Exploring Bio-inspired Algorithm for Service Discovery in Pervasive Environments

Xiao Zheng
School of Computer Science
Anhui University of Technology
Maanshan, China
zhx@ahut.edu.cn

Qingyong Wu
School of Computer Science
Anhui University of Technology
Maanshan, China

Abstract—In pervasive computing, one of the key open challenges is the efficient discovery of the device that provides target services. This paper proposes a novel distributed service discovery algorithm in pervasive computing environments. This method is based on the basic principle of social network and bio-inspired intelligence. It exploits a social behavior and ant inspired query routing mechanism, which views query messages as artificial ants, and utilizes pheromone and social utility as routing hints that direct query messages to nodes owning more services. This paper presents generation and update rule of pheromone, routing policy for artificial ants as well. In order to avoid getting into local optimization, the roulette wheel technique and pheromone limitation policy are used in our algorithm. Simulation results show that our method significantly improves the search quality as well as reduces traffic cost compared with classical approaches, which supports high mobility and is suitable for pervasive environments.

Keywords—pervasive computing; service discovery; ant algorithm; social network

I. INTRODUCTION

Pervasive computing [1], which has been extensively studied in recent years, supports using computing and storage resources anywhere and anytime. Pervasive computing has the potential to radically transform the way people interact with computers. It is motivated by the observation that computing and networking technologies are becoming increasingly powerful and affordable, with the result that a wide variety of computing devices can be deployed throughout our living and working spaces[2]. Pervasive computing environments are composed of handheld, wearable, and embedded computers in addition to regular desktops. These are connected by some combination of wireless ad hoc networks and wireless infrastructure-based networks, such as WLANs.

In pervasive computing, one of the key open challenges is the efficient discovery of the device that provides target services. Service discovery is defined as a process enabling networked entities to advertise their services, query about services provided by other entities, and select the most appropriately matched services [3]. In the past, service discovery mainly was addressed in the context of wired networks. However, in the context of pervasive computing, the following new challenges arise. Firstly, node mobility affects service availability. Secondly, frequent intermittent connections of the service provider or the client or intermediate nodes breaking or changing the path and the service selection parameters. Lastly, channel variability leads to significant communication characteristics variability (data rate, delay, etc.)

Research in the area of service discovery in pervasive computing is relatively new. Current methods use broadcast, selective transmitting and caching mechanisms, or combine them to deal with high mobility and intermittent connectivity. None of them take into account combing the social relationship and bio-inspired algorithms. Lindemann etc. [4] proposed an epidemic-based peer-to-peer lookup service, namely passive distributed indexing. Query and response messages are transmitted using local broadcast and query results are cached in participating nodes for later use. Most queries could be resolved locally due to the implicit dissemination of index entries in the network using node mobility. Chakraborty etc. [5] suggested a peer-to-peer caching of service advertisements and group-based service discovery protocol for pervasive environments. Service advertisements are used to disseminate service information, which can help to selectively forward the service request. We [6] recently proposed an ant-like courseware lookup mechanism in pervasive learning environments. Its contribution is exploring social relationship among mobile nodes, and building a socially organized overlay over a mobile ad hoc network.

Our solution of service discovery is inspired from biological social behaviors. Biological phenomenon is used to mark path that is ever passed. Our proposed algorithm uses previous accumulated pheromone, which can be seen as experience, to direct latest query routing. Social relationship shows social members’ common interests and habit, which can be used as heuristics of query delivery.

II. MOTIVATION AND PROBLEM DESCRIPTION

A. Motivation

In reality, human movements are solicited by various social relations (e.g. family, workplace). An individual usually has some social relations with several acquaintances (e.g. family members, friends, colleagues). To look for a piece of information, one often starts by contacting as soon as possible those who believe to be more likely to have a favorable answer. In case of unsuccsses, they may then lean...
on their acquaintances to help find that information. Likewise, these contact nodes in a social network can also be considered as good forwarding candidates in query delivery.

Ant colony algorithm [7], based on behavior of the real ant, is first applied to TSP. In the algorithm, there are some artificial ants deployed on the vertexes in a graph. They imitate real ants’ behaviors to find a shortest path. The core of ACS is using pheromone remaining in paths to indirectly communicate the path information among ants, whereas the pheromone is deposited by former ants passing by the path, which represents some experience knowledge. In the algorithm, the positive feedback of global updating reduces the search scope, and the hidden negative feedback also retains the scope.

Several researchers have attempted to use the notion of ant pheromones for network routing [8], service selection [9] and service discovery [10]. Our work also follows these thoughts.

B. System Model

In a typical pervasive computing environment, services are stored in mobile devices or fixture devices that interconnected by wireless networks. A social network is built according to the social behaviors of the owners of the devices. These social behaviors are interest, acquaintance, mobility, etc. Once a client wants to query some services, the system will deliver query by social behaviors of neighbors, because those who have common interest maybe have the target services most possibly, acquaintances maybe much more know the location of target services, and the mobile neighbor maybe take the query to further geographical location. So there is a social behavior table(SBT) in each device of the system. The entry of a SBT consists of contact and social attributions. Contacts are acquaintances of the owner of the device. Social attributions include interest, frequency and mobility. Interest is contact’s hobby. Frequency represents the meeting frequency with contact. Mobility represents the movement area of the node.

C. Problem Description

A pervasive computing network is defined as $G(V,E,S,F,I)$. $V(t)$ is a set of nodes denote all peers at time $t$ in the network, i.e. $V(t) = \{v_i | i=1,...,n\}$. $E(t)$ is a set of edges denote all links at time $t$, i.e. $E(t) = \{ e_{ij} (v_i, v_j) | v_i, v_j \in V\}$. $S = \{ s_i | i=1,...,m\}$ is a service set denote all shared services in $G$. $F$ is a function $F: V \rightarrow P(S)$ that maps each node $v \in V$ to a subset of $S$, where $P(S)$ is a power set of $S$. The network dynamism implies that $V$ and $E$ evolve over time as nodes can be switched off or out of transmission range.

The problem is that given such $G$ and $s' \in S$, how to solve $V'$ subject to $V' = \{v| s' \in F(v) \cup v \in V\}$. In a real pervasive computing environment, nobody knows the global information of available services, and there is not any directory of whole services, still not any position information of services just like in a DHT-based overlay. Hence the algorithm must traverse nodes in terms of a certain policy, i.e. from a source node, along edges to reach some nodes, and query the target service $s'$ at reached nodes. A tradeoff must be made between recall and cost, e.g. response time, load. A well-performed algorithm should strike a best recall in a desired response time, and simultaneously make use of fewer query messages, to lower the load in the network as possible.

III. ALGORITHM DESIGN

Our discovery algorithm is informally described as follows. For each service discovery request, $n$ query messages are generated. A query message is corresponding to an artificial ant. The artificial ant selects a node from its local neighbor table as its next hop in terms of a routing rule, and deposits pheromone on its passing path. The routing rule considers the amount of pheromone on the path and node’s social behaviors. When finding the target service on a certain node, it will update the pheromone around the node. Moreover, the pheromone remaining on the path will volatilize in period. The pheromone is also limited in a range. Each ant has a life-span. When its life value is zero, the message denoted by the ant will be abandoned. The objective of artificial ants is to find target service as many as possible under certain restricted conditions.

A. Routing rule

Simulating the behavior of ants, our algorithm uses pheromone to direct the routing of query messages. Accordingly, a pheromone table is built in each node, which consists of pheromone on the paths to neighbors. The entry of a pheromone table is composed of a neighbor node and the pheromone on the relative path, i.e. a pair $(n, p)$. $n$ denotes a neighbor node, $p$ denotes pheromone amount, and $(n, p)$ denotes that the amount of pheromone on the path from the local node to neighbor $n$ is $p$. The amount of pheromone represents the success ratio of previous searching at the corresponding node. More pheromone represents higher success ratio. Therefore forwarding query messages to the node having more pheromone may get a higher searching ratio. However, because pheromone represents rather a probability than certain knowledge, a roulette selection algorithm is used to select a node to transfer. In addition, pervasive environment is a mobile network. The neighbor of a node often changes. SBT shows its neighbors’ social behaviors, which could be viewed as a heuristic to help to discover target service.

There are two kinds of social attributes. Some of the attributes such as interest could be positive, i.e., the higher the value, the more possibility having targeted services. Other attributes are negative, i.e., the higher the value, the less possibility. To normalize the attribute values, we scale positive attributes according to (1). For negative attributes, their values are scaled according to (2).

$$\text{SA}_{s_j} = \begin{cases} \frac{sa_{-j} - sa_{-i}^{\min}}{sa_{ij}^{\max} - sa_{ij}^{\min}}, & \text{if } sa_{ij}^{\max} - sa_{ij}^{\min} \neq 0 \\ 1, & \text{if } sa_{ij}^{\max} - sa_{ij}^{\min} = 0 \end{cases}$$
\[
SA_i = \begin{cases} 
\frac{sa_i^{\max} - sa_i^{\min}}{sa_i^{\max} - sa_i^{\min}}, & \text{if } sa_i^{\max} - sa_i^{\min} \neq 0 \\
1, & \text{if } sa_i^{\max} - sa_i^{\min} = 0 
\end{cases}
\]

where \(sa_i\) denotes the value of the \(j\)th social attributes of node \(i\), and \(SA_i\) denotes the normalized value of \(sa_i\). Then, we define the utility function as

\[
U(n_i) = \sum_j (SA_i \times W_j)
\]

where \(W_j \in [0,1]\) and \(\sum W_j = 1\). \(W_j\) is a normalized weight given by users to represent the importance of attribute \(sa_i\).

Then, the probability of artificial ant \(k\) selecting node \(j\) at node \(i\) is defined as

\[
p_k(i,j) = \begin{cases} 
\frac{\text{ph}(i,j)\alpha U(i,j)\beta}{\sum_{j \in J(i) - \text{Tabu}(k)} \text{ph}(i,j)\alpha U(i,j)\beta}, & j \in J(i) - \text{Tabu}(k) \\
0, & j \notin J(i) - \text{Tabu}(k) 
\end{cases}
\]

where \(\text{ph}(i,j)\) denotes the pheromone amount on the path from \(i\) to \(j\), \(U(i,j)\) represents the utility of selecting \(j\), \(\text{Tabu}(k)\) is a set of neighbors of node \(i\), \(\text{Tabu}(k)\) represents the nodes ant \(k\) having passed, and \(\alpha\) and \(\beta\) represent the importance of pheromone and utility respectively.

B. Pheromone Generation and Update

How to generate and update pheromone is a key to influence algorithm performance. Our policy is based on the principle of classic ant algorithm and makes some improvement to satisfy search requirements. In the following cases, pheromone will be generated and updated.

1) As the behavior of real ants, query messages will deposit pheromone on the path passed by. Its generated new pheromone will add to the pheromone remained on the path. Let \(\text{ph}(n)\) denote the pheromone amount on the path from local node \(i\) to neighbor \(n\),

\[
\text{ph}(n) = \alpha \cdot \text{ph}(n) + (1 - \alpha)\Delta \rho,
\]

where \(\alpha \in (0,1)\) and \(\Delta \rho\) is a constant.

2) When having found a target service at node \(n\), the ant would diffuse pheromone to all neighbors. Receiving a message of pheromone update, a neighbor would update pheromone in the corresponding item of its pheromone table.

\[
\text{ph}(n) = \beta \cdot \text{ph}(n) + (1 - \beta)\Delta \rho, m \in J(n)
\]

where \(\beta \in (0,1)\), and \(\Delta \rho\) is a constant. \(m\) belongs to a set of neighbors of nodes \(n\).

3) For each item of the pheromone table, an update will be done in period.

\[
\text{ph}(n) = \rho \cdot \text{ph}(n) \quad \text{with } \rho \in (0,1)
\]

Then if a neighbor always is not visited, its pheromone amount will be closer to zero. Such node may be short of services.

4) If a new node enters the pervasive environment, it will send update messages to all its neighbors. Under this situation, update neighbors' pheromone tables by (6).

In order to control pheromone amount, we set an adaptable pheromone range, which could change the maximum and minimum value with current neighbors’ utilities. The maximum pheromone from node \(i\) to its neighbors is defined as

\[
\text{ph}^{\max}(i) = \begin{cases} 
Q, & \text{at beginning} \\
\text{Utility}^{\max}(i), & \text{otherwise}
\end{cases}
\]

where \(Q\) is a constant, \(\text{Utility}^{\max}(i)\) is the maximum of node \(i\)'s neighbors utility.

Likewise, the minimum pheromone from node \(i\) to its neighbors is defined as

\[
\text{ph}^{\min}(i) = \begin{cases} 
0, & \text{at beginning} \\
\text{Utility}^{\min}(i)/m, & \text{otherwise}
\end{cases}
\]

where integer \(m\) is an adjust factor, \(\text{Utility}^{\min}(i)\) is the minimum of node \(i\)'s neighbors utility.

C. Life-span Control

Every artificial ant has life-span, which could control the number of messages transmitting in the network and insure there are no more messages transferred ceaselessly after a query is over. Our algorithm also uses TTL to control message’s life-span. When an artificial ant is created, its TTL will be set to an initial value. Hereafter at each passing node, the artificial ant’s TTL will be updated. If not found target service, the ant will decrease TTL. If found, nothing done so the ant will visit more nodes. If all neighbors are in the tabu table, the ant’s TTL will be set to zero. So the ant will not be transferred and then be killed. When ant \(k\) at node \(i\), the update rule of its TTL is:

\[
\text{TTL}(k) = \begin{cases} 
\text{TTL}(k) - 1, & r(k) \notin S(i) \\
\text{TTL}(k), & r(k) \in S(i) \\
0, & J(i) \subseteq \text{Tabu}(k)
\end{cases}
\]

where \(r(k)\) represents the target service of ant \(k\) wanting to find. \(S(i)\) is the set of services at node \(i\).

Based on upper discussion, query message routing algorithm is described as follow.

Algorithm 1. query message routing algorithm.
boolean MessageRouting (Message Query, Node node) {
    // if the node ever received this message, delete it.
    if SeenMessage(Query, node) return false;
    // if there is not a target service, the TTL minus 1, otherwise diffuse pheromone to neighbors.
    if (NoFound) Query.TTL--;
    else
        for any nodei in node.neighbour
            UpdatePheromone(node, nodei, ph);
    // if all neighbors have been visited, set TTL to zero.
    if (all node.neighbour in Query.Tabu) Query.TTL = 0;
    // if TTL equals to zero, delete the message.
    if (Query.TTL == 0) return false;
    // Select a neighbor by formula (4), and forward the message to it.
    nextnode = RouletteSelect(node.neighbour);
    ForwardMessage(Query, nextnode);
    // Update the pheromone on the path to the next hop by formula (5).
    Updatepheromone(node, nextnode);
    // Limit pheromone in \([ph_{\text{min}}, ph_{\text{max}}]\). Limitpheromone(node, nextnode);
    return true;
}

IV. SIMULATION AND PERFORMANCE ANALYSIS

We use NS-2[11] to simulate a pervasive mobile environment and primarily compare our algorithm with a simple broadcast-based algorithm and a random walk algorithm [12, 13]. In the former one, node keeps its own queries in a buffer to periodically broadcast them to its neighbors. In the latter one, node does not only keep its own queries but also periodically send them to a random subset of its current neighbors. We did not consider message loss or low layer’s detail in this study.

The main simulation parameters in our experiment are showed in table 1. During the simulations, nodes move following two mobility models: Random Way Point (RWP) and Community-based (CMM) [14]. The former is a classic model where each node chooses a random target location, moves to the destination with a random speed, then waits for a random period of time before repeating this process. The latter is a recently proposed model that exploits social networks to generate more realistic mobility traces. The experiment selects randomly a node as a start node of query. Each experiment consists of at least 10000 trials and the average values are calculated as experiment results subsequently.

This paper implements two kinds of experiments. Experiment 1 calculates our algorithm’s search efficiency and message traffic under the variety number of ants. Search efficiency is the ratio of the number of success search to the total. Message traffic is defined as the total number of query messages generated during the algorithm runs. The number of messages could impact network load and computing resource in the node.

Fig.1 shows that with the number of ants increasing, search efficiency will properly increase, but message traffic will increase simultaneously. This is because the increasing number of ants expands the width of lookup. In order to ensure search efficiency, increasing the number of ants is an effective method. We also could reach a tradeoff between search efficiency and message traffic. For example, under the scenario of experiment 1, when the number of ants is set to 25, a better search efficiency and acceptable message traffic can be gotten.

<table>
<thead>
<tr>
<th>TABLE I.</th>
<th>PARAMETERS IN SIMULATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Values</td>
</tr>
<tr>
<td>Simulation time</td>
<td>6000s</td>
</tr>
<tr>
<td>Network area</td>
<td>(1000 \times 1000)</td>
</tr>
<tr>
<td>Transmission range</td>
<td>50m</td>
</tr>
<tr>
<td>Transmission throughput</td>
<td>30kbps</td>
</tr>
<tr>
<td>Transmission delay</td>
<td>Uniform in ([0.1,0.5]) s</td>
</tr>
<tr>
<td>Mobility</td>
<td>Uniform in ([1,5]) m/s</td>
</tr>
<tr>
<td>Request interval</td>
<td>10s</td>
</tr>
<tr>
<td>No. of nodes</td>
<td>100</td>
</tr>
<tr>
<td>No. of services at each node</td>
<td>Uniform in ([5,10])</td>
</tr>
<tr>
<td>Initial TTL</td>
<td>7</td>
</tr>
</tbody>
</table>

Experiment 2 compares our algorithm with other two classical algorithms, i.e. broadcast-based and Random Walk-based. Three metrics are calculated that are average path length, search efficiency and message traffic. The simulations run with RWP model and CMM model respectively.

The simulation results in Table 2 and Table 3 show that our algorithm works considerably better than other algorithms with the RWP and CMM model. Our algorithm has shorter average path length and higher search efficiency, and considerably decreases message traffic. With the RWP model where a node’s movement is completely random, the performance of our algorithm is not as good as with the CMM model but its performance is not worse than the other algorithms.

The simulation results clearly show that our proposed algorithm is really appropriate for pervasive and mobile networks.
message as an artificial ant, and target service as the food which ants wanting to search. The algorithm utilizes pheromone and social utility as heuristic information that directs an ant to select next hop. By adjusting the number of ants, a better tradeoff between search efficiency and cost could be achieved. Moreover, the maintenance cost of a pheromone table is cheap, which hardly increases the burden of the resident node.

ACKNOWLEDGMENT

This work is supported by Anhui Provincial Higher Education Excellent Young Talents Foundation of China under Grants No. 2010SQRL040.

REFERENCES