

Design and implementation of real-time information sharing system of IoT using distributed social network

Chang-Su LEE*, Dong-Hee JUNG**

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

lcs3708@korea.ac.kr, jdh0139@korea.ac.kr

Abstract— Registering a large number of IoTs with a central main server and coordinating the collaboration of the IoT on that server for the services they want to provide can cause significant traffic to the network. If many IoTs request cooperation on the main server to provide services, many data will be concentrated on the main server and the network to which the IoT is connected, and the server will have to use its performance to the maximum for the many calculations of the collaboration required by the various IoTs. And the security level will have to be very high as the central main server has important information on all IoTs as well as collaborative calculations for service offerings, which can be targeted by hackers. To solve these problems, IoT structure and collaboration algorithm were designed and implemented by IoT to find cooperative partners through distributed social networking. If IoTs collaborate and provide services directly without a main server, they will avoid network traffic and security issues that are concentrated on the server.

Keywords— Autonomous collaboration of IoT, Distributed social networking, Distributed social network of IoT, Connected car, WebID

I. INTRODUCTION

The architecture of registering many IoTs on the central main server and coordinating the collaboration of IoTs on that server for the services they want to provide can cause serious traffic to the network. If many IoTs request cooperation on the main server to provide services, many data will be concentrated on the main server and the network to which the IoT is connected, and the server will have to use its performance to the maximum for the many calculations of the collaboration required by the various IoTs. The more IoTs are registered with the central main server, the higher its performance will have to be in order to avoid overload. And how to register all IoTs with a central main server and collaborate through that server comes with an important problem of security. The security level will have to be very high since the central main server has important information on all IoTs as well as calculating collaboration to provide services. If there is a problem with the central main server, many linked IoTs will be almost useless. Therefore, server performance, surrounding network, and security should be thoroughly configured so that the server can operate smoothly

at any time. In order to solve these problems, IoT has designed the IoT structure and collaboration algorithms to find social partners through deep learning, form social groups, and provide services through distributed social networks [1][2][3]. If IoTs collaborate and provide services directly without a main server, they will avoid network traffic and security issues that are concentrated on the server. In this paper, one of the many applications developed for connected cars [4][5][6], using a vehicle number recognition function and distributed social network, simply designed and implemented a system in which driving vehicles share key driving information in real time. Without a central main server, driving vehicles directly recognize the vehicle number of the vehicle in front of them, create a social group, and share each other's vehicle and driving information in real time through distributed social networking.

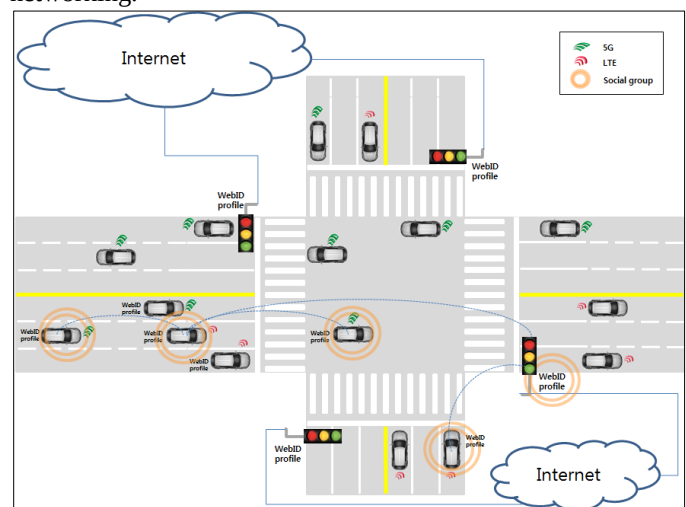


Figure 1 Connected car sharing traffic light information and front / rear vehicle driving information

If the solution for autonomous vehicles can utilize various sensors, artificial intelligence and even sharing driving information among vehicles through distributed social networks, it could become a safer autonomous vehicles solution in the connected car environment. In particular, if the driving information of the vehicle is shared in real time, it will be very effective to develop a solution for predictive driving

required during autonomous driving. In addition, traffic signal structures such as traffic lights and road signs could be made available for sharing information over distributed social networks. Figure 1 shows a car sharing the traffic light information and the driving information of nearby vehicles through a distributed social network.

We defined a vehicle as an IoT. The identity information of the IoT refers to the web server address and WebID of the IoT, and the driving information refers to speed, direction signal, and braking state. All IoTs were implemented through laptops. In session II, we explained WebID, WebID server, SPARQL, and Apache Jena Fuseki as the background of related technologies, and in session III, we explained the structure of IoT, vehicle WebID and profile, and vehicle number recognition algorithm using Convolution Neural network(CNN) [7][8]. Session IV describes the design and implementation of the real-time driving information sharing system among vehicles in detail, and Session V describes the conclusion and future research plan.

II. BACKGROUND

A. WebID and WebID Server

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 [9]. 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 among WebID URI and WebID Profile document.

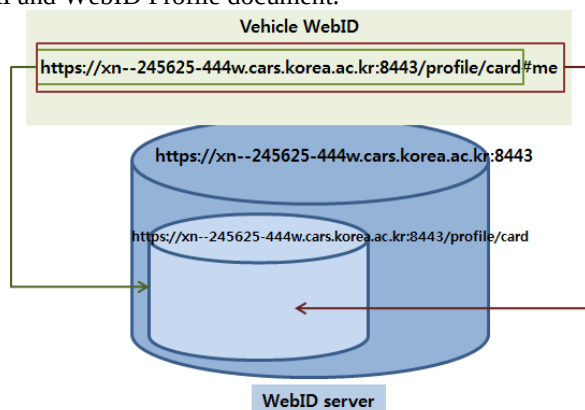


Figure 2 Relationship between WebID and WebID profile

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 a WebID URI using the vehicle's WebID as an example. <https://xn--245625-444w.cars.korea.ac.kr/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 [9].

In this paper, we create a WebID using a unique vehicle number that all cars have. The unique license plate number creates a unique WebID that does not overlap in the world. In this paper, we used Solid Server program provided by SOLID (Social Linked Data) as WebID Server. WebID Server has functions of WebID creation and management. It provides a set of rules and tools that we propose to build distributed social applications based on Linked data principles [10][11].

B. 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 [12]. The main purpose of SPARQL is to provide a formal language for expressing questions with semantic information [13]. 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 request searches the query and returns the result. SPARQL Endpoint uses the encoded URI to avoid special characters when it receives a query via HTTP, and has a variable named query in the SPARQL query string. In this paper, we use Apache Jena Fuseki (hereinafter Fuseki) as SPARQL server. Fuseki is a SPARQL server that provides a user interface for data security, server monitoring and management, and a TDB (Triple Database), an RDF data store [14]. Fuseki also uses the SPARQL 1.1 protocol to update RDF data as well as queries. Each IoT has a Fuseki, and can make queries using HTTP remotely through the Fuseki's Endpoint.

C. Apache Jena Fuseki

It is a SPARQL server provided by Apache Jena. Fuseki provides a user interface for server monitoring and management and provides TDB, an RDF repository for RDF storage and query. Fuseki provides SPARQL queries and SPARQL updates by connecting to query endpoints and update endpoints via HTTP communication. The SPARQL query is a function that Fuseki queries the TDB owned by its own TDB or other Fuseki, and the SPARQL update refers to the function of inputting and modifying data in the TDB. A SPARQL endpoint is a data provider such as DBpedia that enables SPARQL queries through the HTTP API [14].

III. APPROACH

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

A. IoT structure

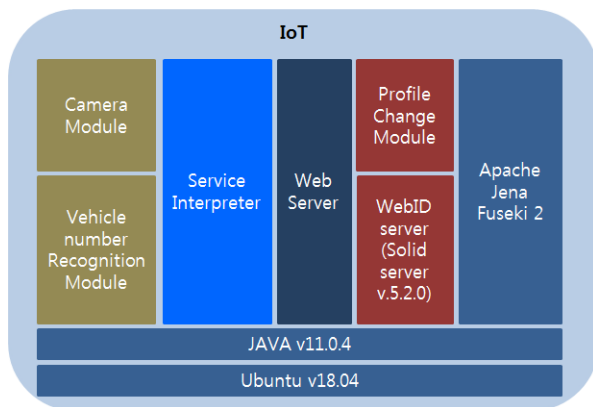


Figure 3 IoT structure

In this paper, the IoT uses Intel (R) Pentium (R) CPU 2117U @ 1.80GHz 1.80 GHz, RAM 8GB, Ubuntu 18.04 notebook and Intel (R) Core (R) i5 CPU U 470 @ 1.33GHz, RAM 16GB, Ubuntu 16.04 notebook. The structure is shown in Fig. 4. Based on Ubuntu 18.04 and JAVA v11.0.4, we installed a web server to communicate with other IoT, and installed a vehicle number recognition module to recognize the vehicle number of the front vehicle. And Apache Jena Fuseki2 was installed to load the profile of partner IoT, SPARQL to endpoint, and save information such as experience knowledge in TDB. Finally, we installed WebID server (Solid server) for WebID and profile creation and management [17].

B. WebID and Profile of IoT

We created a Web Server address and WebID using a unique vehicle number that every car had. In this paper, all IoTs for vehicles and traffic lights have a web server and a webID. WebID and profile of each IoT are shown in Table 1.

TABLE 1. IoT'S WEBID AND PROFILE

IoT	Vehicle Number or Traffic light	Profile
IoT A	24 가 5624	https://xn--245625-444w.cars.korea.ac.kr:8443/profile/card#me https://24가5625.cars.korea.ac.kr:3000 Speed:24, TurnSignal:off, Brake:off
IoT B	30 누 1298	https://xn--301298-4x3x.cars.korea.ac.kr:8443/profile/card#me https://30누1298.cars.korea.ac.kr:3000 Speed:32, TurnSignal:off, Brake:on
IoT C	30 루 6051	https://xn--306051-x30y.cars.korea.ac.kr:8443/profile/card#me https://30루6051.cars.korea.ac.kr:3000 Speed:26, TurnSignal:off, Brake:off
IoT D	Traffic light	https://trafficlight.korea.ac.kr:8443/profile/card#me https://trafficlight.korea.ac.kr:3000 RED:on, TimeRemaining:15, GREEN:off

Profile of vehicle IoT includes web server address and WebID, driving information such as direction signal, braking status and driving speed. In this paper, we simply defined the direction signal, braking state and driving speed as vehicle driving information for the test. IoT D acts as a traffic light. It shows the Web Server address and WebID, the light status and the time remaining to change the light signal. Figure 4 shows the programmed profile of IoT A [17].

```

@prefix solid: <http://www.w3.org/ns/solid/terms#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix pim: <http://www.w3.org/ns/pim/space#>.
@prefix schema: <http://schema.org/>.
@prefix ldp: <http://www.w3.org/ns/ldp#>.

<>
  a foaf:PersonalProfileDocument ;
  foaf:maker <https://xn--245625-444w.cars.korea.ac.kr:8443/profile/card#me> ;
  foaf:primaryTopic <https://xn--245625-444w.cars.korea.ac.kr:8443/profile/card#me> .

<https://xn--245625-444w.cars.korea.ac.kr:8443/profile/card#me>
  a foaf:Person ;
  a schema:Person ;

  <http://sac.korea.ac.kr:3000/ontology#Server> <https://24가5625.cars.korea.ac.kr:3000> ;
  <http://sac.korea.ac.kr:3000/ontology#Speed> 24 ;
  <http://sac.korea.ac.kr:3000/ontology#TurnSignal> "off" ;
  <http://sac.korea.ac.kr:3000/ontology#Brake> "off" ;

  foaf:name "car1" ;

  solid:account </> ; # link to the account uri
  pim:storage </> ; # root storage

  ldp:inbox </inbox> ;

  pim:preferencesFile </settings/prefs.ttl> ; # private settings/preferences
  solid:publicTypeIndex </settings/publicTypeIndex.ttl> ;
  solid:privateTypeIndex </settings/privateTypeIndex.ttl> .

```

Figure 4 Profile of IoT A

C. Vehicle Number Recognition using CNN

License plates used in Korea are divided into three types. There is a 7-digit general vehicle license plate with a combination of Korean and numbers. Recently, in order to secure a new car number, it has been reorganized into an 8-digit license plate in numbers and Korean. There are also nine single or two lines of business license plates of Korean and numeric combinations used for taxis, buses, etc. In this paper, for the convenience of implementation, an algorithm for recognizing 7-digit general vehicle license plates is implemented except for commercial vehicles. A single seven-digit license plate is shown in Figure 5. The letters of general vehicles consist of 10 numbers from 0 to 9 and 32 Korean characters.



Figure 5 7-digit general vehicle license plate

3500 characters were extracted and labelled from 500 images of 240 × 480. The patch image size was set to 32 × 64 by calculating the distribution of height and width of the labelled character image.



Figure 6 Example of sliding window

When the image is input, the text part image is extracted while moving the 32×64 (width x height) window by 2 pixels in the horizontal and vertical directions using the sliding window method as shown in Figure 6. The structure of the CNN model is shown in Figure 7. Since the vehicle number has seven characters and one empty space in one column in a 240×420 image, the following character confidence matrix [8] can be obtained.

$$P_c(a, b) = \max_{1 \leq k \leq 43} R_c(a, b, k) \quad (1)$$

In Equation (1), R denotes a $240 \times 480 \times 43$ probability output value of the CNN for one full image. And (a, b, k) means pixels in the vertical direction $1 \leq a \leq 240$, horizontal direction $1 \leq b \leq 480$, $1 \leq k \leq 43$ rank.

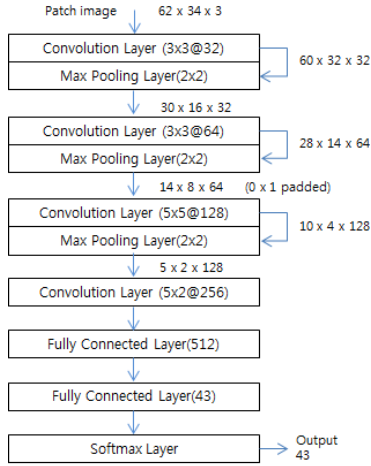


Figure 7 Structure of the CNN Model

The score function of the algorithm for classifying characters simultaneously with character area detection is as follows.

$$T(y, x, w) = \sum_{i=1}^8 P_c(y, x + (i-1)w) \quad (2)$$

In Equation (2), $1 \leq y \leq 240, 1 \leq x \leq 420 - 8w$ denote pixel positions in the horizontal and vertical directions of the first character of the string in the character matrix, respectively. And $0.8 \times (\frac{32}{2}) \leq w \leq 1.5 \times (\frac{32}{2})$ is to find the optimal character spacing considering the size of the matrix. The best way to maximize the score function is

$$(y^*, x^*, w^*) = \operatorname{argmax}_{y, x, w} T(y, x, w) \quad (3)$$

The problem-class S_i showing the highest probability of its position using (y^*, x^*, w^*) is as follows. Here is $i = 1..8$

$$S_i = P_c(y^*, x^* + (i+1)w^*) \quad (4)$$

Although there are many excellent vehicle number recognition programs in Korea, it is difficult to implement them in conjunction with distributed social network programs, so a simple recognition program using CNN is implemented.

It is implemented to link the two programs and does not take into account the excellent license plate recognition rate. For the CNN algorithm used in this study, refer to “Billet Number Recognition Algorithm using Deep Learning” published in Journal of Institute of Control, Robotics and Systems 23(6), 2017[8].

IV. IMPLEMENTATION

A. System implementation environment

For implementation, LTE routers and laptops were configured in each vehicle as shown in Figure 8. I used a camera built into my laptop. Each notebook is named IoT A, IoT B, and IoT C so that there is no confusion.

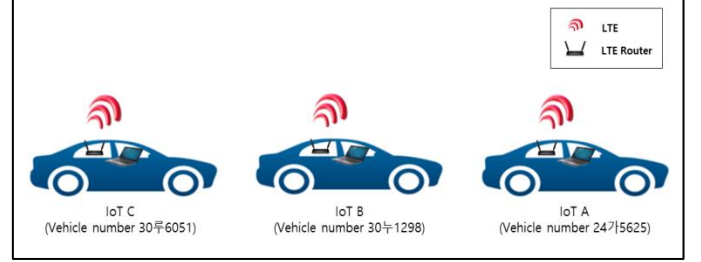


Figure 8 Driving information sharing system implementation environment

In this experiment, we used the LTE network provided by LG U Plus Co., Ltd., and driven the vehicles within 80 km/h within Bundang-gu, Seongnam-si, Gyeonggi-do, South Korea.

B. Create social through identification process(IDP)

Vehicle IoT has to create social with the in-driving another vehicle IoT and social with the traffic light IoT with fixed location. Vehicle IoT whose location is not fixed can be recognized by vehicle number. By recognizing the vehicle number and confirming the web server address generated by the vehicle number, the vehicle IoT can create social with the in-driving another vehicle IoT. The vehicle IoT informs social members of the brake status, turn signal status and driving speed through its profile.

The navigation map may be used as a way for the vehicle IoT to recognize the IoT, such as a fixed location of the traffic light. The web server address of the traffic light is entered in the navigation map. Then, the vehicle IoT can check the Web Server address of the traffic light displayed on the navigation map while driving and create a social to share the traffic light IoT's profile. Traffic light IoT provides information such as current signal and time remaining until change. In addition, as shown in Figure 1, the traffic light IoT located at the intersection can provide information on the vehicle IoT of driving on another road.

In this paper, we implemented only the parts that vehicle IoT creates social and shares its profile. In the future, we will plan to design and implement that the vehicle IoT and the traffic light IoT create social by connecting with the navigation map.

It recognizes the vehicle number of the previous IoT and makes social through IDP [1][2]. The process is shown in Figure 9 below. (1) When the camera of IoT B illuminates the

license plate of IoT A, (2) Vehicle number recognition module recognizes the vehicle number.

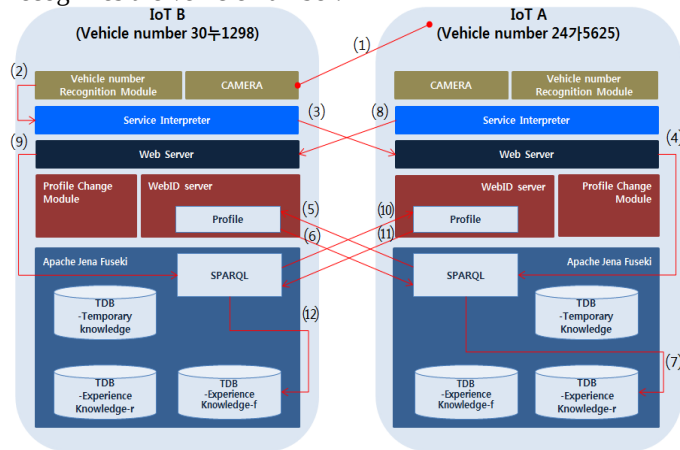


Figure 9 Create social through identification process

(3) Service Interpreter combines the web server domain with the recognized vehicle number. The screen shot of the process is shown in Figure 10. In this case, `<https://24-75625.cars.korea.ac.kr:3000>`.

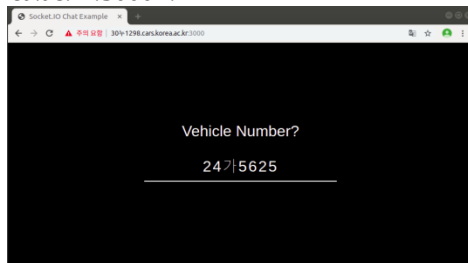


Figure 10 Recognized vehicle number input screen

Connect to this web server and log in with WebID `<https://xn-301298-4x3x.cars.korea.ac.kr:8443/profile/card#me>`. The program is shown in Table 2.

TABLE 2. LOGGING INTO THE WEB SERVER OF IOT WITH THE RECOGNIZED VEHICLE NUMBER

1	function setFrontCar(){
2	//Number of Vehicle entered
3	frontCarName = cleanInput(\$usernameInput.val().trim());
4	if (frontCarName.length > 0){
5	//Create a web server address to log in with the vehicle number entered.
6	var frontCarUrl = 'http://'+frontCarName+'.cars.korea.ac.kr:3000/login/'
7	// Query to find out WebID
8	var queryMyInfo = 'SELECT ?MyWebID ?Server WHERE { ?MyWebID
9	<http://sac.korea.ac.kr:3000/ontology#Server> ?Server}';
10	var reqMyInfo = 'http://localhost:3030/temp/query?query='+
11	encodeURIComponent(queryMyInfo);
12	var http1 = new XMLHttpRequest();
13	http1.open('GET', reqMyInfo);
14	http1.withCredentials = true;
15	http1.onreadystatechange = function(){
16	if (this.readyState == this.DONE){
17	if (this.status === 200){
18	var result1 = http1.responseText;
19	var resultJSON1 = JSON.parse(result1)
20	if(typeof(resultJSON1.results.bindings[0]) === 'undefined'){
21	console.log(' Please enter a web server address');
22	} else {
23	// own's WebID
24	My_WebID = resultJSON1.results.bindings[0].MyWebID.value
25	My_Server = resultJSON1.results.bindings[0].Server.value
26	var sendData = {'WebID': My_WebID};

25	var sendJson = JSON.stringify(sendData)
26	// Log in to the web server of the IoT in front
27	var http2 = new XMLHttpRequest();
28	http2.open('POST', frontCarUrl);
29	http2.setRequestHeader('Content-type', 'application/json');
30	http2.withCredentials = true;
31	http2.onreadystatechange = function(){
32	console.log(this.status)
33	if (this.readyState == this.DONE){
34	if (this.status === 200){
35	console.log('Successful data transfer')
36	} else {
37	console.log('Data transfer failed')

(4) Send WebID to Fuseki, (5) Fuseki SPARQL to profile of IoT B through WebID, and (6) Load profile of IoT B. The program is shown in Table 3.

TABLE 3. GET PROFILE OF IOT LOCATED BEHIND

1	function rearcar (WebID) {
2	// Attempt to log in to IoT located at the back, and check if the information already exists in the WebID in the loaded profile.
3	var querySN = 'SELECT ?Server WHERE { <'+WebID+'>
4	<http://sac.korea.ac.kr:3000/ontology#Server> ?Server}';
5	var reqSN = 'http://localhost:3030/sn/query?query='+
6	encodeURIComponent(querySN);
7	var http1 = new XMLHttpRequest();
8	http1.open('GET', reqSN);
9	http1.withCredentials = true;
10	http1.onreadystatechange = function(){
11	if (this.readyState == this.DONE){
12	if (this.status === 200){
13	var result1 = http1.responseText;
14	var resultJSON1 = JSON.parse(result1)
15	if(typeof(resultJSON1.results.bindings[0]) === 'undefined'){
16	//If you do not have a WebID, log in and import the profile.
17	var url = 'http://localhost:3030/sn/update'
18	var query = "LOAD <'+WebID+'>";
19	var http2 = new XMLHttpRequest();
20	http2.open('POST', url);
21	http2.setRequestHeader('Content-Type', 'application/sparql-update');
22	http2.withCredentials = true;
23	http2.onreadystatechange = function() {

(7) And store the profile in the TDB-Experience knowledge-r. The structure of the TDB is shown in Table 4.

TABLE 4. TDB STRUCTURE IN IOT

TDB	Role
TDB-temporary knowledge	Store temporary data
TDB-experience knowledge-f	Stores the driving information of the front IoT
TDB-experience knowledge-r	Stores the driving information of the rear IoT

(8) IoT A checks Web Server information of IoT B in the saved profile. In the Service Interpreter, log in to IoT B's Web Server as `https://xn-245625-444w.cars.com.8443/profile/card#me`. (9) Send WebID to Fuseki, (10) Fuseki SPARQL to profile of IoT A through WebID, and (11) LOAD profile of IoT A. The program is shown in Table 5.

TABLE 5. GET PROFILE OF THE IOT IN FRONT

22	if (this.readyState == this.DONE){
23	if (this.status === 204){
24	//After importing the profile, log in to the web server of IoT located behind.
25	var queryMyInfo = 'SELECT ?MyWebID ?Server WHERE
26	{ ?MyWebID <http://sac.korea.ac.kr:3000/ontology#Server> ?Server}';
27	var reqMyInfo = 'http://localhost:3030/temp/query?query='+
28	encodeURIComponent(queryMyInfo);

```

27 var http3 = new XMLHttpRequest();
28 http3.open('GET', reqMyInfo);
29 http3.withCredentials = true;
30 http3.onreadystatechange = function(){
31   if (this.readyState == this.DONE){
32     if (this.status === 200){
33       var result3 = http3.responseText;
34       var resultJSON3 = JSON.parse(result3)
35       if(typeof(resultJSON3.results.bindings[0]) === 'undefined'){
36       } else {
37         My_WebID =
resultJSON3.results.bindings[0].MyWebID.value
38         My_Server = resultJSON3.results.bindings[0].Server.value
39         // Find the web server address of IoT located behind in the profile.
40         var queryRearInfo = 'SELECT ?Server WHERE
{ <'+WebID+'> <http://sac.korea.ac.kr:3000/ontology#Server> ?Server}';
41         var reqRearInfo = 'http://localhost:3030/sn/query?query='+
encodeURIComponent(queryRearInfo);
42         var http4 = new XMLHttpRequest();
43         http4.open('GET', reqRearInfo);
44         http4.withCredentials = true;
45         http4.onreadystatechange = function(){
46           console.log(this.status)
47           if (this.readyState == this.DONE){
48             if (this.status === 200){
49               var result4 = http4.responseText;
50               var resultJSON4 = JSON.parse(result4)
51               if(typeof(resultJSON4.results.bindings[0]) ===
'undefined'){
52             } else {
53               // Found the web server address of the IoT behind.
54               Rear_Server =
resultJSON4.results.bindings[0].Server.value
55               var Rear_Login = Rear_Server+'login/'
56               // WebID in JSON format to login to the IoT behind.
57               var sendData = {'WebID': My_WebID };
58               var sendJson = JSON.stringify(sendData)
59               // Login to the IoT behind you with a WebID.
60               var http5 = new XMLHttpRequest();
61               http5.open('POST', Rear_Login);
62               http5.setRequestHeader('Content-type', 'application/json');
63               http5.withCredentials = true;
64               http5.onreadystatechange = function(){
65                 console.log(this.status)
66                 if (this.readyState == this.DONE){
67                   if (this.status === 200){
68                     console.log('Successful data transfer')
69                   } else {
70                     console.log('Data transfer failed')

```

(12) And save the profile in TDB-Experience knowledge-f. The contents are shown in Figure 11.

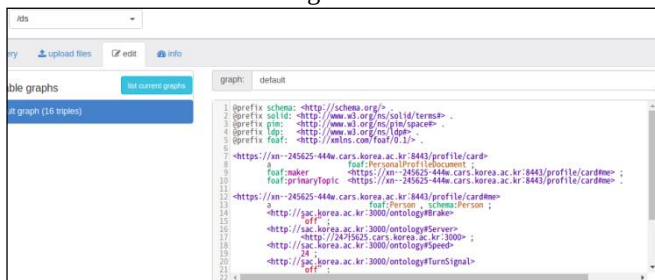


Figure 11 Profile of IoT A stored in IoT B's TDB

IoT A and IoT B, having profiles of each other, become social members. And in this way, IoT B becomes a social member with IoT C. When the front vehicle number changes, the vehicle number recognition is performed again, erasing past social members and creating a new social. And the delay time of the flow is shown in Figure 12 and Table 6.

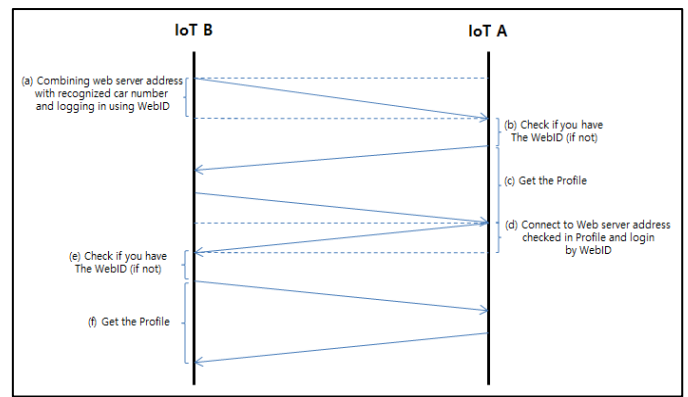


Figure 12 IDP flow

TABLE 6. DELAY TIME IN THE FLOW

	(a)	(b)	(c)	(d)	(e)	(f)
Test 1	1,174ms	7ms	907ms	122ms	37ms	903ms
Test 2	1,141ms	12ms	1,245ms	658ms	46ms	1,370ms
Test 3	1,512ms	9ms	993ms	149ms	31ms	1,071ms

C. Profile update using subscribe/publish mechanism

When the driver of the vehicle ahead changes the driving information, such as braking or turning on the turn signal, the social vehicle updates the changed driving information to the TDB in real time. The rear vehicle can drive safely by checking driving information of the front vehicle in real time. This feature is designed and implemented as shown in Figure 13 and Table 6. When the profile was changed through the profile change module in IoT B, IoT A and IoT C, which are socializing with IoT B, confirmed the change by updating the changed profile of IoT B in real time using the subscribe / publish[15][16] mechanism. Figure 13 shows the update process of IoT C when the profile of IoT B is changed.

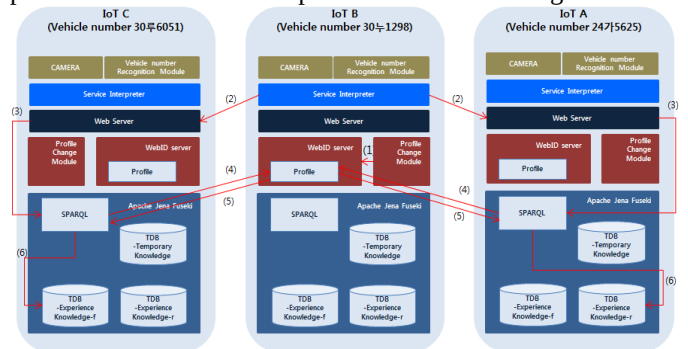


Figure 13 Profile update process using subscribe/publish mechanism

(1) Change the profile of IoT B using the profile change module as shown in Figure 14. (2) When the profile is changed, the Service Interpreter logs into the Web Server of IoT A and IoT C, the social members stored in the TDB-Experience knowledge.

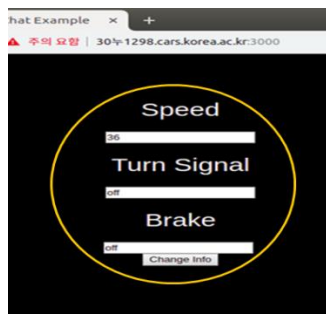


Figure 14 Profile of IoT B changed by profile change module

(3) Send IoT B's WebID to Fuseki. (4) Fuseki SPARQL the profile of IoT B, and (5) Load the profile of IoT B. The program is shown in Table 7.

TABLE 7. PROFILE UPDATE

```

87      http2.send(query);
88    } else {
89      // If you already have a profile, update the profile and do not log in to IoT later.
90      SN_Server = resultJSON1.results.bindings[0].Server.value
91      var url = 'http://localhost:3030/sn/update'
92      var query = "LOAD <'"+WebID+"'>";
93      var http6 = new XMLHttpRequest();
94      http6.open("POST", url);
95      http6.setRequestHeader('Content-Type', 'application/sparql-update');
96      http6.withCredentials = true;
97      http6.onreadystatechange = function() {
98        console.log(this.status)
99        if (this.readyState == this.DONE){
100          if (this.status == 204){
101            console.log("WebID profile Input success. "+WebID);
102          } else {
103            console.log("WebID profile Input failed.")
104          }

```

(6) And update the TDB-Experience knowledge-f as shown in Figure 15.

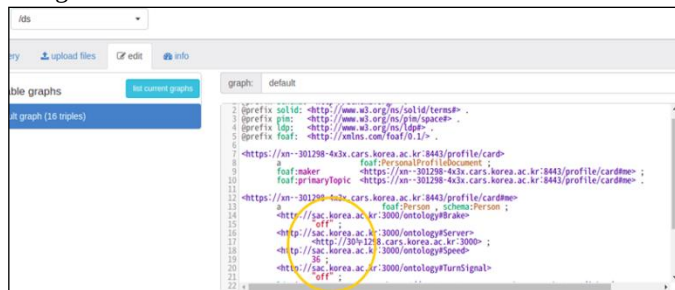


Figure 15 Profile of IoT B updated in IoT C's TDB

In this way, IoT B can also update IoT B profile by logging into the web server of IoT A, a social member stored in TDB-Experience Knowledge-r. And the delay time of the flow is shown in Figure 16 and Table 8.

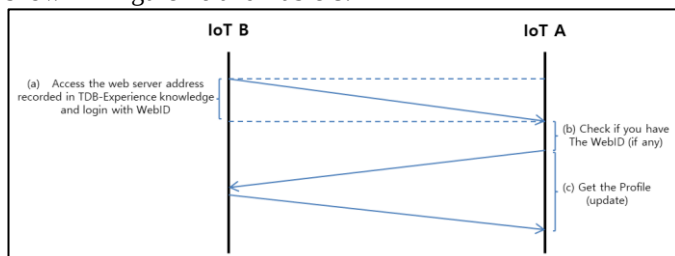


Figure 16 Profile update flow

TABLE 8. DELAY TIME IN THE FLOW

	(a)	(b)	(c)	(d)	(e)	(f)
Test 1	1,174ms	7ms	907ms	122ms	37ms	903ms
update	321ms	13ms	1,729ms			
update	444ms	6ms	915ms			
update	515ms	7ms	1,528ms			
update	551ms	7ms	1,211ms			
update	721ms	10ms	1,159ms			
update	406ms	7ms	1,081ms			
update	500ms	7ms	1,734ms			
Test 2	1,141ms	12ms	1,245ms	658ms	46ms	1,370ms
update	92ms	9ms	899ms			
update	211ms	9ms	1,300ms			
update	980ms	12ms	910ms			
update	114ms	13ms	645ms			
update	486ms	7ms	664ms			
update	81ms	13ms	644ms			
update	511ms	17ms	767ms			
update	113ms	10ms	759ms			
Test 3	1,512ms	9ms	993ms	149ms	31ms	1,071ms
update	236ms	14ms	760ms			
update	493ms	9ms	798ms			
update	292ms	13ms	899ms			
update	355ms	12ms	826ms			
update	478ms	12ms	667ms			
update	557ms	8ms	910ms			
update	297ms	10ms	783ms			

V. CONCLUSIONS

If a large number of vehicles register with a central main server and share information in real time through the server, huge amounts of data will drive traffic on the network. However, by identifying vehicle numbers directly in the vehicle and confirming that they can collaborate through distributed social networking, this approach shows the possibility of preventing traffic on the network and the security problems of the main central server. In the future, we plan to test the system and analyse performance in several ways to reduce delay time. In addition, we will analyze the autonomous collaboration performance of IoT by creating a more complex application service that various types of IoT collaborate using distributed social networks.

ACKNOWLEDGMENT

Thanks to Professor Min-Ho CHO for teaching me about machine learning, Professor Yong-Wha CHUNG for teaching me about computer vision. Also thanks to Professor Chul HAN, professor Hyun-Kook KAHNG, and Dr. Seong-Soon JOO of the Electronics and Telecommunications Research Institute, for helping me get ideas about the application of distributed social networks. And thanks to junior Yu-Seok JI who helped with experiment. This research was supported by the Korea Evaluation Institute of Industrial Technology project, "Standardization of Networking of Everything Architecture and Protocols (20002532)".

REFERENCES

- [1] Chang-Su LEE, "Design and Implementation of Autonomous Collaboration System Using Distributed Social Network of IoTs," 2019 Summer Annual Conference of IEIE, pp1056-1058, 2019.

- [2] Chang-Su LEE, Sweung-Won CHEUNG, Seong-Soon, JOO, Hyun-Kook, KAHNG, "Design and Implementation of Autonomous Collaboration System of Smart Things using accumulated Experience knowledge," International Conference on Advanced Communication Technology(ICACT) ISBN:979-11-88428-03-8, pp305-313, 2019.
- [3] Tramp, Sebastian, "Distributed Semantic Social Networks: Architecture, Protocols and Applications," Universitätsbibliothek Leipzig, 2014
- [4] Miller Michael, "The internet of things :how smart TVs, smart cars, smart homes, and smart cities are changing the world," 2016 ISBN 9788931451887
- [5] Shun-Ren Yang , Yu-Ju Su, Yao-Yuan Chang, and Hui-Nien Hung, "Short-Term Traffic Prediction for Edge Computing-Enhanced Autonomous and Connected Cars," IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 68, NO. 4, pp 3140-3153, 2019
- [6] Fatma Raissi, Sami Yangui, Frederic Camps, "Autonomous Cars, 5G Mobile Networks and Smart Cities: Beyond the Hype," 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)," pp180-185, 2019
- [7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Computation, vol. 1, no. 4, pp. 541-551, 1989.
- [8] Gyogwon Koo, Sang Jun Lee, Sang Woo Kim, "Billet Number Recognition Algorithm using Deep Learning," Journal of Institute of Control, Robotics and Systems 23(6), p411-415, 2017
- [9] Tim Berners-Lee, Andrei Sambra, Henry Story, <https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html>
- [10] Tim Berners-Lee, Christian Smith, Sarven Capadisli, Dmitri Zagidulin, Nicola Greco, Ruben Verborgh, <https://github.com/solid/solid>
- [11] Steve Speicher, John Arwe, Ashok Malhotra, Linked Data Platform 1.0, 26 February 2015
- [12] Bob DuCharme, Learning SPARQL, Second Edition, June 2013, ISBN 978-1-449-37143-2
- [13] David Wood, Marsha Zaidman, Luke Ruth, Michael Hausenblas, Linked Data Structured data on the Web, 2017, 3, ISBN 979-11-950726-4-4
- [14] Apache Jena, <https://jena.apache.org/documentation/fuseki2/>
- [15] Luca Roffia, Paolo Azzoni, Cristiano Aguzzi, Fabio Viola, Francesco Antoniazzi, Tullio Salmon Cinotti, "Dynamic Linked Data: A SPARQL Event Processing Architecture," Future Internet; Basel Vol. 10, Iss. 4, MDPI AG, 2018.
- [16] Jae-Gon, Jung, "Node.js programming," easies publishing, 2017. ISBN 979-11-873708-0-2
- [17] Chang-Su, Lee, Dong-Hee Jun "Design and implementation of real-time driving information sharing system among vehicles using deep learning and distributed social networks" 2019 2019 KICS Korea-Vietnam International Joint Workshop on Communications and Information Sciences, p50-56, 2019, ISBN: 978-89-950043-7-1 [93560]

2018. His research interest includes, Deep Learning, IoT, autonomous collaboration system, and future networking technology.



Chang-Su Lee 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 conference 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, Deep Learning, IoT, autonomous collaboration system, and future networking technology



Dong-Hee Jung is associate research engineer in SW research center, METABUILD Co., Ltd. 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 2015, and received Master's degree from same University in