CSDST- A Cluster Based Service Discovery Approach with Service Type in Pervasive Computing

Ali Golzadeh

Master of Computer Engineering, Iran

Summary
the goal of pervasive computing is to provide computing and communication services, particularly for mobiles that interact through ad hoc connections, anytime and everywhere. This paper presents a distributed approach for service discovery in MANET, one of the most important networks in pervasive computing, and combines both centralized and decentralized approaches in order to take the best advantages of them. The proposed approach is based on nodes clustering in which every cluster has a head node acting as a centralized directory as well as replies service requests. In our combinatory approach service types are used to enhance service discovery process. Compared to the existing approaches, the proposed approach has the advantage of decreasing the required time for service discovery, and applying less traffic in the network.

Keywords
Pervasive computing, Service discovery, MANET, Clustering

1. Introduction

Pervasive computing makes the actual computing and communication essentially transparent to the users [1]. Recently, much attentions has been directed to one of the networks in pervasive computing called MANET; Mobile Ad-hoc Network. MANET is a self-configuring network that is formed and deformed by a collection of mobile nodes without the help of any prior infra-structure or centralized management [2]. One of the most critical issues in MANET, as stated by [3], is Service Discovery, which is defined as the process of mapping a service description to a so called service location. It should be noted that service in a network is any software or hardware entity a user may wish to use.

Several approaches have been presented for service discovery; some of them tried to propose for internet applications such as SLP [4], SSDP [5], and DNS-SD [6] while the others tied to a high-level application technology, like Jini [7] and Salutation [8]. These approaches do not use any performance metrics and do not directly dwell on MANET environments where self-configurability is the key and proper for wired network [14].

More recently, other service discovery approaches have been proposed which are specifically designed for ad hoc networks. Some of these are tied to a particular sub network technology (e.g. Bluetooth's SDP [9]), while others broach service discovery and routing together (e.g. GSD [10], and HSID [11]) or at an application level (e.g. DEAPspace [12], and Konark [13]). These approaches broadcast a message to discover and advertise services. Due to their heavy consumption of bandwidth and energy, broadcasting mechanisms are not suited to MANETs. Therefore, the network size it supports is limited. Some of these approaches will be discussed later.

Due to the dynamic nature of MANET, there are always spatial and temporal variations in the availability of services [16]. Hence, a service discovery strategy should be highly robust, efficient and dynamic in nature. In this paper, we present a distributed cluster based service discovery that divides the nodes of the network into clusters. Clustering uses the resources in MANET effectively [15]. For each cluster, there exists a head that services register themselves in it. When in a service is needed, the user first peruse its memory; if the asked service couldn't be found, a request would be sent to the head. If the head could find the service, it would reply it; otherwise, it sends the request to the nearest cluster where the demanded service type exists in.

The main idea of this paper is using service type in service discovery. Each head keeps all the service types included in its cluster and the requests are sent to a cluster where the type of demanded service exists in. In each cluster, information about the other clusters is saved with their time distances to it. This information makes an algorithm which would be discussed later.

2. Related Work

Service discovery addresses the general problem in which some elements in a network, the clients, need to know the services offered by other elements in the network, the servers. We can classify Service discovery in three parts: Centralized directory, Directory-less (push and pull mode), and Distributed directory.

A. Centralized directory

In the centralized approach, a directory is a single physical host or a hierarchy of hosts which contains

Manuscript received June 5, 2026
Manuscript revised June 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.6.6

descriptions of all services available in the network. Providers register their services in the directory and clients send their requests to it. For instance, Jini and SLP are two approaches which are developed based on centralized directory.

- Jini approach: it supports discovery of Java based services. A client must first find a nearby lookup service using the Jini discovery and join protocols. The client instantiates a 'multicast response' service, and then issues a search for available lookup services using the multicast request protocol. The Available lookup servers will respond to the client's multicast using the unicast discovery protocol.
- SLP approach: The Service Location Protocol (SLP) is a protocol for automatic resource discovery in IP based networks. The discovery mechanism of it is based on service attributes. SLP service agents are responsible for advertising service to the directory agents, which make services available to the user agents. This protocol also supports a simple service registration mechanism.
-

B. Directory-less (push and pull mode)

In this state there is no directory and the discovery of services can be done in two ways: (i) push mode, in which servers send unsolicited advertisements, and clients listen to these advertisements and select the services they are interested in; (ii) pull mode, in which clients request a service when they need it, and servers that offer the service to answer the request. DEAPspace and Konark are two examples of this kind of approach.

- DEAPspace approach: DEAPspace Algorithm proposes a push solution, in which all the devices hold a list of all known services, called "World View". Each device periodically broadcasts its "world view" to its neighbors, which update their "world view" accordingly. DEAPspace confines itself to a small network by assuming a single hop ad hoc network and uses broadcasts to send the messages.
- Konark approach: Konark is a middleware designed specifically for the discovery and delivery of services in multi-hop ad hoc networks. Konark supports both push and pull modes, with a cache in all devices. Konark has defined a new discovery mechanism, called Konark service gossip protocol. This is based on a message exchange round, which is triggered by a service request message or by a service announcement message.
-

C. Distributed directory

In Distributed directory, cluster headers take a role as central lookup servers in their clusters. So, centralized service discovery is locally performed in each cluster at first. And, in order to achieve global discovery over all clusters, cluster headers summarize their cached service descriptions and then deliver the summarized information to all other cluster headers. Compared to other cases, fewer approaches have been presented for this directory. Sandman and CSDM are two examples of this approach.

- SANDMAN approach [16]: SANDMAN uses a node clustering approach, in which cluster head always stays active and answers discovery requests on behalf of the nodes in its cluster. This allows the nodes in the cluster to maximize their sleep times. This approach is shown to be energy efficient for scenarios with large idle times. This approach only specifies service discovery in cluster.
- CSDM [19] Approach: we call it as CSDM. It is a Cluster based Service Discovery Model. In this approach, when a service provider does not exist in a cluster, the head sends a request to the nearest cluster that saved its information before.

Directory-less architectures usually broadcast a message to discover and advertise services. Broadcasting mechanisms are not suited to MANETs due to their heavy consumption of bandwidth and energy; both have limited usage in mobile devices. In Centralized discovery, the central server constitutes a bottleneck and as a result of exiting mobile nodes, setting a directory is almost impossible for them.

Distributed directory architectures are well suited to the MANET scenario, as distributed directory has directory state and exchanges less messages in the network. This approach operates based on distributed directory, assumes cluster of nodes and a head for each cluster and proposes an algorithm for service discovery among clusters. Table 1 shows some approaches and compares them based on clustering and connection between nodes.

3. The Proposed Approach CSDST

In recent years, many clustering algorithms have been proposed to determine clusters of MANET in order to improve the efficiency of routing protocols and save energy, to implement efficient flooding and broadcasting mechanisms. For example, Kettaf et al. in [17] presented a clustering approach for leader election in MANET based on QoS parameters such as the bandwidth, the mobility of nodes, energy and interference. Chatterjee et al. in [18] presented a clustering approach that uses a proactive routing protocol and information from neighboring nodes for the cluster formation and maintenance processes.

The presented approach in this paper is called CSDST (Cluster Based Service Discovery with Service Type) which is independent from clustering algorithm. In this clustering, each node is only member of one cluster and each cluster only has one head. The head is elected

based on some parameters like battery longevity, mobility, and presentence time of nodes among nodes in the cluster. For observing equanimity as well as preventing the use of only one node's resource for head, head selection algorithm is repeated in cluster and the head is changed. Each head has two tables. The first table is a service provider's directory or SPsL (Service Provider's List) that keeps all the information about service providers existing in the cluster. The information includes service provider's name, its address and time availability, as well as service name and service type.

Table 1 . comparing of some approaches in clustering

approach	Type of service discovery	Head	Service discovery among clusters
Jini	Centralized directory	×	×
SLP	Centralized directory	×	×
DEAPspace	push and pull	×	×
Konark	push and pul	×	×
SANDMAN	Distributed directory	✓	×
CSDM	Distributed directory	✓	✓
Our approach (CSDST)	Distributed directory	✓	✓

Second table is heads directory or Hsl (Head's List) that keeps information about other clusters. The information includes heads name and address, and presented service types in these clusters. In order to communicate with other clusters, each head broadcasts Request-HsL in the network after election of it in cluster. The Request-Hsl includes name and address of head and service types that exists in cluster. After receiving this request, other heads first store requester head's information in the HsL table, then they send a unicast reply by the name of Reply-HsL, which includes its own head name and address and cluster service types to the requester. The first head sets all the replies in the HsL table in order. Sequence of setting replies in the table is based on RTT (Round Trip Time) or go and back time of request. This means if Reply-HsL of a head is received sooner, it is set upper in the table than other replies received after it, which also shows that this cluster is closer to the head. In this manner, all of the Reply-HsLs are set downward in HsL table based on their RTT. After receiving a Reply-HsL from another head, the first head sends an Ack-HsL to it in reply that contains RTT; thus, the second head has now its distance time to the first head and can set dependent record of it in HsL table in proper place based on its RTT. When a head is made invalid in cluster for any reason, like being moved to another cluster, turned off or regard to resources, a new head is first elected, then previous head sends a message to other heads that are in HsL table for replacing the new head's information. At the end, the information of HsL and SPsL tables from the old head are moved to the new one.

Service type is an important field that shows the service belongs to which service provider groups. For example, if a provider is used for printing service, its service type would be considered as printer group type, while its service name field would show if it is black and white, or colored. It should be noted that Service type field has an important role in decreasing request numbers between clusters in the network because a request does not send to a cluster when service type does not exist in it.

A. Service Registration

When a head is elected in a cluster, service providers have to register their services in the cluster. Registration information consists of the name and type of the service, the name and address of the service provider, and the more information existed in description file in service provider. If a new service provider enters the network, it uses the cluster algorithm to detect its cluster. Then, it sends a request to its neighbors in cluster to find the cluster head and register its services in the head. Moreover, each service provider registers the availability time of the service, which means the time that the service is available in the cluster. If, for any reason, service provider wants to leave the cluster before this time, for example to move to another cluster or turn off, it must send Deregister request to the head to delete its record. The head considers type of services when information about a new service provider sets in SPsL table.

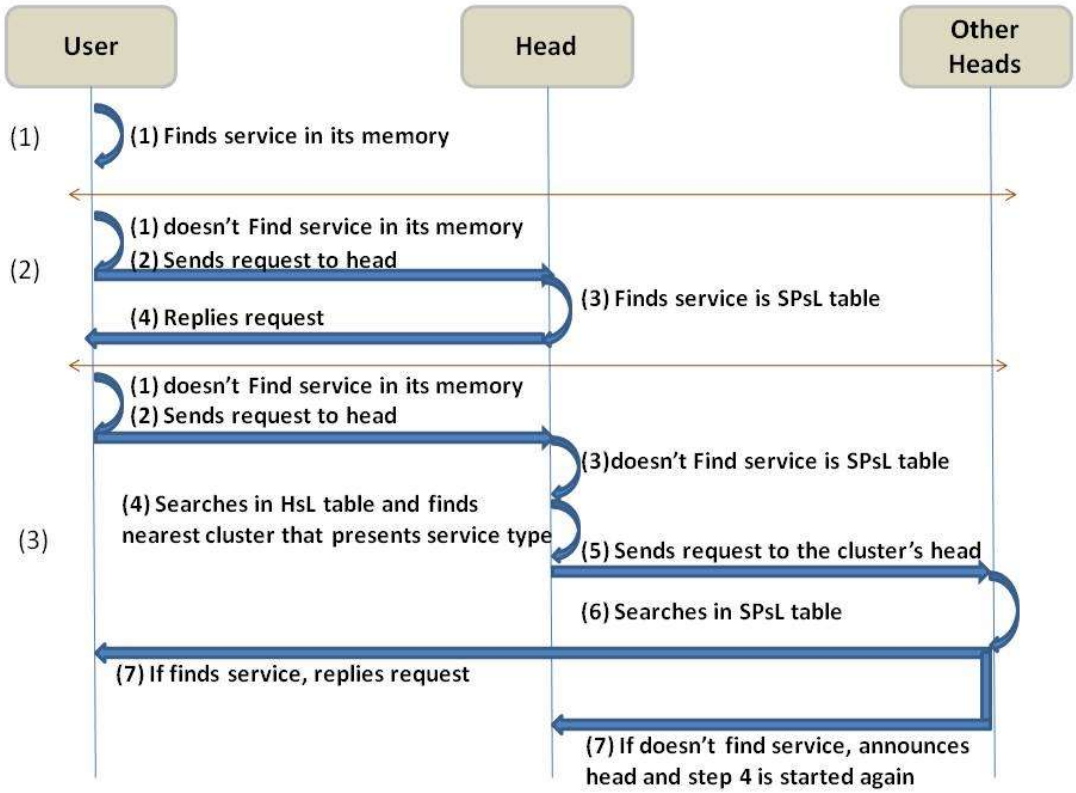

Figure 1. sequence diagram of service discovery- (1) in a node, (2) in a cluster, (3) among clusters

If its services are new and haven't been existed before in SPsL, the head takes out addresses of other heads from the HsL table and announces them the new service or services which they could register to their HsL table. If some service providers leave a cluster and a service type does not exist in the cluster anymore, the head has a task to announce to all of heads that are in its HsL table to delete the service type from record of this head in their HsL tables.

B. Service discovery

Service requester first considers its directory (memory) to find the information about the service provider. If the information isn't found, Service requester would send a Lookup-service request to its own head. Head considers its SPsL table if it finds any Provider, and replies to the requester; otherwise, it sends a request to the first record of its HsL table that has the same type of request service. So it saves time and cost by not sending it to far clusters. Number of service discovery requests among clusters has been reduced by the service type field. When another head receives Lookup-Service request, it searches in its SPsL table and sends the address of service provider and primary information to the requester. It is possible that the service request type and presented service in cluster seem similar at this state but they don't present the exactly the same service, which means the service names are different for them for example, the type of service can be print in the request and the name of service can be black and white, but this cluster can present a color printer. In such cases, the cluster head signals to the first head to select the next record in its own HsL table with the same type of service.

Figure 1 shows a sequence diagram presenting phases of service discovery in a node, a cluster and among clusters. Figure 2 illustrates phases of service discovery clearly. When the User 3 in cluster 1 asks for a service, it sends a Lookup-service request to its head after considering its memory and not finding any information (1). The head first searches in its SPsL table and when it can't find any

answers (can't find a service provider in its cluster), then it searches in its HsL table and filters it based on service type that user's need. Suppose cluster 2 is the nearest cluster to the cluster 1, but the requested service type does not exist in it. If the next nearest cluster is the cluster 3 in which the requested service type exits, the head sends a Lookup-service to its head (2). Based on its SPsL table, Cluster 3's head knows that although it presents the service type, it does not present the exact service (i.e. service name does not match), so it denies the request of cluster 1 with an acknowledgement (3). The head of Cluster 1 transfers the request to the cluster 4's head that presents the same service type (4). In this cluster the name of service is the same of request service name and its head considers the SPsL and sends the primary information about the service provider that presents the exact service to the requester (5). Requester sends an invocation request to the service provider and registers its information in its memory for future requests (6). As it shows in figure 2, message 3 is colored lightly since it is only an acknowledgment and it is disregarded.

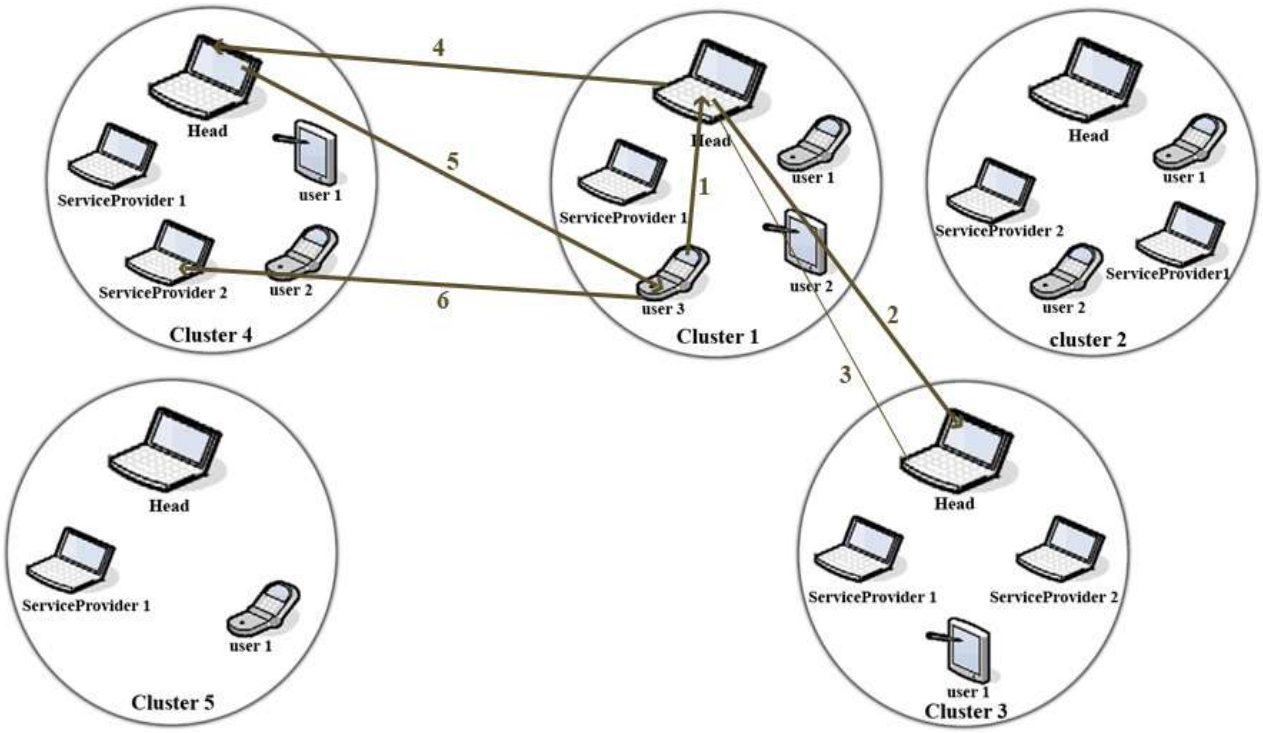

Figure 2. An Example of service discovery scenario

4. Evaluation

As mentioned in part 2.C, CSDM and SANDMAN approaches, like CSDST, are based on distributed directory. SANDMAN approach doesn't mention about connection state among clusters when there does not exist any service provider in the cluster (does not refer to service discovery among clusters). In CSDM approach service discovery is done based on sending request to a cluster in neighborhood. However, CSDST exploits the idea of using service types for faster service discovery among clusters. CSDM approach transmits request hierarchy among clusters until it finds a service provider whereas in CSDST approach, having service type in its HsL table and the service request, CSDST sends a request to the nearest cluster that presents the type of service request. CSDST makes faster discovery of services and applies less traffic in the network.

To prove the less service discovery time, we assume that each cluster has a distance of t time with its nearest cluster, $t+\alpha$ time ($\alpha<t$) for next nearest cluster, $t+2\alpha$ time for the third nearest cluster and $t+(n-1)\alpha$ time for the distance of cluster n. In fact, time distance may not be limited and specific but for simplifying evaluation, the time distance is assumed t between a cluster and the nearest cluster and t plus a multiple of α between other clusters to

it. α has smaller value than t, and is disregarded in computation; disregarding it means the distance between clusters is almost the same. For the discussed approach in this paper (CSDST) if we assume that the cluster n is the first cluster that has the requested service type, the time of service discovery in best case equals:

$$T=t+(n-1)\alpha \quad (1)$$

for CSDM approach, if the closest group, e.g, group two to the head, has a service request with t time distance, the cluster near to the second cluster has t+α time distance. When these steps are continued, following sequence is obtained:

$$t, t+\alpha, t, t+\alpha,\ldots,t \text{ or } t+\alpha$$

Therefore, such services should pass medial clusters (n-1 cluster) in order to reach cluster n, the whole time equals:

$$T=nt+(n/2)\alpha \quad (2)$$

Figure 3 compares formulas 1 and 2 without considering α (we assume that t equals 0.1 in formulas). As it is clear in this figure, at the best case, service discovery time in CSDST equals the time distance of the nearest cluster.

For the medium case, it is assumed that the first cluster, to which the request has been sent, doesn't reply service exactly. This means some other clusters were investigated before finding a service provider in cluster n. For this state, we assume n/2 clusters have been investigated before cluster n (possibility of occurrence of this state with service type is small). When n is an odd number the sequence is:

$$t, t+2\alpha, t+4\alpha, \ldots, t+(n-1)\alpha$$

Total time equals:

$$T=((n+1)/2)t+(n\hat{}2)\alpha \quad (3)$$

And when n is even:

$$t+\alpha, t+3\alpha, t+5\alpha, \ldots,t+(n-1)\alpha$$

Total time equals:

$$T=(n/2)t+n(n-1)\alpha \quad (4)$$

Formula 2 in each case is common for CSDM approach, as some clusters should always have been considered for service discovery. Figure 4 compares formula 3 (which is greater than Formula 4) to formula 2 with disregarding α and figure 5 compares them with regarding α. These figures show that CSDST approach gives better result when α is smaller, which is true in cases that clusters are near to each other but as shows figure 5 the CSDM approach is better when clusters are far from each others with many nodes in the network. In the worst case, when all clusters before the cluster n were considered, the total time is calculated as following:

$$T=nt+(n(n-1)/2)\alpha \quad (5)$$

Probability of this case is almost zero, because this state happens when service type does not exist in HsL table and the head sends requests to all the clusters in the network.

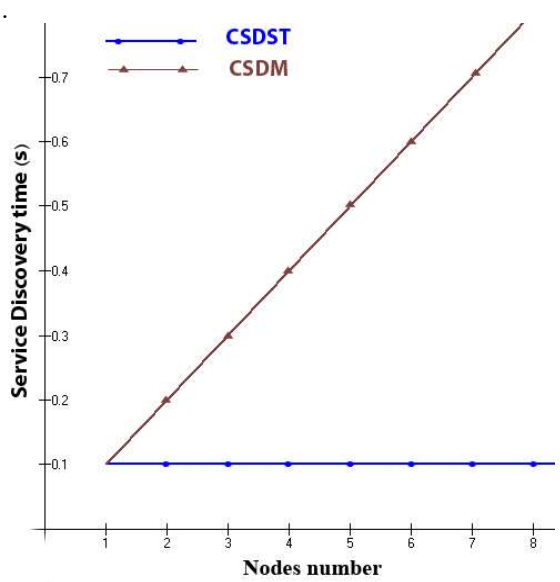

Figure 3. the best case

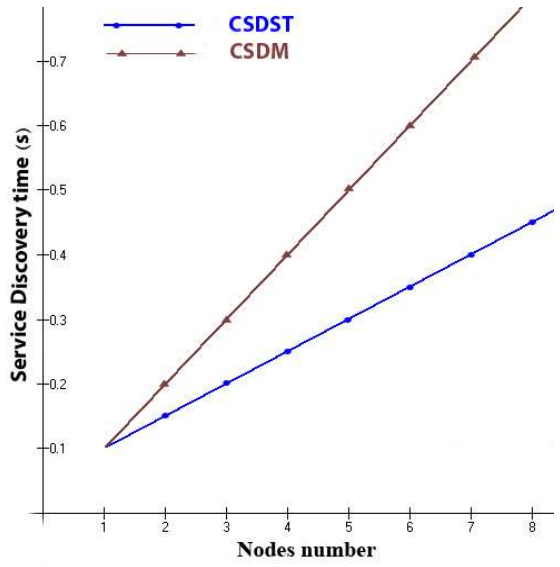

Figure 4. The medium case without α

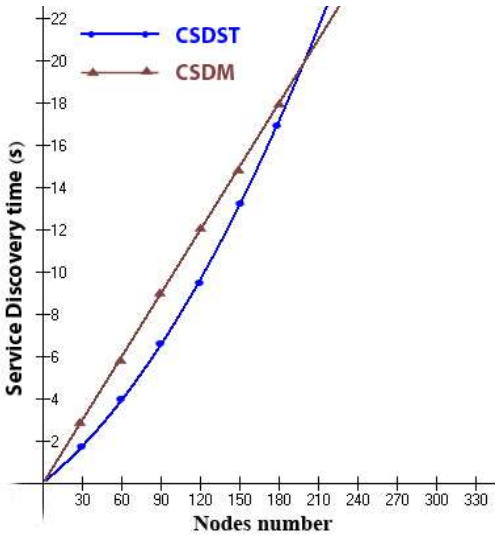

Figure 5. The medium case with α

To prove that less traffic is applied in the network, it should be considered that medial clusters do not exist in CSDST to transmit the requests and the requests are sent directly to the head presenting the service type. Therefore, consuming of nodes battery is decreased and less traffic is applied.

5. Conclusion and Future Work

In this paper, a service discovery approach based on nodes clustering is presented. Each cluster has a head that plays the role of a directory in that cluster. Unlike centralized directory approach, there is not a central directory that brings bottleneck. Similarly, unlike push and pull service approach, nodes resources aren't wasted with announce and request service. In this approach each service provider should register its services and type of services which presents to one of the heads. When its memory consideration for service directory was unsuccessful, Service requester sends a request to its head.

The head first searches its SPsL table; if it can't find the service provider in this table, it then considers its HsL table. In this table, in addition to information about other clusters, there exists a service type field to show what types of services are in each cluster and decreases request numbers among clusters. The head sends a request to the nearest cluster that has the service type. If the nearest cluster presents the exact service, it replies the request; otherwise, the head selects the next pursuant head from HsL table. This process would continue until a service provider is found. Since the request isn't sent to clusters that don't have the service type and far clusters from the requester, service discovery time decreases, which economizes in using nodes resources. As we show in evaluation section, service discovery time is almost fixed in best case when clusters have same distance time. Future effort will work on how to specify service types in clusters and will present a candid algorithm to detect the head in a cluster.

References

[1] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, August 2001, pp. 10-17.

[2] Y. Xiao, X. Shen, and D.-Z. Du, "A Survey on Intrusion Detection in Mobile AdHoc Networks",Chapter 7, Wireless/Mobile Network Security, pp. 170 - 196, 2006

[3] C. I. Katsigniannis et. al. "Architecture for Reliable Service Discovery and Delivery in MANETs Based on Power Management Employing SLP Extensions", IEEE Wireless Communications, October 2000

[4] E. Guttman, C. Perkins, J. Veizades, M. Day. RFC 2608: Service Location Protocol, Version 2, June 1999.

[5] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu, Simple service discovery protocol/1.0. Internet draft (work in progress), April 1999. draft-cai-ssdp-v1-03.txt.

[6] Stuart Cheshire, DNS-Based Service Discovery. Internetdraft (work in progress), February 2004.

[7] Jini Architectural Overview, White Paper, 1999.

[8] Salutation Consortium, 1998. Available from: <http://www.salutation.org>.

[9] Bluetooth Specification v1.1, Part E: Service Discovery Protocol (SDP).

[10] Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, Tim Fini, GSD: A novel group-based service discovery protocol for MANETS, in: 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm. Sweden, September 2002.

[11] Chang-Seok Oh, Young-Bae Ko, Jai-Hoon Kim, A hybrid service discovery for improving robustness in mobile ad hoc networks, in: The International Conference on Dependable Systems and Networks, DSN-2004, Florence, Italy, July2004.

[12] Michael Nidd, Service discovery in DEAPspace, IEEE Personal Communications, August 2001.

[13] Sumi Helal, Nitin Desai, Varum Verma, Konark—A service discovery and delivery protocol for adhoc networks, in:Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.

[14] C. Dabrowski and K. Mills, “Analyzing Properties and Behavior of Service Discovery Protocols using an Architecture-based Approach”, Proc. Working Conf. on Complex and Dynamic Systems Architecture, 2001.

[15] C. Perkins “Ad Hoc Networking” Addison-Wesley, Second Edition, 2004.

[16] Filip Perich, Anupam Joshi, Rada Chirkova, "Data Management for Mobile Ad-Hoc Networks"

[17] N. Kettaf, H. Abouaissa and P. Lorenz, A Self Organizing Algorithm for Ad hoc Networks, Personal Wireless Communications PWC’05, Proceedings of the 10th IEEE/IFIP Conference. Colmar, August 2005.

[18] Nevadita Chatterjee, Anupama Potluri, Atul Negi, A Self-Organizing Approach to MANET Clustering, HiPC06, Goa, India, 2006.

[19] Hassan Artail, Haidar Safa, Hicham Hamze, Khaleel Mershad, “A Cluster Based Service Discovery Model for Mobile Ad hoc Networks,” Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007).