The Community Stack: Concept and Prototype

Tuan Nguyen, Seng W. Loke, Torab Torabi
Department of Computer Science and Computer Engineering, La Trobe University, Australia
Email: {s.loke,t.torabi}@latrobe.edu.au, at6nguyen@students.latrobe.edu.au

Abstract:
Location awareness and mobility are main factors in pervasive computing. There has been a lot of work focusing on developing location aware applications. Generalizing from location awareness is the notion of mobile virtual communities which users enter or leave, analogous to pushing or popping items on or off a stack. The multi-agent paradigm is suitable for modeling distributed systems and can help reduce the complexity of system design. In this paper, we introduce a prototype for the community stack concept that is based on the multi-agent paradigm.

Keywords: virtual community, community stack, mobile computing, multi-agent, context-aware computing.

1. Introduction
Technologies of growing popularity include the Internet, computer-based communication such as IRC, WIFI, and wireless devices like mobile phones, and PDAs (personal digital assistants). Peer to peer networking and pervasive computing are changing the ways in which people organize complexity of system design. In this paper, we introduce a prototype for the community stack concept that is based on the multi-agent paradigm.

This paper aims to prototype the concept of the community stack using the multi-agent paradigm in virtual communities. This paper is organized as follows. In section 2, we introduce the concept of the community stack. In section 3, we describe a case study in a shopping center that can illustrate the concept of the community stack. Section 4 explains the use of a multi-agent architecture to implement the concept. The next section shows the prototype of the system. Thereafter, we briefly discuss related work and sketch out future developments before concluding.

2. Community Stack
Users may interact with multiple PBE-communities. A PBE-community may overlap with or be located within another community. For example, users go to a shopping centre; they can be both in the shopping centre community and the supermarket community within that shopping centre at the same time. Users can dynamically join communities to use their services when they are inside the communities. After stepping out of the zone, they can automatically leave the communities. Thus, effectively, the user is within a stack of PBE-communities which changes with the user’s movements into and out of communities. The stack structure is implied by the nesting of one community within another area, corresponding to simple geographical containment or overlap. There are different types of interactions between the user and a PBE-community [2]:

- One user – one device interact with one PBE-community
- One user – many devices interact with one PBE-community
- One user – many devices interact with many PBE-communities
- Many users – many devices interact with many communities

In each type of interaction, we need a different kind of protocol for users to communicate with the communities. In this paper, we introduce the joining/leaving protocol that can apply for all types of interactions between users and communities.

3. A Case Study
3.1 Scenario
To present the idea of the community stack, we consider an application, where a person Jack with three digital devices enters a community within a community (e.g., entering the Myers departmental store community, after entering the shopping centre community (e.g., Chadstone)). He can travel around the shops, supermarket and finally leave.
There are three devices on the user – PDA, watch and jacket with embedded computers inside, all connected in a PAN using Bluetooth technology (which we imagine will happen in the future to different extents).

The PDA is the master device in the PAN. It is involved in managing other devices in the PAN, doing joining/leaving community activities. The PDA can also receive advertising from shops. When Jack is in the shopping center, he wants to buy cheese, the PDA will create a buying cheese proposal message and send that to a (pre-configured) server, so that the shopping centre software agent can help him find a trader with the best price.

The jacket receives a broadcast of trendy colors (assuming that the jacket can change colors) and digital images which it can display (as a fashion thing).

The watch interacts with some servers of the community to synchronize time and downloads a list of events in the area adding them to its alarm facility.

The idea is that after joining a community, the Head device (i.e. in our example, a PDA) tells the other devices that they can also start interacting with various software agents in the community performing the interactions such as those mentioned above.

The PDA also acts as a proxy agent, as the watch and the jacket do not have enough computation power to host an agent. The watch and jacket computers can send and receive messages to or from the PDA via the Bluetooth network.

3.2 Multi-agent system solution

An agent based system approach was selected for developing the infrastructure. Intelligent agents and multi-agents system have become a frequent computing archetype in the field of pervasive computing. One reason is that agents can reduce the complexity of application domains by delegation of tasks. In addition, agents are very suitable for distributed environments [15]. According to [16], JADE/LEAP is a reliable and stable framework for multi-agent applications. So we use the JADE/LEAP framework to implement the infrastructure.

The shopping center can be modeled like in figure 2. In this example, there are five supermarkets in the Chadstone community (CS1): Myer Supermarket (CS2), David Jones Supermarket (CS3), Target Supermarket (CS4), Coles Supermarket (CS5) and Bi-Lo Supermarket (CS6). This scenario illustrates the idea of communities within the communities.

3.3 Connectivity

There are two types of connectivity in this case study:

- Bluetooth connection: this is for devices on the user to connect to each other. This forms a personal area network on the user.
- Wi-Fi connection between PDA and Community Servers (CS): the PDA interacts with agents in the community via a wireless network.

4. The Application Architecture

Based on the scenario, we design a multi-agent interaction model as illustrated in figure 3. To describe how agents interact with each other. Each agent in the system plays a different role and uses different protocols to communicate.
of agents: position agent, community agents, and advertising agents. The HEAD will ask the position agent where it is over an appropriate positioning technology. According to its physical location and the communities induced by being at a location, the HEAD will join or leave communities. When the HEAD joins a community, the community agent will inform the advertisement agent that a new customer has come, and the advertisement agent will then send advertisement information to the HEAD.

**Community Server agent** - (CS, x=1…n) this agent is a representative for a PBE-community, e.g., for the PBE-community of a shopping center or a supermarket. It communicates with users from joining to leaving the community. The community server agent can also communicate with other agents, e.g., with shop agents to get their price lists. Finally, the community server agent can talk to the advertisement agent whenever a new customer has joined the community for broadcasting the advertisement information.

**Position agent** – Based on the user/customer’s physical location, the position agent will map the customer to a logical position, i.e. the boundary of some PBE-community (if any). In this prototype, the position agent simulates the users’ positions as they move around the shopping center.

The arrows labelled A, B, and C in figure 3 are the protocols that the HEAD uses to communicate with other agents. A is an advertise protocol. B is the Join/Leave protocol, C is a position information acquisition protocol.

The remaining arrows D, E, F and G are for message exchange between the other agents. D shows the information that the community server agent sends to the advertisement agent to tell it that one user/customer has stepped into the community. E illustrates shop agents sending their locations to help the position agent be aware of them.

![Figure 4. A mapping between physical position and logical position](image)

When the user is in a physical position; for example, people at the position P0 ask for “What communities in this area ?” The position agent will return information that the user is standing within the communities by mapping the physical position to the boundaries of the communities which it stores. The results return to the customer is the name of the community server agent for the community. When the customer knows the agent name of the community server (CS), s/he can start to join the community by talk to the CS agent via the joining protocol.

**Advertisement agent** - To provide services in a formal way, the advertisement agent have to follow a protocol. This can help customers avoid mass advertisement information, flooding his/her device. The advertisement agent also receives information from shops or other communities to forward as advertisements.

**Shop agent or Supermarket agent** - These agents represent shops and supermarkets within the community of the shopping center. They also update advertisement information with the advertisement agent and interact with user agents when required (e.g., for transactions, etc).

5. **Protocols**

5.1 **Protocols**

To implement the system, we design four protocols for agents to communicate. In the sequence diagrams below, the HEAD agent on the PDA device or HEAD device is a customer agent.

5.1.1 **Joining Protocol**

Before the user agent on the PDA (or HEAD device) can communicate with the community, it has to join the community to get the information. The joining protocol shown in figure 5 is for the HEAD Agent and Community Server Agent when the customer enters within the boundary of the community. When the user enters a community zone, the HEAD will send a HELLO message to the CS-Agent (Community Server Agent) where the CSAgent name is received via the position protocol. The CSAgent will send the message INTRODUCE to HEAD Agent. HEAD Agent answers CSAgent with the JOINPROPOSE message. CSAgent will decide whether to REJECT or ACCEPT the HEAD Agent’s request to join. CSAgent will reject the HEAD Agent if the community is over crowded or it is closing time; otherwise, CSAgent will accept the HEAD Agent to join.

![Figure 5. Join protocol](image)

5.1.2 **Leaving Protocol**

Figure 6 shows the leaving protocol; this protocol is used by the customer agent (HEAD Agent) and Community Server Agent when the customer leaves a community (by issuing a
When the user leaves the community, s/he has to inform the community. This to avoid the other agents in the community trying to contact the HEAD agent when it has left the community or is out of the community zone. To perform the leaving protocol, HEAD Agent simply sends GOODBYE to CSAgent to inform that it is leaving the community. The when the CSAgent receives the GOODBYE message from HEAD Agent, it also replies with GOODBYE message and notice that the user has left the community. Application-specific or community-specific "housekeeping" operations might be performed on the CSAgent side or the HEAD Agent side, such as saving state or recording this event.

![Figure 6. Leaving Protocol](image)

5.1.3 Position Protocol

Figure 7 shows the position protocol. The protocol provides a mapping operation between physical locations and logical locations (or community zones). In this prototype, the user agent asks the position agent periodically to send the community zone information to help the user agent know where it is, or which community(-ies) it is currently located within. When the HEAD Agent wants to know its logical position, it first starts sending the WHEREAMI message including its physical location to the Position Agent. The Position Agent, based on the physical location of the user, will determine that the user is standing in certain community(-ies) and then returns the list of these communities to the HEAD Agent. The community list is a list of CSAgent names, corresponding to the communities on the list.

![Figure 7. Position protocol](image)

For example, in our case study, if the user is standing in the MYER supermarket, the list will be CS1, CS2 because CS2 is within CS1.

5.1.4 Advertisement Protocol

To avoid the user/customer being flooded by a mass of advertising information, the advertiser has to follow the advertise procedure to supply the information.

![Figure 8. Advertise protocol](image)

Once the user is within a community, HEAD Agent can receive the advertising information from other shop agents within the community. The advertise protocol works as follow: advertisement agent will send a message to the HEAD Agent of the user. If the user is willing to receive the advertising information, s/he will reply with a YES message and then DATA from the Advertisement Agent will be sent to the HEAD Agent. The user can choose never to receive advertisement information by replying with NEVER or NOTTHISTIME for a temporary no.

5.2 Algorithm for joining/leaving communities

The most important thing of this application is the ability of the HEAD Agent to detect its locations. To implement this prototype, we make use of the TickerBehaviour class of the JADE Agent Behaviour package. The TickerBehaviour class can perform the action within a period of time. In this scenario, the customer agent will ask for its position every 20 seconds. After the customer knows his/her current position, s/he will decide to join/leave communities. In our prototype, we implement joining manually (i.e., the community appears on the user’s device automatically, but the user has to select the community in order to join it) and leaving automatically.

To implement the joining/leaving protocol when the user is moving around the shopping center, we define the following concepts:

Let U be the set of all communities, i.e.

\[ U = \{CS_1, CS_2, CS_3, \ldots CS_n\} \]  

P denotes the set of communities that the user is currently in, given his/her current location, and Q denotes the set of communities the user is in after moving to a new location.

L is a set of communities that the user leaves when moving around the shopping center. So, L is defined as below:

\[ L = P \setminus Q = \{x | x \in P \text{ and } x \notin Q\} \]  

N is a set of new communities that the user joins when moving around the shopping center. Q is defined as follows:

\[ N = Q \setminus P = \{x | x \in Q \text{ and } x \notin P\} \]

The algorithm for the HEAD agent doing the joining or leaving action is then:

```c
if (P \cap Q) != \emptyset {
    /* user change communities */
    L = P \setminus Q;
    N = Q \setminus P;
```
LeaveCommunities(L);
JoinCommunities(N);
} else {
  /* No change in communities */
}

Note that when there is no change in position, the HEAD Agent also has to compute the P \( \cap \) Q. We can avoid this by executing this algorithm just when a change in user position is detected.

5.3 Demonstration results
The demonstration scenario is described in Figure 9. The user Jack first goes to the Chadstone shopping center (CS1). In CS1, there are many community servers corresponding to inner communities within the Chadstone community such as MYER(CS2), DAVID JONES(CS3), TARGET(CS4), COLES(CS5) and BILO(CS6).

![Figure 9. User road map.](image)

The HEAD agent on the PDA will perform the join/leave protocol whenever the user steps into or out of the community zones. Say, the user then goes into CS2, CS3, and stops in the middle of CS1. He then decides to buy food, “cheese” for example.

Figure 10 shows the interaction between the customer agent and the community agent. Because this shopping center is very big and complex, the user needs help from the shopping center agent (CS1). He asks CS1 to help him find a best seller with the best price.

![Figure 10. Communication model](image)

The CS1 agent will query supermarkets in the area, for example, COLES and BILO. After the CS1 agent has a result, it will send the result to the customer agent. In this example, BILO is a best seller. Then, Jack goes to CS6 (BILO) to buy the food.

Figures 11 to 13 show the screen shots of the above demonstration scenario.

![Figure 11. First 2 communities to join](image)

When the user stepped into CS1, and CS2, the HEAD agent will display the CS1 agent name on the display for the user. The join action can be done automatically, but for debugging and demonstration purposes, we allow the user to join manually by selecting CS1.

Then, the user steps to position 3, which is an intersection area between CS2 and CS3. The user is now standing in three communities at the same time as shown on the left screen in Figure 12.

![Figure 12. Changing between communities](image)

Then, he leaves CS2, and CS3. At this time, he needs help from CS1 to buy food. He first lists the services provided by CS1, for example, Food and Liquor.

When he makes a request to buy food, the CS1 agent will query all other agents to find the best price and best seller for the customer. Then, the CS1 agent will return the results to the HEAD agent, which will display the results on the customer’s PDA as shown in Figure 13.

![Figure 13. Using services provided by Community Agent](image)
In this example, the CS6 (BILO) is the best seller with the best price so the user moves to the CS6 area to buy food.

6. Related Work

There has been a lot of work on developing location awareness applications based on a multi-agent architecture. The AmbieSense system, implemented at the Oslo International Airport (OSL) [15], is similar to the notion of location context awareness in a community. Once users are in the services area, they can contact agents that can help them use services provided by the community. However, the location had to be determined manually. Furthermore, this system does not consider the problem of a community within a community.

There has also been a lot of work on using Jade-Leap to implement agents in mobile devices such as [5,6]. In our prototyping, we are not composing individual services but sets of services in sets of communities. Users might be present in many communities at the same time. Thus, there is a range of services made available for the user structured into communities, some of which might be useful for different tasks and at different locations.

7. Conclusion & Future work

We have investigated a type of location-based roaming abstraction which we call the community stack. Each community has its own set of services. We have also explained the case for a multi-agent approach for solving the computational aspects of our concept. We have discussed the case of the user being in multiple communities at the same time. The prototype is a proof-of-concept, and its evaluation demonstrates the feasibility of the community stack.

Future work will involve improvements to the performance of the prototype, exploring location detection technologies indoor or outdoor for scalable community stacks (e.g. RFID location detection technology for indoor services or GPS technology for outdoor services).

We can also improve the connectivity and interaction between devices in the PAN network using TCP/IP over Bluetooth wireless ad-hoc network [19]. Then, we can implement the multi-agent system on a wireless ad-hoc network [20].

Our notion of context has been location and communities, and we can generalize the work to involve other aspects of context such as agent communication and coordination. Finally, while our architecture permits a large number of users to join or leave communities, we do not claim completeness. Our contribution is the philosophy and framework of user PAN connected devices within communities of communities.

Acknowledgment

The author Tuan Nguyen is supported by Project 300, (Ho Chi Minh City, Vietnam)

References

[6] A. Moreno & D. Isern, A first step towards providing health-care agent-based services to mobile users, AAMAS’02, July 2002, Bologna, Italy