #3 Web Servers before departure. Collaboration configuration is shown in Figure 7.

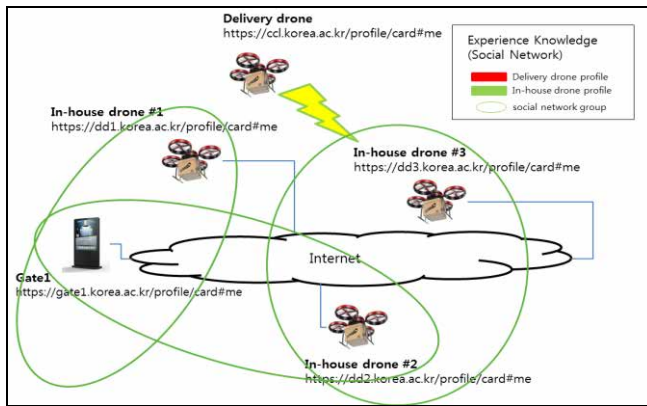


Figure 7 Collaborative diagram of delivery drone vs. in-house drone #3

- 1) The delivery drones arriving at the company log in to Web Server of in-house drones #3 with WebID. Then do IDP.
- 2) In-house drone #3's fuseki checks the profile of the delivery drone (see Table 5) and loads it.

Table 5 Check the WebID of the smart thing logged into the web server

1	function Devicecheck (WebID) {
2	username = WebID;
3	var url = WebID;
4	if (url.length > 0){
5	var http = new XMLHttpRequest();
6	var params = "?url="+url;
7	http.open('HEAD', url);
8	http.withCredentials = true;
9	http.onreadystatechange = function(){
10	if (this.readyState == this.DONE){
11	if (this.status === 0){
12	console.log('could not connect to server.');
13	} else if (this.status === 200){
14	var chatpage = chatroom20;
15	// Confirm the WebID with the HTTP HEAD method.
16	// If it is an existing WebID, it returns HTTP
17	// STATUS CODE 200. In this case, you log in and
18	// pass the profile

- 3) In-house drones #3 check the profile through the Web ID of the delivery drones and login to the delivery drone's web server.
- 4) The delivery drones get the profile of in-house drones #3 (see Table 6).

Table 6 LOAD a profile

1	var UpdateDroneData = function(data){
2	var webid = data.webid
3	var url = 'http://localhost:3030/temp/update'
4	var query = "LOAD <"+webid+">";
5	var http = new XMLHttpRequest();
6	http.open('POST', url);
7	http.setRequestHeader('Content-Type',
8	'application/sparql-update');
9	http.withCredentials = true;
10	http.onreadystatechange = function() {
11	if (this.readyState == this.DONE){
12	if (this.status === 204){
13	// Update the profile to Fuseki using the LOAD
14	// statement on the endpoint 'http://localhost:3030 // ds /
15	update'. If the update succeeds, it returns HTTP // STATUS
16	CODE 204.

The delivery drones doing SPARQL to the endpoint of the in-house drones #3 (see Table 7). The questions are "Can you carry 0.8 kg of mail?", "Can you deliver mail to Chang-Su Lee? All Q & A (Query and Answer) follow DRRFS in Table 4.

Table 7 SPARQL to Endpoint

1	var query1 = 'SELECT ?Device ?Server WHERE { ?x
	http://sac.korea.ac.kr:3000/ontology/ccl#Name>
	""+DronReceiver+"";
	<http://sac.korea.ac.kr:3000/ontology/ccl#haveTel>
	""+DronReceiverHP+"" ?Device
	<http://sac.korea.ac.kr:3000/ontology/ccl#MaxWeight> ?W
	eight. FILTER(''+DronWeight+'<=?Weight)'
2	var url1 = DeviceEndpoint+'?query=' +
	encodeURIComponent(query1);
3	var http1 = new XMLHttpRequest();
4	http1.open('GET', url1);
5	http1.withCredentials = true;
6	http1.onreadystatechange = function(){
7	if (this.readyState == this.DONE){
8	if (this.status === 200){
9	// Query the Endpoint to verify the DronReceiver,
10	// DronReceiverHP, and DronWeight.

- 5) Confirm the answers of in-house drones #3 (see Table 8).

Table 8 Answers to queries

1	var url = Server+'?query=query=' +
	encodeURIComponent(query);
2	var http = new XMLHttpRequest();
3	http.open('GET', url);
4	http.withCredentials = true;
5	http.onreadystatechange = function(){
6	if (this.readyState == this.DONE){
7	if (this.status === 200){
8	var result = http.responseText;
9	var resultJSON = JSON.parse(result)
10	if(typeof(resultJSON) === 'undefined'){
11	console.log('cannot find data')
12	} else {
13	var answer = 'YES'
14	message = {recipient:resultJSON.Server,
	CollaborateServer:resultJSON.Server,
	CollaborateDevice:resultJSON.Device}
15	socket.emit('collaborate worker', message);
16	}
17	//If the result of the query is 'Yes', the function
18	//collaborateDeviceSave1 is executed to deliver the //Web
19	Server address to the delivery drones

- 6) The delivery drone hands the mail to the in-house drones #3, which answer with web server address. In this paper, it is assumed that the physical movement of each Smart Thing and the mail transfer are replaced by separate application using GPS and mechanical device, so that in-house drone #3 is replaced by importing the mail information of the delivery drone.

- 7) The delivery drone adds BII of the in-house drone #3 to TDB-Experience knowledge. In-house drone #3 also adds the BII of the delivery drone to his TDB-Experience knowledge.

C. Accumulation of Experience knowledge (expansion of social network)

Note 1: Solid is a new project led by Prof. Tim Berners-Lee, inventor of the World Wide Web. It provides a set of rules and tools that we use in this paper

Figure 8 shows the social network of delivery drone and in-house drones # 3 expanded after collaboration.

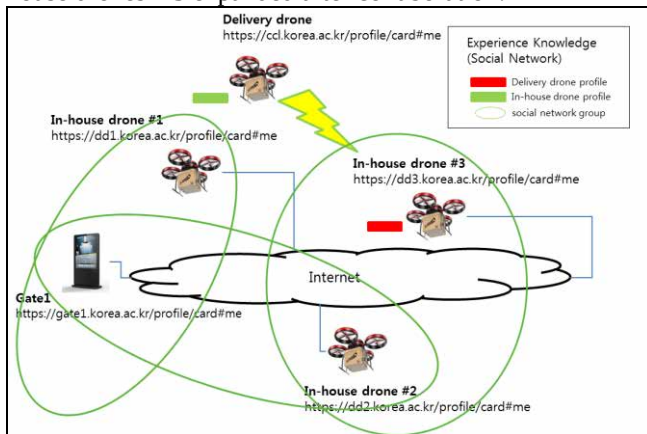


Figure 8 Expanding social network after collaboration

The delivery drone adds the BII (green) of the in-house drone #3 to his experience knowledge, and the in-house drone #3 add the BII (red) of the delivery drone to his experience knowledge. Adding BII of collaboration partners is an expansion of the social network of Smart Thing as the number of social member's increases. Therefore, 'accumulation of experience knowledge' can be defined as 'expansion of social network'. By accumulating experience knowledge after collaboration, Smart Thing can handle the same thing faster and more accurately in the future.

D. Delivery drone vs. Gate1

It is a collaborative scenario in which a delivery drone who brings mail to the recipient's company hand over the mail to gate1. The delivery drones know the location of the recipient's company and the web server information on gate1 before departure. The collaboration configuration is shown in Fig 9. The delivery drones arriving at the company log into the web server of gate1. Then those do the IDP.

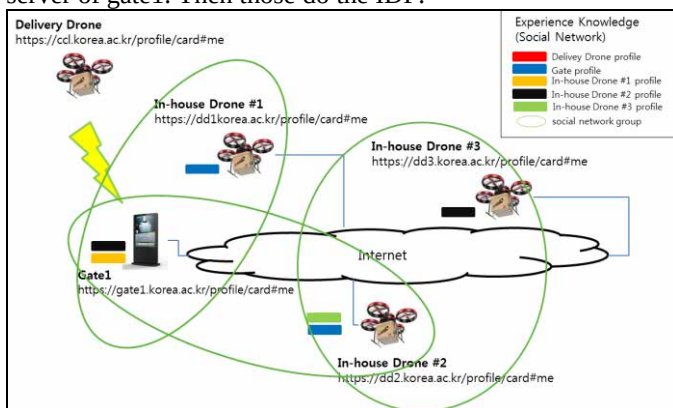


Figure 9 Collaboration diagram of delivery drone and gate1

1) The delivery drones doing SPARQL to the endpoint of the Gate1 (see Figure 10). The questions are "Can you carry 0.8 kg of mail?", "Can you deliver mail to Chang-Su Lee? All

Q&A follow DRRFS in Table 3. The progress of SPARQL is shown in Figure 10.

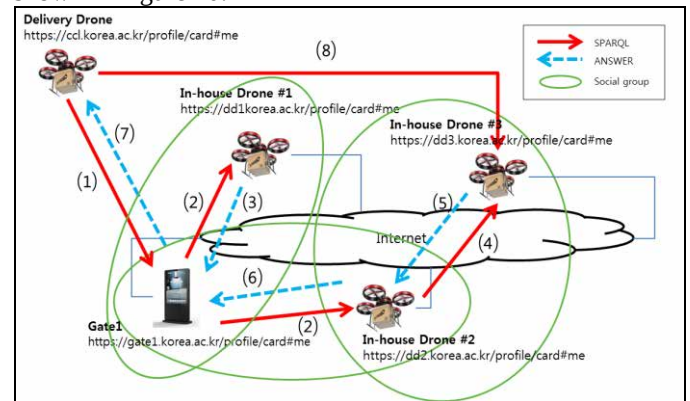


Figure 10 Answer flowchart based on DRRFS

2) Gate1 do SPARQL to the in-house drones #1 and #2 registered in its social network according to <DRRFS-4> before responding 'No'
 3) In-house drone #1 responds with 'No' to Gate1 according to <DRRFS-4>
 4) In-house drones #2 do equally SPARQL to in-house drones #3 according to <DRRFS-4>
 5) In-house drone #3 responds with the web server address 'http://dd3.korea.ac.kr: 3000' to in-house drone # 2 according to <DRRFS-5>
 6) In-house drone #2 responds with 'http://dd3.korea.ac.kr + http://dd2.korea.ac.kr' to gate1 according to <DRRFS-5>
 7) And gate1 responds with 'http://dd3.korea.ac.kr +http://dd2.korea.ac.kr +http://ccl.korea.ac.kr' to the delivery drones.
 8) The delivery drone log in to the web server address of in-house drone #3 which is confirmed by the answer of gate1 according to <DRRFS-6>. Then those do the IDP. The delivery drones do SPARQL to the endpoint of the in-house drones #3, and check the responds 'http://dd3.korea.ac.kr: 3000'. Since the delivery drones have the same web server address in TDB, log in to the web server of in-house drones # 3 as an endpoint. In-house drone #3 LOADs the mail information from the delivery drone and stores it in TDB-Temporary knowledge.

As shown in Figure 11, the delivery drone adds BII of the in-house drone #3 to TDB-Experience knowledge. In-house drone #3 also adds the BII of the delivery drone to his TDB-Experience knowledge.

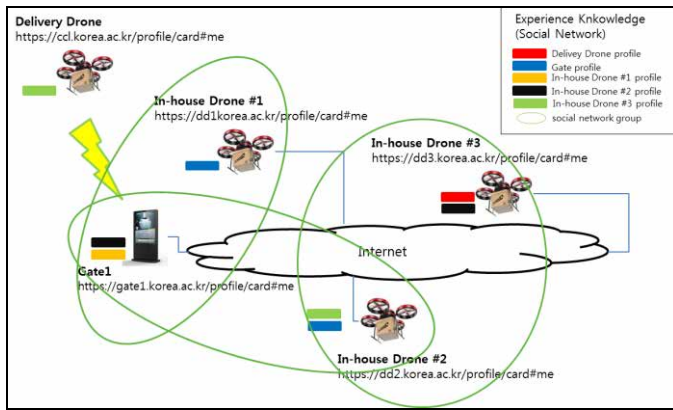


Figure 11 Expanding social networks after collaboration

E. If the delivery drone doesn't know the web server address to log in to

If the delivery drones that bring mail to the recipient's company do not know the address of the web server to log in to. The configuration is shown in Figure 12.

- 1) Smart Things keep informing their web server information to around. In this paper, we use the advertisement function of wireless router.
- 2) When the delivery drone is connected to the open Wi-Fi, it is automatically connected to the Web server of the Smart Thing, which is designated as the notification / advertisement site (see Table 9).

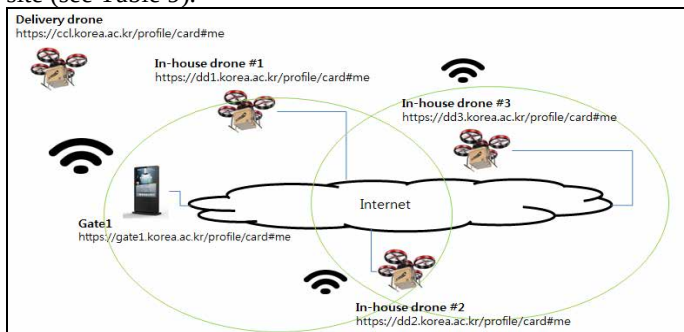


Figure 12 Log in to the Smart Thing web server in the vicinity

- 3) If the delivery drone connects to the Web server, login with WebID. And do IDP.
- 4) The delivery drones look for in-house drones # 3 according to DRRFS.

Table 9 Logging in to the Smart Thing's web server in the vicinity

1	function autologin(checkdata){
2	var url = "http://localhost:3000/autologin";
3	var http = new XMLHttpRequest();
4	http.open("GET", url);
5	http.withCredentials = true;
6	http.onreadystatechange = function(){
7	if (this.readyState == this.DONE){
8	if (this.status === 200){
9	var data = http.responseText;
10	var chaturl = checkdata.Server + '?url=' + data;
11	location.href = chaturl;
12	}else {
13	console.log("login Fail")

14	//The autologin function asks the WebID to Smart Thing to
15	//log in. When the WebID is returned, add the WebID to the
16	//domain and log in. The format is 'protocol + domain +? Url
17	//+ WebID'.

V. CONCLUSIONS

We have confirmed in the mailing and delivery collaboration system among drones that Smart Thing can provide autonomous collaboration services using social networks. We confirmed that gate1, which was requested by the delivery drones, used social network to find in-house drones # 3 and added BII to each other after collaboration. We also confirmed that when a delivery drone who has accumulated experience knowledge is asked to do the same task again, he can use the experience knowledge to deliver mail faster and more accurately than the first. Also, when Smart Things follow the DRRSF for collaboration, we have confirmed that the searching and Q & A procedures follow recursive querying. We also confirmed that when Smart Thing follows the DRRSF for collaboration, the Q & A process follows a recursive query. If so many Smart Things provide autonomous collaboration services using distributed social networks, the Web will be able to maintain a more stable network environment

In the future, we will define various tasks as ontologies and make application services where hundreds or thousands of smart things collaborate concurrently through hundreds of social networks. Therefore, we will comprehensively analyse the performance of the social network based on the semantic Web and the autonomous collaboration of Smart Thing using experience knowledge.

ACKNOWLEDGMENT

This work was supported by the project "Design and validation of architecture of Context-based collaborative smart network for heterogeneous things" funded by Electronics and Telecommunications Research Institute (ETRI) (No. 5311-2017-00010).

REFERENCES

- [1] Vlad Trifa, Dominique Guinard, David Carrera, Web Thing Model, 30 December 2017
- [2] Tim Berners-Lee, Andrei Samba, Henry Story, <https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html>
- [3] Toby Inkster, Henry Story, Bruno Harbulot, <https://www.w3.org/2005/Incubator/webid/spec/tls/>
- [4] Tim Berners-Lee, Christian Smith, Sarven Capadisli, Dmitri Zagidulin, Nicola Greco, Ruben Verborgh, <https://github.com/solid/solid>
- [5] Steve Speicher, John Arwe, Ashok Malhotra, Linked Data Platform 1.0, 26 February 2015
- [6] Bob DuCharme, Learning SPARQL, Second Edition, June 2013, ISBN 978-1-449-37143-2
- [7] David Wood, Marsha Zaidman, Luke Ruth, Michael Hausenblas, Linked Data Structured data on the Web, 2017,3, ISBN 979-11-950726-4-4
- [8] Apache Jena, <https://jena.apache.org/documentation/fuseki2/>



Chang-Su Lee was born in Republic of Korea at Sep. 1969. He is a Ph.D graduate student in Department of Electronics and Information Engineering at Korea University, Korea. His Major is ICT Convergence Technology.

He received the Bachelor's degree in Information & Engineering from Korea University in 1995, and received Master's degree from same University in 1999.

From 1995 to 1997, he worked in the field of video conferencing system at LG Information & Communication Co., Ltd. From 1999 to 2015, he worked in the field of CCTV and video security system at POSDATA Co., Ltd and RIFA Co., Ltd. His research interest includes, IoT, autonomous collaboration system, and architecture & protocol for the networking of everything (NOE).



Dr. Sweung-won Cheung is a principle research engineer in Hanwha Techwin co. Pankyo city, Korea.

He received bachelor's degree in Information Engineering from Korea University in 1994, He received master's degrees in Information Engineering from Korea University in 1996, in Electrical Engineering from University of Colorado in 1999, in Electrical and Computer Engineering from University of Florida in 2002 respectively. And he received Ph.D. in Electrical and Computer Engineering from

University of Florida in 2009.

From 2010 to present, he has been working in Hanwha Techwin co. Major interest is Web software research and development, especially multimedia streaming and processing. Recently research interest is Deep Learning in the Browser



Seong-Soon Joo received the B.S. from Hanyang University in 1980 and received the M.S. and the Ph.D. degree from Seoul National University, Korea, in 1982 and 1989 respectively, all in electrical engineering.

He joined in 1983, where he is currently a principal member of the research staff with the Department of Knowledge converged Super Brain. Since September 2004, he is a professor with the University of Science and Technology, Taejeon, Korea.

He has worked in a range of fields, including packet switching and frame relaying for ISDN switching system, design of high-speed IP router and all-optical cross-connect system, a new transport mode for the post-Internet era, low power wireless communication for wireless sensor networks, design of evolving Internet of Things Infrastructure, and design of autonomous collaborative work platform for intelligent things.

His research interests include intelligent multi-agent system, distributed collaborative learning, procedural knowledge representation, and infrastructure for the networking of everything.



Dr. Hyun-Kook Kahng is a Professor of Department of Electronics and Information Engineering at Korea University Sejong campus, Sejong city, Korea. He received bachelor's degree in Electronics Engineering from Korea University in 1982, master's degree in Computer Engineering from University of Michigan, Ann Arbor in 1984, and Ph.D. in Computer Networking from Georgia Institute of Technology in 1990.

From 1991 to 1994, he was a Senior Research Engineer at ETRI, Korea. Since 1994, he has been a professor in Korea University. He has been standardizing, as a Korean delegate, in various SDO's such as ITU-T, ISO/IEC JTC1 SC6 and IETF in the field of computer networking. His current research interest includes, future networking technologies, especially for IoT, Blockchain, wearable devices, and autonomous cooperative system.