MapUme: Scalable Middleware for Location Aware Computing Applications

Warsun Najib, Martin Klepal, Sigit Basuki Wibowo
Nimbus Centre for Embedded System Research
Cork Institute of Technology, Cork, Ireland
Email: {warsun.najib, martin.klepal, sigit.wibowo}@cit.ie

***Abstract*—In this paper, a scalable middleware for supporting location aware applications, MapUme is presented. Scalability feature will enable development of a broad range localisation systems from small size to a large number of localised objects. Distributed data processing supports the scalability requirement by distributing data processing load into several computing machines. The distributed data processing is implemented using service oriented architecture which fits well with the scalability requirement. This feature promotes scalability since a main server may forward requests to multiple service instances without the knowledge of the service client.**

***Index Terms*—middleware, scalability, localisation system, location aware application**

I. Introduction

Over recent years the development in the area of context aware computing have been growing rapidly. An important aspect of context is location, which is typically an output of a localisation system. Location information is usually not for direct use in application but to enable other business applications to utilise the location. Therefore middleware is the perfect solution to bridge between business application with location sensing technologies. A business application can subscribe to a location update of an object provided by the middleware.

A number of middleware platform have been proposed for providing location service to location based applications. Some middleware platforms developed for location service primarily focus on positioning method and hiding the process from location aware application such as Java Location API, PlaceLab [1], MiddleWhere [2], and TraX [3]. Other platforms concentrate on presentation and access to location information and geographic content such as Open GIS Location Service and Nexus [4]. However most of the middleware lack of functionality to support distributed data processing and therefore limits scalability of the system. Furthermore, many emerging localisation systems also need distributed data processing for scalability and performance reasons.

In this paper, we propose a middleware for location aware computing applications called ”MapUme” (Map You and Me). The middleware provides a platform for multi-sensor data fusion with distributed data processing capability to enable scalability and increase performance. MapUme offers location service which can represent location information in both physical and symbolic location. The middleware is also extensible to enable evolution of the middleware component and augmenting new location technology. MapUme provides middleware infrastructure for location awareness that separates business applications from location detection technologies as we can see in Figure 1.

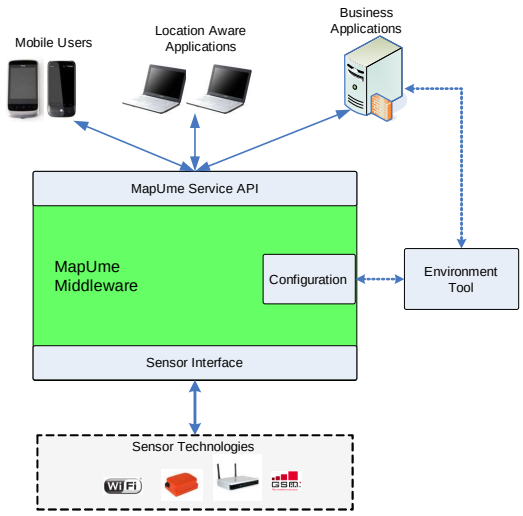

Fig. 1. MapUme provides middleware infrastructure for location awareness that separates business applications from location detection technologies

The remainder of the paper is organised as follow. In Section II the requirements of the location middleware are briefly presented. Section III describes middleware architecture. Section IV presents the experiment and evaluation of the middleware. Finally section V concludes the paper.

II. MapUme Middleware Requirements

The user requirements of a localisation system has been increased and now required more sophisticated architecture to meet those requirements. In this section these requirements are briefly identified and described.

A. Scalability and Distributed Data Processing

User requirement of the localisation system can be a range from simple to a large scale location aware application. Therefore, the middleware should be able to support development of a broad range localisation system from small size to a large number of localised objects. This requirement is related to distributed data processing requirement since the more object

we need to track the more computing resource is needed. When a computing machine is not adequate to process all localised objects we need to distribute the data processing load into several computing machines.

B. Usability

One of important property of a location middleware is usability. A middleware need to be flexible so that it can be implemented in a broad range of application domain. Hybrid location representation increases usability by providing two types of location information: physical and symbolic location [5]. Location aware applications may represent their locations either in terms of coordinates of reference, or symbolic names such as floor, corridor, and room number.

C. Extensibility

Extensibility is important property of a middleware since it will increase usability of the middleware and enable evolution of the localisation system. Interfaces definition of middleware components will enable extensibility of component implementation. A new implementation of a component such as fusion engine with different algorithm can be added by implementing those interfaces definition. Another important component is measurement which needs to be extended when different sensor technology is introduced in the localisation system.

Extensibility can also be seen in term of capability to incorporate different location sensing technologies. Different technologies give location information in different formats, different resolution and confidence. Middleware will enable the fusion of these different location sensing technologies.

III. MapUme Middleware Architecture

Mapume location middleware is developed based on a layered software development model for localisation system presented in [6]. The model provides common terminologies and abstraction layers for heterogeneous localisation systems and applications. It offers also an opportunity for constant evolution of localisation system development and deployment of new technologies into existing applications.

A. Middleware Components

The architecture of MapUme middleware can be seen in Figure 2. The following descriptions will give a brief explanation of the middleware components.

1) Measurement Component: The measurement layer defines a data structure of the sensor data and interfaces which has to be implemented for each type of measurement.

2) Aggregator Component: The Aggregation layer consists of Configuration, Measurement Logger, Distribution Manager component, and Database.

3) Fusion Engine Component: The fusion engine uses sequential non-linear Bayesian filter technique, implemented as particle filter [7]. The estimated position from the engine is stored into Object Logger component.

4) Arrangement Component: The Arrangement component provides information about relationships between objects and its environment description (map, floor plan).

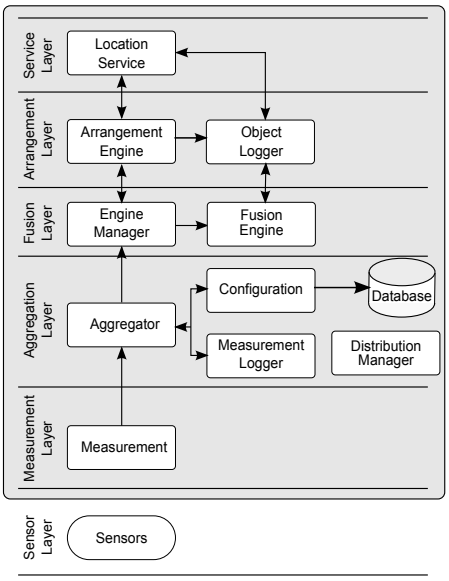

Fig. 2. MapUme Architecture which is developed based on layered software model for localisation system

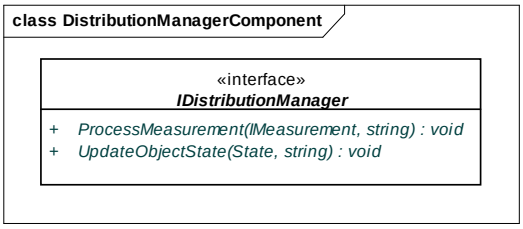

Fig. 3. Interface definition of Distribution Manager Component for supporting distributed data processing

5) Location Service Component: MapUme offers location service allowing other applications to utilise the location information available in the middleware. Location service gets location information from both Object Logger and Arrangement component.

B. Implementation of Scalability and Distributed Data Processing

The middleware can operate in distributed mode to support scalability and distributed data processing. The Distribution Manager is responsible for controlling communication and data distribution among several involved computing machines. It includes distributing the measurement data and updating object position based on result from fusion engine from remote machine. The communication among MapUme middleware in different machine is developed using Windows Communication Foundation (WCF) part of .NET framework which provide application programming interface for developing service oriented applications.

Figure 3 shows interface definitions of Distribution Manager component which will be used for distributed data processing. Distribution Manager provides two important interfaces: the

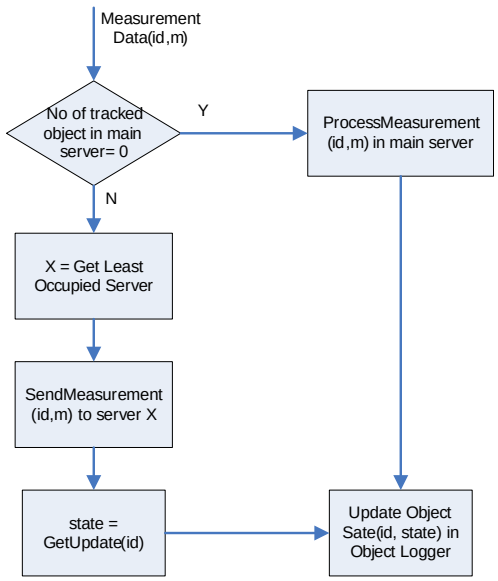

Fig. 4. Flowchart shows how to manage distributed data processing

first is for sending measurement data from main server to remote computing machine, and the second is for updating object state (position, direction, etc) after obtaining estimate position from fusion engine component. All object states are stored in Object Logger component of the main server.

When MapUme middleware operates in distributed mode, Distribution Manager employs a simple algorithm as depicted in the flowchart in Figure 4 to determine into which remote machine (server) a new sensor measurement data will be forwarded. The algorithm will look for least occupied server to determine the next destination to process measurement data.

IV. Middleware Evaluation

This section describes evaluation of the MapUme middleware with particular emphasise on scalability and distributed data processing aspect - the thrust of this paper. The test-bed consists of eight access points installed evenly distributed in the two-story building with dimension 25 x 70 meters. Smart phone equipped with WiFi sensor was used as mobile client which communicate with MapUme through wireless network. In this testbed the MapUme middleware is used to developed a WiFi based localisation system. Several tools have also been built to support the experiment. Mobile fingerprint tool is developed using Qt language and deployed in the windows mobile based smart phone. The RSSI fingerprint collected with this site survey tool then stored in the database. Environment tool is developed to configure location middleware and create configuration file in XML format.

A. Processing Time in Standalone Mode

The evaluation in stand alone mode is done by running MapUme middleware implementation only in one machine in contrast to distributed mode which involves several machines. The processing time is the time required by middleware to process measurement data and calculated from the time

TABLE I
Comparison of processing time in stand-alone mode using different computing machines and different number of particles

Number of Particles	Server I Dual Core 2 x 2GHz, 2GB Ram (milliseconds)	Server II Quad Core 4 x 2.1GHz, 4GB Ram (milliseconds)
50	544	197
100	556	209
200	567	229
500	689	286
1000	1183	528
2000	1757	664

receiving measurement data until the location engine finishes to estimate object position. Table I shows the comparison of processing time needed using different computing machine.

B. Scalability and Distributed Data Processing

For evaluation purpose, offline measurement data was collected by a walk in the building with smart phone device. A simulator opened the recorded data and then sent them to a WiFi localisation system which implements MapUme middleware.

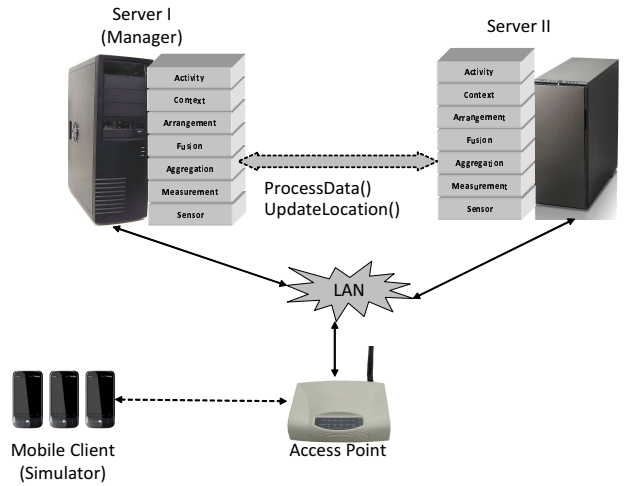

Fig. 5. Configuration for scalability and distributed data processing test. Server I = Dual Core 2 x 2GHz, 2GB Ram. Server II = quad core 4 x 2.1GHz, 4GB Ram. Server I acts as a manager while server II as a member.

Figure 5 shows the hardware configuration used to evaluate scalability and distributed data processing. This configuration involves two computing servers where each server runs MapUme GUI which implements the middleware. Server I runs as a manager which is responsible for managing distribution of measurement data while Server II run as a member. Distribution Manager component controls distributed data processing including serialisation of measurement data and location update from all member servers to the manager server. The communication between these servers uses WCF

TABLE II
CPU USAGE OF MAPUME IN DISTRIBUTED MODE

Tracked Objects	Server I 2x2GHz 2GB RAM		Server II 4x2.1GHz 4GB RAM	
	No of Thread	CPU (%)	No of Thread	CPU (%)
1	1	33.80	0	n/a
2	1	37.37	1	10.92
5	3	77.02	2	18.86
10	5	80.28	5	24.59
20	10	82.09	10	23.73
40	20	81.26	20	23.35
80	40	82.67	40	23.55

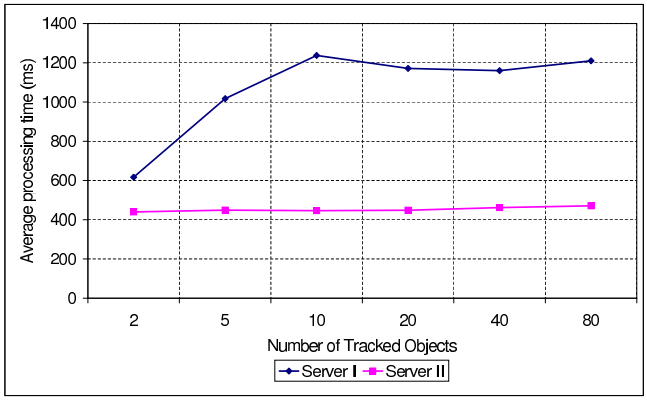

Fig. 6. Average processing time of MapUme middleware in distributed mode involves 2 servers with different number of tracked objects. Number of measurement data = 222. Time interval between measurement data = 1 second. Engine = particle filter with 500 particles.

framework for building service oriented application provided by .NET.

1) CPU Usage: Table II shows CPU usage of each MapUme server with different number of tracked object. Currently, measurement data are distributed evenly into each server which makes computation load of each server uneven since the processing speed of CPU is different. Each thread is assigned to track one object. The ongoing work will improve the distribution of the load by introducing load balancing algorithm. It will take consideration the processing capability of each server in the load distribution process.

2) Processing Time and Latency: The number of tracked object affects average processing time of measurement. The bigger number of tracked object, the bigger processing time needed for each thread to fuse a measurement data. Figure 6 illustrates average of processing time of one measurement data of the MapUme middleware with different load (number of tracked objects). The distribution of measurement data from one server to another server introduces a network latency 3.436 milliseconds. This latency is still reasonable since it is relatively small compare with average processing time.

Table III shows total and average processing time required to fuse all 222 measurement data from simulator with different number of tracked object. These results is calculated when the middleware is running in distributed mode involving both of server I and II. The average processing time each measurement converges to a number between 366 - 392 milliseconds as the number of tracked object increasing. This is much faster compare with average processing time when number of tracked object smaller. This is because in distributed mode, the load are distributed to all participating computing machines.

TABLE III
TOTAL PROCESSING TIME ON DISTRIBUTED MODE, MEASUREMENT = 222, PARTICLE = 500

Tracked Object	Total Processing Time (minutes)	Average Processing Time Each Measurement (seconds)
1	4.233	1.144
2	4.583	0.619
5	7.650	0.414
10	14.517	0.392
20	29.100	0.393
40	54.100	0.366
80	115.033	0.388

V. CONCLUSION

The paper has presented and evaluated the MapUme, a scalable middleware for localisation systems. The scalability feature enables scaling up localisation system to facilitate tracking a large number of object with support of distributed data processing. Service oriented architecture approach is used to implement distributed data processing since this feature promotes scalability by allowing a main server to forward requests to multiple service instances without the knowledge of the service client. Future research work will focus on usability and extensibility of MapUme middleware.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of EI under the Proof of Concept PC/2007/0073 and the EC FP7 ICT LocON project in funding the work reported in this paper.

REFERENCES

[1] A. LaMarca, Y. Chawathe, and S. Consolvo, "Place lab: Device positioning using radio beacons in the wild," *Pervasive*, 2005.
[2] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. Mickunas, "MiddleWhere: a middleware for location awareness in ubiquitous computing applications," in *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Springer, New York, 2004.
[3] A. Cupper and G. Treu, "Trax: A device-centric middleware framework for location-based services," *IEEE Communications Magazine*, no. September, pp. 114–120, 2006.
[4] D. Fritsch and S. Volz, "Nexus - the mobile GIS-environment," *SympoTIC'03. Joint 1st Workshop on Mobile Future and Symposium on Trends in Communications*, pp. 1–4, 2003.
[5] I. Satoh, "A location model for smart environments," *Pervasive and Mobile Computing*, vol. 3, no. 2, pp. 158–179, Mar. 2007.
[6] W. Najib, M. Klepal, and D. Pesch, "A Software Development Model for Localization Systems," in *POCA - Positioning and Context Awareness Conference*, Antwerp, 2009.
[7] Widyawan, M. Klepal, and D. Pesch, "A Bayesian Approach for RF-Based Indoor Localisation," in *IEEE ISWCS*, Trondheim, Norway, 2007.