World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:3, No:8, 2009

B-VIS Service-oriented Middleware for RFID Sensor Network

Wiroon Sriborrirux, Sorakrai Kraipui, and Nakorn Indra-Payoong

***Abstract*—**One of the most importance of intelligence in-car and roadside systems is the cooperative vehicle-infrastructure system. In Thailand, ITS technologies are rapidly growing and real-time vehicle information is considerably needed for ITS applications; for example, vehicle fleet tracking and control and road traffic monitoring systems. This paper defines the communication protocols and software design for middleware components of B-VIS (Burapha Vehicle-Infrastructure System). The proposed B-VIS middleware architecture serves the needs of a distributed RFID sensor network and simplifies some intricate details of several communication standards.

***Keywords*—**Middleware, RFID sensor network, Cooperative vehicle-infrastructure system, Enterprise Java Bean.

I. INTRODUCTION

BANGKOK city and its vicinity have been facing traffic and transport problems for long time. This substantially diminishes the logistics efficiency of the city. A working group from Burapha University is the one of initiative cooperative vehicle-infrastructure system teams who proposes the RFID sensor network infrastructure to alleviate such problems [1]. The proposed middleware architecture for RFID sensor network is a part of success of the system.

We propose the B-VIS middleware architecture that serves the needs of our distributed RFID sensor nodes for real-time vehicle fleet tracking and control and road traffic monitoring systems. In addition, we present an incorporated programming model to interested application developers to retrieve the real-time vehicle information generated from the RFID sensor stations. The incorporated model reduces the complexities of the heterogeneous communication layers and data sources.

This paper is structured as follows; we first present the current efforts in the field of CVIS [2]. Then, we present the B-VIS middleware architecture and propose the service interface for some service layers in order to make the B-VIS to be a service-oriented middleware. The functionalities of each service layer are also discussed is this section. In the last section, we evaluate the performance metric of end-to-end delay time connection and conclude our RFID testbed implementations in real-use aspect.

Wiroon Sriborrirux and Sorakrai Kraipui are with Electrical Engineering Department, Faculty of Engineering, Burapha University, Chonburi, 20131, Thailand (e-mail: sriborrirux, kraipui@gmail.com).

Nakorn Indra-Payoong is with Faculty of Logistics, Burapha University, Chonburi, 20131, Thailand (e-mail: nakorn.ii@gmail.com).

II. CURRENT EFFORTS ON CVIS

One of the most important functions of intelligent transport system (ITS) applications is to support vehicle-to infrastructure system. Generally, the system comprises a beacon connected to the roadside infrastructure and used to maintain wirelessly contact with passing vehicles.

In many countries, like US, EU and Japan [3], a number of projects exist for the development and deployment of cooperative-vehicle infrastructure systems (CVIS). In the US, called Vehicle Infrastructure Integration (VII) is an initiative research and applications development for a series of ITS technologies directly linking vehicles to their roadside infrastructure in order to improve road safety as well as traffic efficiency. In European countries, CVIS underlines the importance of intelligence in-car and roadside systems for improving traffic safety and efficiency and environmental impact [4]. In Japan, the national Smartway project enables communication among vehicles, drivers and pedestrian with advanced ITS technologies [5]. In Thailand, there are only few CVIS working groups, and are still on going research and analyzing testbed implementation.

In this paper, we propose a specific service-oriented middleware for handling cooperative vehicle-infrastructure systems and also propose the Eclipse plug-in for enterprise application development environments built on the Eclipse platform for interested developers.

Our first CVIS effort is made for bus fleet monitoring system in the Bangkok city [6]. It is a coordinated deployment of communication technology for public transport operations, and shows major service improvements through real-time communication between vehicles and roadside RFID reader stations.

It is noted that the problems of heterogeneous applications and ITS information are still one of main issues in ITS exchangeable services. Therefore, the service-oriented middleware development has emerged as an important architectural component in supporting our distributed RFID sensors. We present a unified programming model to application developers and to mask out problems of heterogeneity and distribution. As increasing visibility of standardization activities, such as ISO/ITU-T Reference Model for Open Distributed Processing [7], Open Management Group (OMG)'s CORBA, the Java RMI [8], J2EE/EJB (Sun) and Microsoft's.NET, we then decide to focus on the service-oriented middleware development.

III. Propose of B-VIS Middleware

The proposed B-VIS middleware architecture serves the needs of our distributed RFID sensor network for vehicle fleet monitoring and control as well as the travel time estimation systems in Bangkok road network using RFID-vehicle probe data. We also present an incorporated programming model to the interested developers to retrieve the real-time data generated from RFID sensor stations directly and to reduce the complexities of the heterogeneous communication layers and data sources.

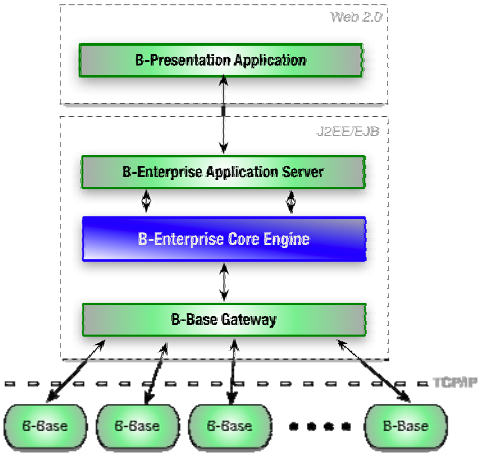

Fig. 1 The B-VIS middleware architecture

Fig. 1 shows B-VIS middleware architecture consisting of four layers: B-Base (RFID Reader) Gateway, B-Enterprise Core Engine, B-Enterprise Application Server, and B-Presentation Application.

Fig. 2, B-Base is installed inside the public telephone booths along the roadside, and a small active RFID tag is installed on a vehicle. When moving into the coverage beacon area, the tag will transmit vehicle ID to B-Base. All the data from B-Base are in subsequent sent to B-Base Gateway layer and then moved up to upper layers.

Fig. 2 B-Base (RFID Reader) Infrastructure

The B-Base Gateway is able to communicate and control each RFID B-Base station via VPN-based ADSL connection. In this layer, B-Base Gateway aggregates real-time data, e.g. vehicle identification, timestamp, and vehicle's passing time period. The B-Base Gateway will then filter out unnecessary data and send the data to B-Enterprise Core Engine layer. After that the B-Enterprise Core Engine layer will take the data into the business application processes, e.g. vehicle fleet monitoring and control. In this layer, the B-Enterprise Core Engine acts as an intermediary between the B-Base Gateway and B-Enterprise Application Server.

Having received the data from B-Enterprise Core Engine, the B-Enterprise Application Server processes the data and translates it into meaningful business events, which are defined by the users, such as vehicle routes, vehicle location, bus departure, and link travel time between two RFID reader stations. At the top layer, B-Presentation Application Server represents a specific business application, such as online web-based bus fleet monitoring, shown in Fig. 3.

Fig. 3 Online web-based bus fleet monitoring

IV. Propose of Service-Oriented Middleware

According to the proposed B-VIS middleware, we present the exchangeable information services from each layer to meet the service-oriented concept.

To follow a service-based framework, we present it based on open architecture methods, loose coupling between software components and hardware components that leverage existing hardware and communication media. This is shown in Fig. 4.

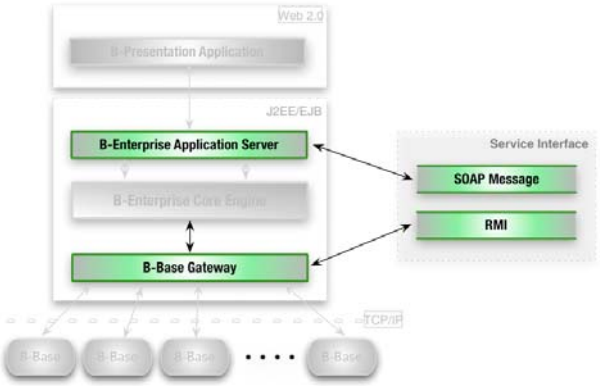

Fig. 4 B-VIS Middleware's Service Interface

The B-Base Gateway layer and Enterprise Application Server layer will be selected to add more Service Interface (SI) as the exchangeable information services. Firstly, we propose the SI at B-Base Gateway layer by using the RMI (Remote Method Invocation) technology [7], shown in Fig. 5.

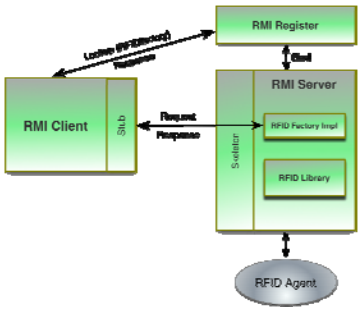

Fig. 5 RMI Interface

From technical viewpoint, RMI allows applications to call object methods located remotely, sharing resources and processing load across systems. Unlike other systems for remote execution, which require that only simple data types or defined structures be passed to and from methods, RMI allows any Java object type to be used - even if the client or server has never encountered it before. RMI allows both client and server to dynamically load new object types as required. Therefore, if a business application requests a real-time vehicle information, such as vehicle identification and timestamp, our proposed SI at B-Base Gateway layer could provide those data. See Fig. 6, the snapped codes of RMI Service Interface and RMI client.

```
// RMI Service Interface
import java.rmi.Remote;
import java.rmi.RemoteException;
...

public interface GatewayServiceInterface extends Remote {
....
public String getTimeStamp(String tagNo)throws RemoteException;
public String getPassingTime(String tagNo)throws RemoteException;
....
}
```

```
// RMI Client
import java.rmi.Naming;
import java.rmi.RMISecurityManager;
import ballabs.rmi.gateway.common.GatewayServiceInterface;
public class ApplicationClient {
public static void main(String[] args) {
  try {
    System.setSecurityManager(new RMISecurityManager());
    GatewayServiceInterface inventory =(GatewayServiceInterface)
            Naming.lookup("rmi://www.its-thailand.org/BVIS_Service");

    RecordTimeStamp(gateway.getTimeStamp("MO4"));
    AverageSpeed(gateway.getPassingTime("M04"));
                ....
    } catch (Exception e)
            {
                e.printStackTrace();
            }
}
```

Fig. 6 Service Interface and RMI Client

Secondly, we propose the SI at Enterprise Application Server layer by using the set of standards of communications called Simple Object Access Protocol (SOAP). It is a lightweight protocol for exchanging of information in a decentralized, distributed environment. It is also an XML-based protocol and is able to potentially be used in combination with a variety of other protocols. However, what is standardized and most commonly implemented of SOAP has been in combination with HTTP and the HTTP Extension Framework. As shown in Fig. 7 below, we provide the application services with SOAP request message.

```
POST /url HTTP/1.1
Host: HostServerName
Content-type: text/xml; charset=utf-8
Content-length: 350
SoapAction: http://www.its-thailand.org/GetBusArrivalTime
...

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <GetBusArrivalTime xmlns="http://www.its-thailand.org/">
<BusID>0454</BusID>
<OutputParam />
  </GetBusArrivalTime>
</soap:Body>
</soap:Envelope>
```

Fig. 7 SOAP Message

V. Eclipse Plug-in for Open Middleware

We develop SDK open middleware for interested developers, implemented in JAVA Enterprise Application using Eclipse IDE [9]. The Eclipse IDE is an open development platform which is designed to be easily and infinitely extensible by third parties. It provides an open source platform for creating an extensible integrated development environment and allows anyone to build tools that integrate seamlessly with the environment and other tools. For the above reasons, we develop our SDK as the Eclipse plug-in (RFID Agent plug-in) for generating the JAVA code. Interested developer could compile as the enterprise java bean (EJB) which is able to communicate with our RFID B-Base Gateway and is able to modify the functionalities and define the type of data regarding to the user's requirements. The RFID Agent Plug-in is shown in Fig. 8.

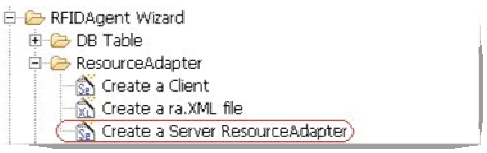

Fig. 8 RFID Agent Eclipse Plug-in

Once the application codes using RFID Agent on Eclipse have been created, we can compile and export war format file. This file will be deployed to Glassfish Open Enterprise Application Server. When a new B-Base Gateway is activated, we can monitor the status log, which is shown in Fig. 9.

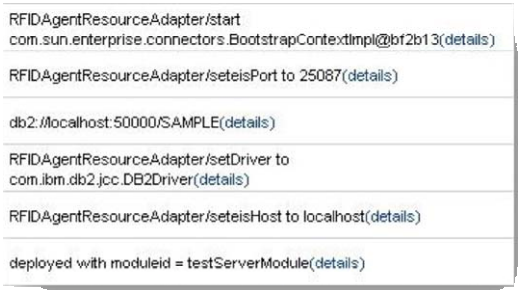

Fig. 9 Working Status on Enterprise Application Server

VI. RFID Sensor Network Evaluation

As the presented RFID sensor network infrastructure, we are now on process of extending B-Base stations in Bangkok city, and all B-Base stations are connected to the control center, which is shown in Fig. 10 below.

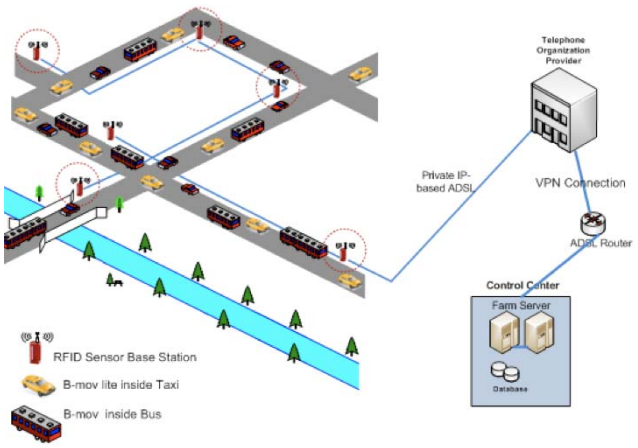

Fig. 10 RFID Sensor Network Infrastructure

We measure the end-to-end delay between RFID B-Base stations and the control center considering the latency time from transmitting the data packet (10 Bytes), which is received from passing vehicle tags. By logging thousand transactions from 20 B-Base stations, the end-to-end delay is reported in Fig. 11.

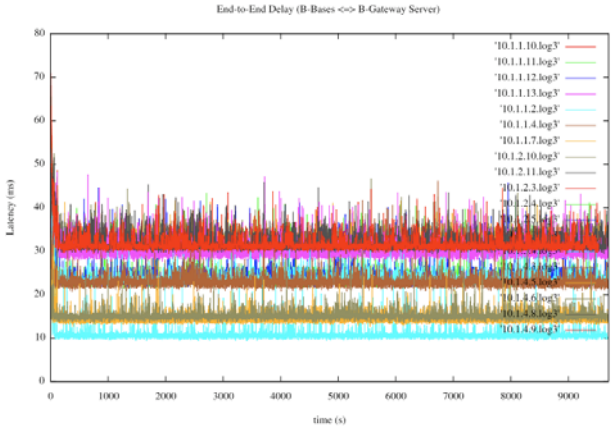

Fig. 11 End-to-End Delay Result

In Fig. 11, the average end-to-end delay time logged for two months (~50,000 transactions) between 20 B-Base stations and B-Gateway server is 26.9694 milliseconds, which can be considered as a cost of data transmission. Particularly, when applying to real-time vehicle fleet control or real-time travel time estimation system, the minimum transaction update rate is 20-30 milliseconds. This makes our testbed RFID sensor network and other real-time ITS applications more viable.

Fig. 12 Data Processing Time

Moreover, from Fig. 10 above, we measure the spent time of B-Base gateway layer for decapsulating the real-time vehicle data packet received from B-Base until that filtered data is inserted into the database server (DB2 IBM platform) completely. The result has shown that the average data processing time is around 5-6 milliseconds.

VII. Conclusion

From our RFID sensor testbed implementation presented in this paper, we address two main aspects. First, the B-VIS middleware which is specifically customized for our distributed RFID sensor system; it also simplifies the intricate details of the several communication standards. Second, providing the data sources via two simple standardized services, we implement the RMI interface that allows application developers to retrieve the data generated from RFID stations directly and SOAP interface that delivers a reliable real-time vehicle information, such as vehicle location, bus arrival time and road traffic information.

ACKNOWLEDGMENT

This research has been supported in part by a grant from the National Innovation Agency and the National Electronics and Computer Technology Center. The authors are also grateful to the Southeast Asia Technology company for help and support.

REFERENCES

[1] BAL-Labs. Burapha Advanced Logistics Labs, Burapha University 2009. http://www.bal-labs.com.
[2] CVIS. CVISProject.org. 2009. http://www.cvisproject.org
[3] Manasseh, C., Sengupta, R.: Middleware for Cooperative Vehicle-Infrastructure Systems, UCB-ITS-PRR-2008-2, California PATH Research Report, University of California, Berkeley (2008).
[4] Reding, V. "Speech delivered at the Intelligent Car Launching Event." The Intelligent Car Initiative: raising awareness of ICT for Smarter, Safer and Cleaner vehicle. Brussels (2006).
[5] Setsuo, Hirai. "Smartway project towards the next generation ITS." COM Safety: Newsletter for European ITS Related Research Projects (2007).
[6] Sriborrirux, W., Danklang P., Indra-Payoong, N.: The Design of RFID Sensor Network for Bus Fleet Monitoring, ITST2008, Phuket Thailand (2008).
[7] Tindale-Biscoe, Sandy: RM-ODP Enterprise Language (ISO/IEC 15414 | | ITU-T X.911), ITU-T/SG17 Meeting, Geneva (2002)
[8] Raj, Gopalan Suresh.: A Detailed Comparison of CORBA, DCOM and Java/RMI (1998).
[9] Eclipse Project. 2009. http://www.eclipse.org