Improving Services using Mobile Agents-based IoT in a Smart City

Pulkit Verma, Mayank Gupta, Tuhin Bhattacharya, and Pradip K. Das
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati, India
{v.pulkit, mayank.2013, b.tuhin, pkdas}@iitg.ernet.in

***Abstract*— Modern-day devices like smart-phones, tablets, televisions etc. possess very powerful processors and huge storage capacities compared to what were available a few years ago. Most of these devices are also connected to the Internet. However, the full capabilities of these devices are not fully harnessed and thus, they are not as intelligent as they could be. These devices, together with the Internet, can be used as "Internet of Things" where each device can be both producer and consumer of information. This framework is realizable in a real dynamic system if there is an intelligent distributed layer above it which can cater to services of all heterogeneous devices as required.**

The existing solutions to this problem are either too hardware dependent, or too abstract. In this paper we present a concept of this layer using mobile agents which makes the system flexible and dynamically adaptable. This layer has been deployed using a publicly available Prolog-based mobile agent emulator (however, any other mobile agent framework can also be used). The proposed approach is capable of updating information like availability and usability of services dynamically. It also has speech processing modules to provide solutions using voice-based commands and prompts. The prototype is scalable and robust to partial network failures. The implementation details and performance analysis of this work are reported and discussed. This framework can be used to deploy systems which can enable people to search for services like health facilities, food services, transportation, law and order using a common interface including voice commands.

***Keywords*—Mobile Agents, Distributed System, Services, Smart City, IoT.**

I. Introduction

Nowadays there are numerous kinds of smart devices available to the common man, each of which has computing powers which were not imaginable a decade ago. But the infrastructure and network on which these devices run and communicate with each other have not evolved at an equal pace. To overcome this gap, the idea of devices interacting with each other was introduced some time back.

DOI: https://doi.org/10.1109/IC3I.2014.7019766

The main challenge in such an arrangement is to develop a framework that makes the system scalable, robust and fault tolerant. Any new device should be seamlessly addable to the existing network. Even if a part of the network becomes dead, the remaining live nodes will continue to work. New services can be added by simply adding another service type.

In this proposed framework various kinds of heterogeneous devices could interact with each other on a network to provide the services to the user. These services may include day-to-day tasks like controlling home appliances remotely and intelligently or navigating around a new city or some other advanced usage.

In this paper we present such a framework that uses mobile agents. We also demonstrate the practicality of the whole concept by implementing a system based on this framework. The mobile agents run on a platform that is totally distributed and fault tolerant, so as to make the overall system more robust. The characteristics of decentralization and load distribution are also included in the system.

The proposed framework also includes execution of services using voice commands and voice prompts. This framework finds major applications in a smart city scenario where people new to the city can use services without taking help from others. This system is shown to be scalable and robust, and can cater to a large population size.

II. Related Work

Originally the concept of smart devices interacting with each other to form a large network was introduced by Mark Weiser [13] in 1991 as a possible peek into the future. Since then many scientists around the world have been trying to put forward a model to be used for realization of this concept.

There are two main design paradigms used in Internet of Things. One of those focuses on "Internet" part of IoT, and the other which focuses on the "Things" part. Approaches focusing on "Internet" are the ones where new protocols are being developed for catering the needs of IoT. Authentication and security were also discussed as possible challenges in using this approach.

Later a well-known standard Universal Plug and Play (UPnP) for global devices interaction was introduced. It was a "middleware platform for distributed semantic services".

Sensors and actuators were also tried to be integrated into UPnP, but it suffered a major security problem as it did not have any authentication method [6].

"UPnP sensor network management architecture" was proposed which used Bridge of Sensors (BOSS) for handling non-UPnP sensor nodes. But UPnP uses a large number of protocols and hence is not suitable for use with embedded devices [12].

The authentication problem of UPnP was addressed by Devices Profile for Web Services (DPWS), but since the protocols used are similar to that of UPnP, it also cannot be used with small devices. Also interoperability problems will be caused by the device discovery mechanism of DPWS [15].

As far as the "Things" part is concerned, IOT came into reality with Radio-Frequency Identification (RFID) tags.

Most of the work related to realization of IoT is done in the field of embedded systems where each smart device has inherent networking capabilities and various such devices are connected to a circuit which can then interact with other such circuits over the Internet. Technologies like RFID and Near Field Communication (NFC) have become integral part of such implementations where the automatic identification of devices and their interconnection is achieved using these [1], [10].

By using Near Field Communications (NFC) and Wireless Sensor and Actuator Networks (WSAN) together with RFID tags, "things" can also perform tasks. Embedding electronics e.g., sensors into everyday physical things makes them "intelligent" and they can be made capable of executing tasks autonomously [1].

Now in the current trend of "Smart City" initiative [11] these "intelligent" things can play a big role. Neirotti et al. [11] gave a general outlook into this concept of smart cities and provided guidelines for policy makers and city managers to define the Smart City strategy. They also showed the various recent work done in fields like energy, environment, healthcare, public security, etc. to be incorporated into these smart cities.

Most of the recent work is now based on the use of Wireless Sensor Networks (WSN) which consists of several autonomous independent sensors which monitor the environment to collect data and send it to some designated place/repository. The use of WSAN [3] to realize IoT was also explored, wherein the requirements of embedded applications on industrial level were also considered.

Recently researchers have tried to explore the use of mobile agents in IoT as mobile agents provide flexibility and decentralization to the system. Also they reduce the latency of the network and the network load.

Mobile agents were initially used to integrate WSN and IoT and were found to distribute load properly, exploit the "locality in communication and system resources" and facilitate "agent-based adaptable service composition" [8].

A mobile agent framework for IoT was also proposed to help heterogeneous devices in completing the tasks cooperatively [4]. It used "Pheromone-Conscientious based mobile agent migration mechanism" (PherCon) proposed by Godfrey and Nair [5] as it provided partial bi-directional search.

Mobile agents based on web technologies were also proposed which used HTML5 applications as agents to collect data from different locations [7]. This particular implementation lacked autonomous and automatic movement of agents and thus could be improved.

III. Methodology

We have implemented a prototype that uses the devices connected to the Internet and are present in a city, to provide the basic backbone services, i.e., police, hospitals & ambulances, transport services, and fire brigades. The police stations, hospitals, transport services, and fire stations are the service providing nodes. The other nodes are the citizens that use their smart phones or tablets that have client applications installed that can communicate with the service providers.

The client application sends a request, which contains a request-type along with other details like the GPS location, an image (of the building in case of a fire emergency) and the details of the user and the device. The request type can be any one of 'Medical', 'Police', 'Transport', or 'Fire'.

The client knows the addresses of all the service providers present in the city (the app can download these addresses over the Internet initially). This information cannot be stored offline as some of the data is dynamic, e.g., the availability of ambulances, etc.

The client's request goes to one of the service providers, irrespective of the request type. The service provider is chosen randomly, so as to efficiently balance the load of handling the requests [2]. Also due to this, the system continues to work irrespective of the number of service providers that are alive. On receiving such a request, the service provider acknowledges the client. If the client does not receive an acknowledgement within a certain time frame, it resends the request (by again selecting a random service provider). Thus, eventually, the client will find a service provider that acknowledges its request.

On receiving a request, the service provider checks the type of the request and forwards the request to the appropriate service provider. Each service provider, independent of its type, maintains a list of all the other service providers, including details like its GPS location, number of resources (like ambulances, fire brigades, police force) available. Considering the fact that the total number and addresses of these nodes are fixed, so this particular operation is not very costly, and any simple update procedure can suffice. On receiving a request, it matches this request against all the servers of that type available, and selects the most appropriate one depending on distance and resources and forwards the request. An acknowledgement is also sent to the client.

The service providers need to be able to distinguish between a request sent by a client and that forwarded by another service provider. In the first case, they need to find the most appropriate provider and forward the request; while in the second case, they just need to serve the request. Thus, for each service, two requests are generated: Type1 and Type2. Type 2 request may be a local request, indicating that the Type 1

request landed on the correct service provider. Thus, there are two acknowledgements for each service request: Ack1 informing the client that its request has been received and Ack2 telling that a response has been initiated.

A. Architecture

The proposed system works above the TCP layer. The requests and responses can be sent as payloads to the TCP data. Each device is uniquely identified by its IP address. The system uses mobile agents for communication.

Typhon: We have used Typhon mobile agent emulator to develop the prototype. It works over the TCP layer and is used to transfer messages and payloads between nodes. It is totally distributed and has been tested for robustness [9].

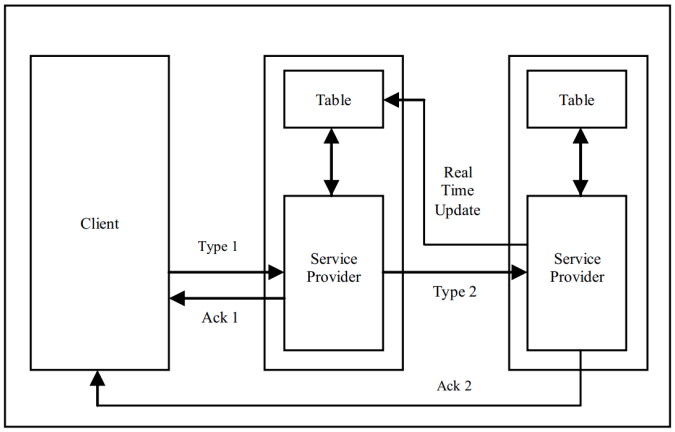

Fig. 1. *A typical request for a service with acknowledgement*

As shown in Fig. 1, each service provider maintains a table, which contains all the relevant information about all other service providers: Their IP addresses, number of ambulances & doctors, fire brigades, police force available and so on. These details help it to find the most appropriate node that can serve the incoming Type 1 request. When the request is forwarded as a Type 2 request and the destined service provider starts to serve the request, some of the resources may need to be updated in all the tables. Thus, a mobile agent is released with the new, updated values to all the other service providers. These, on receiving the agent, update the values in their local tables.

1) Addition of a new node: Whenever a new client node wants to start using the service, it installs the client application on his smartphone, and the URLs/ IP addresses of all the service providers in the city are configured on the client. Now, whenever the user wants to request, he/ she opens the client, and sends the appropriate (Type 1) request.

On the other hand, if a new service provider (a new police station, hospital, file station, transportation or other services like food, gas stations, etc.), joins the network, all service provider tables need to be updated with its address and other details and the clients need to be notified about a new service provider being available.

2) How is it useful?: The system is totally decentralized. Experiments show that it works well and is robust against network failures and congestion. Even if a part of the network fails, the remaining nodes can still work without any glitches. Also, other services can be added without any change in the architecture. The system can be made totally hands-free using voice commands and prompts.

B. Speech Processing Module

Though a text-based input is sufficient, the user may not always be in a position to operate the device using his hands. For example, while driving, for differently abled people, etc. So, it is easier to have a speech control subsystem in place that can operate via voice commands.

C. Sending Additional Information

A simple text message may not always be sufficient for the service providers to act upon in the right manner. Some additional information may be needed as well. For example, if there is a fire somewhere, a cell phone camera photo of the location of the fire may be very crucial. Also, in case of an injury, a wav recording or an image/photo may serve well.

However, a multimedia message consisting of a wav file or an image may be quite large in size. Also, it may need to be compressed. Thus, it may take more time compared to a simple text message. So it is not feasible to send them as a single message.

The text message should go first, followed by the multimedia message which is sent as a payload. Also, since different requests may be sent to different service providers as Type 2 requests, we need to inform the server that a particular image or sound clip belongs to a previously routed Type 1 request, and must be sent to the same destination. A key must be maintained for this purpose. Whenever a Type 2 request is sent by one service provider to the other, it keeps an entry of the key-service provider for some time. When the multimedia message arrives with that key, it is looked up for in the table and sent to the same service provider. This key can be a combination of the client id, GPS location and request timestamp.

IV. Experimental Setup

For our experiment, we have used LPA Prolog and Typhon-A Mobile Agents Framework for Real World Emulation in Prolog [9].

As initial setup, there are several instances of Prolog running, denoting Hospital, Police Station, Cab Services (transportation), and Fire Brigade. These all are service providers.

When the client requests for any of the services, the type of service provider and the location of the client, i.e., the GPS coordinates are embedded as the payload of a mobile agent and sent to any of the service providers chosen at random. Now every service provider has the list of all the service providers available in the city. Based on the request type and location of the client, the service provider that got the request sends the mobile agent to the appropriate service provider. When

the mobile agent moves to that service provider, it takes the necessary action and a mobile agent is sent back to the client as an acknowledgement.

The Typhon emulator that we have used is designed to work only over a local area network. However, to deploy the same system over the Internet, any mobile agent framework which supports Internet-level packet routing can be used.

We maintain a number of nodes, each of them being a Prolog prompt, running on different IP address and port combinations. Each of these supports Typhon framework, and can be used to send and receive mobile agents.

Some of these prompts act as clients and the others as service providers. The clients send Type 1 requests to the service providers using their IP addresses and port numbers and get the acknowledgement.

A service provider, on accepting a Type 1 request, finds the correct service provider and raises a Type 2 request. The appropriate messages are displayed on the screen. Whenever a message is sent or received, the timestamp also gets displayed (All system clocks are synchronized periodically). We study the performance of the system as the number of nodes is varied.

The response time for a service can be described as the total time elapsed between the issuing of the request by the user and the receiving of the acknowledgement for Type 2 request.

It is expected that the response time will increase not more than linearly with the increase in the number of client nodes. Also, it should decrease as the number of service provider nodes is increased. We plot the graphs between these entities.

In this experiment, along with a text-based input, we have also used a speech-based interface. This was developed using HTK, which is a standard toolkit for speech processing tasks [14].

Since the speech subsystem is used only to collect the input from the user, its accuracy does not influence the performance of the proposed architecture; it just suggests how flexible the system can be. This is helpful in a smart environment, when the person is using his device to send a request.

V. Results

We measure the performance of the system under various loads and network traffic conditions.

Table 1 shows the variation in the service time with increase in number of clients. The number of service providers was equal to 10 and it was kept constant throughout the run. Fig. 2 shows the graphical analysis of the same data. It may be noted that the service time increases as we increase the number of clients. Also, the increase in the service time is approximately linear.

Table 2 shows the different service times with different number of service providers. In this experiment 60 clients were used. Fig. 3 shows the graphical representation of the results. The service time decreases when the number of service providers is increased. This may be due to the fact that as the service providers increase in number, the load tends be distributed across more nodes. This trend is not visible in the case of a single service provider. This may be because there is no need to route the packets anywhere else. So, the Type 2 request is in effect a loopback request causing the service time to decrease.

Hence the increase in service time as we increase the number of clients is compensated by the decrease in service time as we increase the number of service providers.

TABLE I
VARIATION IN SERVICE TIMES TAKEN FOR DIFFERENT NUMBER OF CLIENTS USING 10 SERVICE PROVIDERS

No. of clients	Service Time (milliseconds)		
	Minimum	Maximum	Average
1	235	235	235
5	262	469	388.2
10	423	460	446.4
20	302	662	492.1
40	334	2089	716.5
60	437	2330	952.2
85	295	25699	1442.8

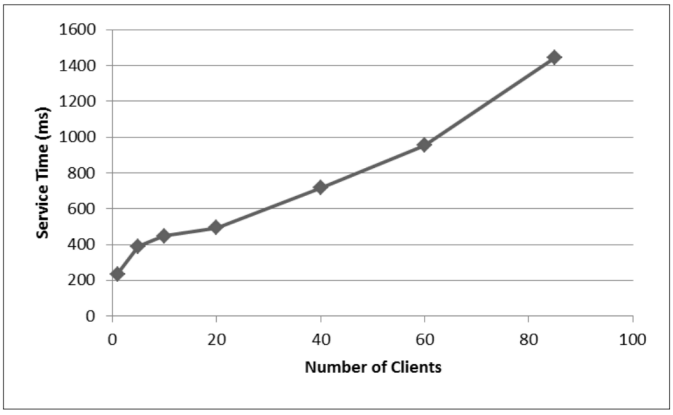

Fig. 2. *Graph for number of clients vs. average service time*

TABLE II
VARIATION IN SERVICE TIMES TAKEN FOR DIFFERENT NUMBER OF SERVICE PROVIDERSUSING 60 CLIENTS

No. of Service Providers	Service Time (milliseconds)		
	Minimum	Maximum	Average
1	517	1696	1008.6
5	516	2985	1332.2
10	585	3115	1389.9
15	577	1953	976.6
20	279	2076	1049.4

The experiments were performed using the local area network of our institute. Thus, it may be argued that the results will not be the same as in a real network or the Internet. However, various real network properties, like traffic and congestion were taken into consideration while conducting the emulation. So, the results reported here should not vary significantly with the real network.

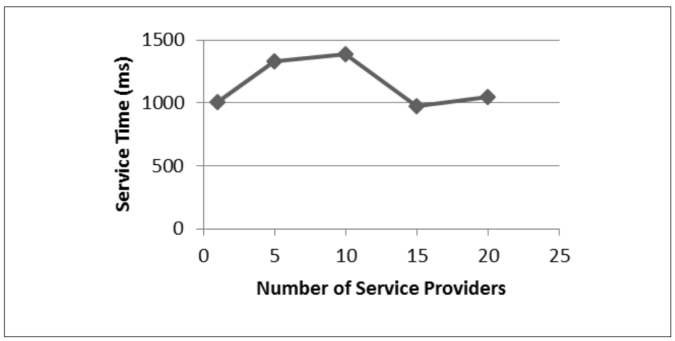

Fig. 3. *Graph for number of number of service providers vs. average service time*

Due to the unavailability of a standard procedure to benchmark different architectures in the domain, comparison between the proposed system and others is not straight forward.

VI. Conclusion

In this paper we proposed a framework for IoT which uses mobile agents for information transfer which in turn provides the services. The proposed method provides better service time for requests and increases linearly with increase in number of concurrent requests. Also, the number of service providers can be increased or decreased dynamically with no need to modify the existing network. Even in case of a failure of some part of the network, the remaining services are not affected. New services and new clients can be added seamlessly. The system is totally distributed and decentralized.

It was implemented over the institute local area network instead of on the Internet. Though the prototyping was done on a small scale as compared to an actual city, the concept seems to be very promising and scalable.

We used only four types of services, but it can be extended to almost any type of services, e.g., gas stations, location of ATM machines, restaurants, entertainment centers and so on.

Speech processing has been used effectively to provide voice commands and voice prompts to give end user a hands free solution. It helps people who cannot readily use the conventional text based input methods because of disabilities or scenarios such as driving.

To test the proposed concept, a prototype for a smart city has been developed. However, it can be used in a number of other domains as well, such as smart homes, healthcare, university campuses, etc. The only requirement is availability of a network and several smart devices.

The scenarios for updating the system like adding new service providers or adding new services to the list of services provided by a service provider can also be implemented with the help of agents or a new protocol can be developed for it. Since these events have low frequency hence they will not have much effect on the performance of the system.

The speech processing module can be replaced with the built-in speech processing APIs available in the devices. This will allow users to connect to the system using the voice interface that he/she is already familiar with.

VII. Future Work

The system can be extended for controlling devices remotely. The identification of devices in case of dynamic IP addresses is a big challenge for a distributed system that doesn't use a unique device ID for each device.

If multiple requests are received from a location, they may be for the same incident or different ones. This will require further investigation based on the type of requests from different sources.

Real-time services like traffic updates can also be implemented using a slightly complex version of this design.

References

[1] L. Atzori, A. Iera, G. Morabito, 2010. The Internet of Things: A survey. Computer Networks. 54(15), 2787–2805.

[2] M. Bramson, Yi Lu, and B. Prabhakar, 2010. Randomized Load Balancing with General Service Time Distributions. SIGMETRICS Perform. Eval. Rev. 38 (1), 275-286.

[3] D. De Guglielmo, G. Anastasi and A. Seghetti, 2014. From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things. Advances onto the Internet of Things. 135-152.

[4] W.W. Godfrey, S.S. Jha and S.B. Nair, On a Mobile Agent Framework for an Internet of Things. Communication Systems and Network Technologies (CSNT), 2013 International Conference on. 345-350.

[5] W.W. Godfrey and S.B. Nair, 2011. A Bio-inspired Technique for Servicing Networked Robots. International Journal of Rapid Manufacturing, Inderscience Enterprises Ltd. 4(2), 258-279.

[6] Y. Gsottberger, X. Shi, G. Stromberg, T. Sturm and W. Weber, 2004. Embedding Low-Cost Wireless Sensors into Universal Plug and Play Environments. Wireless Sensor Networks. 291-306.

[7] L. Jarvenpaa, M Lintinen, A.L. Mattila, T. Mikkonen, K. Systa and J.P. Voutilainen, 2013. Mobile agents for the Internet of Things. System Theory, Control and Computing (ICSTCC), 2013 17th International Conference. 763-767.

[8] T. Leppanen, Meirong Liu, E. Harjula, A. Ramalingam, J. Ylioja, P. Narhi, J. Riekki and T. Ojala, 2013. Mobile Agents for Integration of Internet of Things and Wireless Sensor Networks. Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on. 14-21, 2013.

[9] J. Matani and S.B. Nair, 2011. Typhon – A Mobile Agents Framework for Real World Emulation in Prolog. Multidisciplinary Trends in Artificial Intelligence. 7080, 261-273.

[10] F. Michahelles, F. Thiesse, A. Schmidt, J. R. Williams, 2007. Pervasive RFID and near field communication technology. IEEE Pervasive Computing 6(3), 94–96 (2007).

[11] P. Neirotti, A. D. Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, 2014. Current trends in Smart City initiatives: Some stylised facts. Cities. 38(0), 25-36.

[12] H. Song, D. Kim, K. Lee and J. Sung, 2005. UPnP Based Sensor Network Management Architecture. Proc. International Conference on Mobile Computing and Ubiquitous Networking.

[13] M. Weiser, 1991. The Computer for the 21st Century. Scientific American. 265(9): 66–75.

[14] S. Young, J. Jansen, J. Odell, D. Ollason and P. Woodland, The HTK Book. Cambridge Univ., 1996.

[15] E. Zeeb, A. Bobek, H. Bohn and F. Golatowski, 2007. Lessons learned from implementing the Devices Profile for Web Services. Proceedings of the 2007 Inaugural IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2007). 229–232.