Towards Scalable and Robust Service Discovery in Ubiquitous Computing Environments via Multi-hop Clustering

Wei Gao
Department of Computer Science and Engineering
Arizona State University, Tempe, AZ 85287-8809, USA
w.gao@asu.edu

***Abstract*— Large-scale ubiquitous computing environments require scalable and robust service discovery to enable "anytime, anywhere" computing, which is hard to be satisfied in flat network architecture. In this paper, a multi-hop Cluster-based Architecture for Service Discovery (CASD) in ubiquitous computing environments is presented. Based on the Neighborhood Benchmark (NB) which quantifies the connectivity and link stability of mobile nodes, CASD organizes the network consisting of heterogeneous mobile computing devices to be multi-hop clusters, and constructs each cluster to be a local DHT-based p2p network for distributed storage of service indices. The clusterheads are connected together to form a virtual backbone, based on which service discovery messages are disseminated among clusters. It is shown that CASD can control the communication overhead of service discovery when the network scale increases, and that CASD can achieve robust service discovery by maintaining controllable redundancy of service indices in each cluster. The scalability and robustness of our approach in various types of network settings are shown by intensive simulations.**

I. Introduction

A *service* is defined as a self-contained software entity with a discoverable and invocable interface to provide certain capability. Services developed and provided on different platforms may have different interface descriptions. Ubiquitous computing environments enable mobile users from heterogeneous network domains to discover services provided by others and to collaborate with each other, and thus enable "anytime, anywhere" computing. Due to the highly dynamic nature of such environments, service discovery in such environments needs not only to be *scalable* to discover the desired services quickly, accurately, and efficiently, but also to be *robust* against unpredictable network topology changes.

A service discovery system consists of three components: service repositories, service providers, and service requesters. Service repositories store announced service descriptions as indices, and reply to service requesters with matching service indices. A service discovery process therefore consists of the dissemination of service discovery messages [1] and the matchmaking between the services requested and provided [2].

Due to the service heterogeneity in ubiquitous computing environments, the services provided will rarely match the user's requests perfectly. Instead, an acceptable deviation between the services provided and requested should be used in service matchmaking, and there may be multiple matching services for one request. Since the accurate descriptions of the matching services cannot be decided in advance, a service discovery process has to search the entire network for all the matching services, and the scope of service discovery is fixed to be "global". Therefore, to achieve scalable and robust service discovery, we need a network architecture to select service repositories appropriately, and to disseminate every service request efficiently to all the possible repositories containing matching service indices.

In this paper, we present a multi-hop Cluster-based Architecture for Service Discovery (CASD) in ubiquitous computing environments, and we assume that a usable service matchmaker is available. CASD organizes the network to be multi-hop clusters with better stability and flexibility comparing to simple one-hop clusters, by exploiting autonomous clusterhead selection based on the nodes' Neighborhood Benchmark (NB) scores. Since the NB quantifies the connectivity and link stability of mobile nodes, the clusterheads selected are efficient to be the aggregation points of data flows, and are stable to avoid frequent clusterhead changes. CASD also ensures that all the constructed clusters are connected ones by exploiting a handshake process between the cluster members and clusterheads. Clusterheads are connected together to form a virtual backbone, through which service discovery messages are disseminated to clusters.

Each cluster in CASD is constructed to be a local DHT-based p2p network extending the Content-Addressable Network (CAN) [3], for distributed storage of service indices with controllable redundancy. We forward every message to multiple service repositories in each cluster to achieve such controllable redundancy, and to ensure that all the matching services are found.

Our approach has the following distinguished features:

- It is able to reduce communication overhead of service discovery via multi-hop clustering, and thus to be scalable in large-scale ubiquitous computing environments.
- It maintains controllable redundancy for storing service indices. Hence, it achieves robust service discovery such that, in cases of bounded numbers of node failures, the desired services can still be found successfully.
- It guarantees that a service request can find all the matching services in the network.

II. Related Work

Service discovery has been studied assuming the availability of fixed network infrastructure, static network conditions, and centralized servers, such as Jini [4] and UPnP [5], which are not suitable for ubiquitous computing environments with highly dynamic and heterogeneous nature. P2p networks based on Distributed Hash Tables (DHT) have been well researched [3], [8], and are also used for service discovery [9]. In CAN [3], the network architecture is designed to be a virtual d-dimensional Cartesian coordinate space on a d-torus. The virtual space is dynamically partitioned among all the nodes such that every node possesses its individual zone. The virtual space is used to store $\{$*key, value*$\}$ pairs such that the unique key of a data object is deterministically mapped onto a point, named P, in the virtual space using a uniform hashing function. The corresponding data object is stored at the node that owns the zone within which the point P lies.

However, a global p2p network cannot achieve scalable and robust service discovery in large-scale ubiquitous computing environments. The overlay p2p topology is expensive and vulnerable to node mobility because it is not related to the physical network topology, and to deploy a global p2p network is in low efficiency due to frequent node updates and re-hashing. In addition, a service request may not be forwarded to the nodes containing matching service indices because of the deviation between the services requested and provided and the rigid content lookup mechanism in regular p2p networks.

Instead, the network needs to be organized hierarchically. CARD [7] proposes a hybrid service discovery scheme. However, the nodes in CARD need to maintain too much information for service discovery, including their vicinity information and the random-selected contact lists. Cluster-based routing approaches in MANETs have been well researched [10], and are also exploited for service discovery. Kozat et al. [6] suggested a cluster-based network layer support for service discovery, in which the selected clusterheads form a backbone to disseminate service discovery messages. This approach achieves better efficiency and scalability. However, this work constructs one-hop clusters based on the concept of connected dominating set (CDS) [11], and select clusterheads in a random manner. The stability and flexibility of the clustered network architecture is hence seriously limited. In contrast, D. Kim et al. [12] first defined a k-hop cluster. Some extensions [13], [14] have been made to ensure the connectivity among clusters, but none of them take the quality of the clusterhead selection into consideration. Some other works [15], [16] group nodes according to the node mobility pattern, transmission power and so on, but none of them guarantee to construct connected clusters, which is a common problem when constructing multi-hop clusters.

III. Network Model

Ubiquitous computing environments incorporate heterogeneous mobile computing devices from multiple network domains together. We achieve scalable and robust service discovery in such environments by choosing appropriate service repositories, and forwarding service announcements and requests to the repositories efficiently and reliably. In the extreme case, a ubiquitous computing environment globally is a Mobile Ad-hoc Network (MANET), in which all the mobile computing devices are moving in the application area, and communicate with each other to construct self-organizing network topology. Hence, without loss of generality, in the rest of this paper, we will consider this extreme case for service discovery, and assume that all the nodes are distributed uniformly in the network. Practically, the computing devices may be accumulated in several network domains, in each of which they are relatively close to each other. However, each of those domains can be treated as a sub-network separately.

Without loss of generality, we assume that the network is connected. Since CASD works at the application layer to directly provide service discovery functionality to users, we assume that the network is equipped with appropriate routing and transport layer protocols for packet delivery.

IV. Construction of Multi-hop Clusters

In CASD, the NB score of a node N_i used to indicate the qualification of this node to be a clusterhead, is defined as:

$$NBS_i = d_i / LF_i \tag{1}$$

where d_i is the neighbor[1] degree of N_i indicating the connectivity of N_i's neighborhood, and LF_i is the number of link failures encountered by N_i in unit time indicating the link stability of N_i's neighborhood.

A. Network Initialization

A network initialization phase is needed after the network starts, to initialize the NB scores of mobile nodes. The initialization is done via hello beaconing. The interval of such hello beaconing is $T_H = r_c/v_c$, where r_c is the average transmission range of the nodes, and v_c is the average moving speed of mobile nodes in the network. Such interval is hence positively proportional to the global network connectivity, and negatively proportional to the network mobility level.

The hello beacons sent by node N_i are encapsulated as $\{$*IP*$_i$, *NBS*$_i$, *Head_IP*$_i$, *HEAD_NBS*$_i$, *hop*$_i$, *size*$_i\}$, where *IP*$_i$ and *NBS*$_i$ are for N_i, and *Head_IP*$_i$ and *HEAD_NBS*$_i$ are for the current clusterhead of N_i. *hop*$_i$ indicates the hop count from N_i to its current clusterhead, with the range $[0, R]$. *size*$_i$ is the size of the cluster to which N_i belongs. Specifically, if N_i is in the process of clusterhead selection, *Head_IP*$_i$ and *HEAD_NBS*$_i$ are for the current temporary clusterhead of N_i.

Every node maintains a neighborhood information table (NIT). If a neighbor record in the NIT has not been updated longer than $2T_H$, the corresponding neighbor is considered unreachable, and one link failure is counted. Nodes count their numbers of neighbors and link failures before they send out the hello beacons, and calculate their NB scores in (1). Specifically, LF_i is calculated iteratively in every hello beaconing round as (2):

$$LF_i^{(k)} = (LF_i^{(k-1)} \cdot (k-1) \cdot T_H + LF_{new})/(k \cdot T_H) \tag{2}$$

[1]If not specified, the term neighbor in this paper is equivalent to 1-hop neighbor.

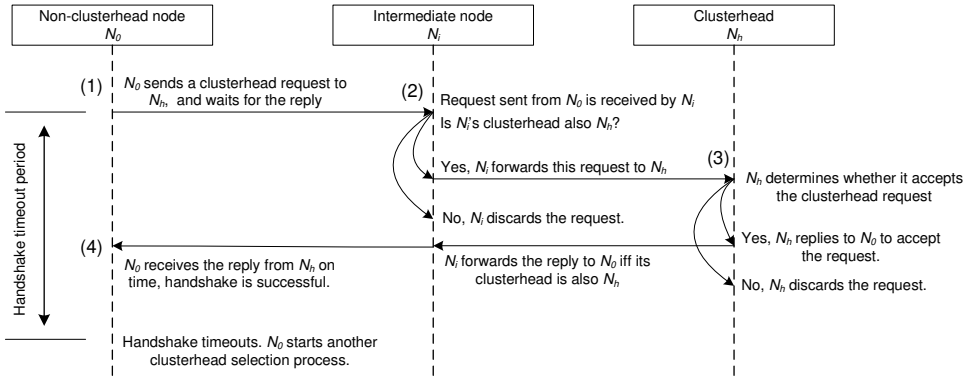

Fig. 1. Handshake with clusterheads

where k is the current hello beaconing round, and LF_{new} indicates the number of link failures detected in this round. After the network initialization, hello beaconing will be conducted continuously in the lifetime of the network to keep updating the NB scores of mobile nodes.

B. Autonomous clusterhead selection

Every node starts to select its clusterhead simultaneously and autonomously after the NB scores of all the mobile nodes have been initialized. The clusterhead selection consists of R consecutive rounds, where R is the cluster radius, and each selection round is corresponding to a hello beaconing round. A selection round will not start at a node until it receives all the corresponding hello beacons from its neighbors.

In a clusterhead selection round, a node N_i puts all the clusterheads of its 1-hop neighbors (obtained from its NIT), and its own temporary clusterhead, into a selection pool. For a node not having clusterhead yet, the node itself is used as its temporary clusterhead. Then, N_i selects the node with the highest NB score in the selection pool to be its temporary clusterhead. If there are nodes with equal NB scores in the selection pool, one of them will be selected at random. The correctness of the clusterhead selection process is proved by the following theorem and corollary.

Theorem 1: The kth round of clusterhead selection on a node N_i will select the node with the highest NB score within the k-hop neighborhood of N_i.

Proof: We prove this theorem by induction. In the first round, the size of N_i's 1-hop neighborhood is equal to the size of its NIT, hence the maximum selection pool includes N_i and all of N_i's 1-hop neighbors. Assume the theorem holds after mth round, then in the $(m+1)$th round, all the possible new clusterhead candidates are selected from N_i's $(m+1)$-hop neighbors, because they are selected from the m-hop neighbors of N_i's 1-hop neighbors. ■

Based on Theorem 1, we can easily prove the following Corollary 1:

Corollary 1: Given a cluster C=$\{N_i|$ *i=0, 1, …..n*$\}$, if a node $N_k \in C$ is the clusterhead of C, then for $\forall i \in [0, n]$, $i \neq k$, $NBS_k \geq NBS_i$.

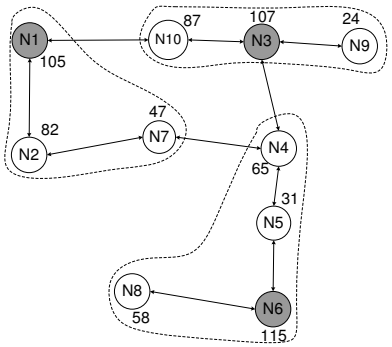

Fig. 2. An example of 2-hop clusters

TABLE I
AN EXAMPLE OF CLUSTERHEAD SELECTION PROCESS

	N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**	**N9**	**N10**
Round1	N1	N1	N3	N3	N6	N6	N2	N6	N3	N3
Round2	N3	N1	N3	N6	N6	N6	N3	N6	N3	N3

C. Handshake with clusterheads

After the autonomous clusterhead selection, the multi-hop clusters are constructed by letting all the nodes handshake with their selected clusterheads. Such handshake process is described in Fig. 1 by using a case including a node N_0, its selected clusterhead N_h, and an intermediate node N_i. In this process, it is possible that N_h is also requesting for another node to its clusterhead, or there are some other nodes which are requesting N_0 to be their clusterheads. In such cases, if a node in its handshake process is requested to be clusterhead by some other nodes, and the request is accepted, it should send another message to its selected clusterhead to cancel the ongoing handshake process, and start to handshake with the requesters as a new clusterhead.

An example of 2-hop clusters is illustrated in Fig. 2, with the NB scores shown aside of the nodes. The clusterhead selection process of this example is shown in Table. I. In this example, although N1 selects N3 to be its clusterhead, N1 cannot join the cluster of N3 because it is also requested to the clusterhead of N2. Besides, because every mobile node only has the knowledge of its 1-hop neighborhood, there will be possible inconsistency during the clusterhead selection and

handshake process. In the example above, in the first selection round, N4 selects N3 as its clusterhead, and notify this to N7 via hello beaconing. In the second round, because the NB score of N3 is higher than that of N2, N7 also selects N3. At the same time, N4 hears from N5 about N6 with a higher NB score, and N4 will change its clusterhead to N6. However, this change will break the link from N7 to N3, and thus produces a disconnected cluster.

This problem is solved in the handshake process such that, a node will only forward the clusterhead requests if it is in the same cluster. Hence, in this example, N7's clusterhead request cannot reach N3 because N4 will not forward it. N7's handshake process will timeout and is restarted, while excluding N3 from the selection pool. In this way, we make sure that all the constructed clusters are connected ones:

Theorem 2: For any node N_i in a cluster C with the clusterhead N_h, there exists a path from N_i to N_h, such that the path only contains nodes in C.

Proof: If N_c is 1-hop neighbor of N_h, the theorem is self-explanatory. Because only through the nodes which also select N_h as clusterhead, can N_c complete the handshake process with N_h, N_c must have at least one of its 1-hop neighbors which is also in C. ■

D. Construction of local CANs

Every cluster in CASD is also constructed to be a local CAN for distributed storage of service indices. We extend local CANs by letting the clusterheads allocate the virtual coordinate space when nodes join or leave the cluster, and enabling local CANs to store service indices with controllable redundancy as described in Section V-B.

The construction of a local CAN is done along with the handshake process between the cluster members and their selected clusterheads. In a local CAN, only the clusterhead knows and maintains the entire virtual space, and each cluster member only knows its neighbors in the virtual space. Originally, the clusterhead owns the entire virtual space. When a clusterhead receives the joining request from another node N_j, it allocates a new zone for N_j by splitting the current largest zone occupied by node N_s in the local CAN into two halves. After that, N_j, N_s, and all the CAN neighbors of the two nodes, will be notified for their updated CAN neighbor lists. N_j and N_s will also be notified with their new CAN zones.

A node can leave its belonging cluster due to node mobility, or become unavailable when it is out of power. The clusterhead detects such node unreachability via localized bilateral beaconing described in Section IV-E below, and selects the node N_t with the smallest zone in the virtual coordinate space of the local CAN to take over the zone occupied by the unreachable node N_l. After such space reallocation, all the CAN neighbors of N_l, besides N_t, will be notified with their updated CAN neighbor lists. N_t will also be notified with its new zone.

Because it is always the largest zone in the local CAN which is split to add the new joining node, CASD can guarantee that the zones in the virtual coordinate space of a local CAN are uniform, such that the deviation of the zones' sizes in a local CAN is always bounded:

Theorem 3: At any time, the deviation between the sizes of the largest and smallest virtual zone in a local CAN will not exceed $1/2^k$, where $k = \lceil \log_2^n \rceil$, and n is the current number of nodes in the local CAN[2].

Proof: In the virtual coordinate space, suppose the smallest zone size is S_{min} after the nth node is added, then the zone having been split to add the nth node must has size $2S_{min}$. Hence, the deviation between the sizes of the largest and smallest virtual zone must be S_{min}. Because $2^k \geq n$, $S_{\min} \leq 1/2^k$. ■

Actually, the deviation will be $1/2^k$ when $k = \lceil \log_2^n \rceil \neq \log_2^n$. When $k = \log_2^n$, the deviation is 0, because the entire virtual coordinate space is divided evenly into n zones, each of which has the size $1/2^k$.

E. Cluster maintenance

CASD utilizes localized bilateral beaconing to maintain the multi-hop clusters. A clusterhead multicasts beaconing messages periodically at the interval T_{cb} to all the cluster members, where T_{cb} is set to be the same as the hello beaconing interval T_H. Every cluster member returns an acknowledgment to its clusterhead upon receiving the beacon message. Only the nodes within the same cluster will forward the beaconing messages and acknowledgments.

If a non-clusterhead node has not received the beacon message from its current clusterhead for a time period longer than $2T_{cb}$, it considers its current clusterhead unreachable, and reselects a clusterhead. If the clusterhead has not received the acknowledgment from a cluster member for a time period longer than $2T_{cb}$, the clusterhead considers the member unreachable and deletes the member from its member list.

V. Hierarchical Service Discovery

In CASD, the clusterheads are used to represent their belonging clusters on the network, and together form a virtual backbone for disseminating service discovery messages. Through the virtual backbone, service discovery is conducted among different clusters in a hierarchical way, by choosing appropriate service repositories in the local CANs, and disseminating service discovery messages to the repositories. For a service announcement, the receiving repositories store the service descriptions in the message as indices, and for a service request, the receiving repositories conducts service matchmaking and reply to the requester if matching service indices are found.

The virtual backbone is constructed in an on-demand manner, i.e., it is constructed and updated along with the dissemination of service discovery messages in the network, after the multi-hop clusters are formed.

A. Dissemination of service discovery messages and construction of virtual backbone

On one hand, within each cluster, service discovery messages are forwarded, as intra-cluster ones, directly to the

[2]Suppose that the range of each dimension of the coordinate space is (0,1), and the total size of the space is 1.

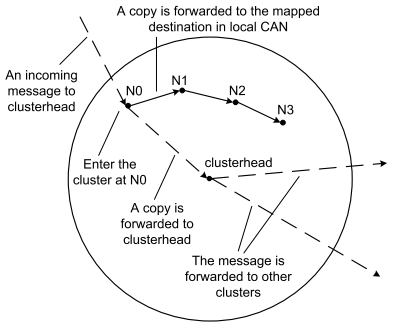

Fig. 3. Forwarding an inter-cluster service discovery message

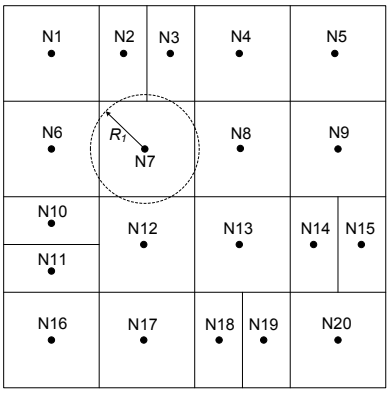

Fig. 4. Forwarding service announcements to multiple destination for controllable redundancy

hashed destination points in the local CAN. On the other hand, the messages are also disseminated as inter-cluster ones to all the clusters through the virtual backbone. CASD marks a service discovery message to be an inter-cluster one by setting an INTER_CLUSTER flag in it.

Every service discovery message is initially an intra-cluster one. The originator of the message sends the message to the destination point in the local CAN according to the local hashing result. Meanwhile, a copy of the message is also sent directly to its clusterhead. When the sender's clusterhead receives the message, it starts an inter-cluster disseminating process for this message.

The initialization of the virtual backbone is conducted on a clusterhead when it is going to send out the first inter-cluster service discovery message. A clusterhead N_a broadcasts this first message to its neighborhood to discover its virtual neighbors. The nodes within the same cluster continue to broadcast the message, and the nodes in a different cluster directly forward the message to their clusterhead N_b. Upon receiving this message, N_b is aware of the existence of its virtual neighbor N_a, and N_b will also sends a notification to N_a to announce itself.

On the virtual backbone, a clusterhead N_a disseminates service discovery messages to other clusters through a unique multicast tree rooted at N_a itself. Such a multicast tree is stored in a distributed manner, in the form of $\{N_b, N_a, S\}$, which means that, when N_b receives a message from N_a, it forwards the message to the nodes in set S, which contains a part of its virtual neighbors. Initially, S includes all the virtual neighbors of N_b, and N_b forms its multicast tree by pruning redundant nodes from S when having received repetitive service discovery messages.

When an inter-cluster service discovery message enters a different cluster, it is forwarded to the hashed destination point in the local CAN, and to the clusterhead. The clusterhead updates the backbone according to this message, and continues to multicast the message to other clusters. Such process is illustrated in Fig. 3.

B. Multiple forwarding destinations in a local CAN

Robust service discovery in CASD is conducted by maintaining controllable redundancy when storing service indices in local CANs. In CASD, such redundancy is achieved by forwarding service announcements to multiple destinations in each local CAN, and keeping duplicates for each service index.

The number of forwarding destinations for a service announcement in each local CAN is determined by its redundancy degree d_r, which is defined as the proportion of the forwarding destinations to the total number of nodes. d_r is decided independently in each cluster to be adaptive to different local network conditions.

When the virtual point mapped from a service announcement falls into a virtual zone, a circular area centered at the center point of the rectangular virtual zone is determined, and the service announcement will be forwarded to all the nodes whose virtual zones overlap with this circular area. The radius of this circular area, named $R1$, is selected appropriately to provide controllable redundancy degree, as stated in Theorem 4. Such forwarding process is illustrated by an example in Fig. 4, in which a service announcement received by N7 will be forwarded to N2, N3, N6, N8 and N12.

Theorem 4: The redundancy degree d_r in a local CAN has a lower bound of a function of R_1.

Proof: Without loss of generality, we consider the case in a 2-dimensional virtual coordinate space. When there are n nodes in a local CAN, according to Theorem 3, the smallest zone S_{min} in the local CAN will not exceed $1/2^k$, where $k = \lceil \log_2^n \rceil$. If we have a circular area C_1 with the radius R_1, the size of such area will be πR_1^2. Therefore, $d_r \geq \pi R_1^2 / S_{\min} \geq \pi 2^k R_1^2 / n$. Since $k \geq \log_2 n$, we can also have $d_r \geq \pi R_1^2$. ■

Service discovery is required to find all the services matching the service request in the network. Because of the deviation between the services provided and requested, a service request needs to be forwarded to all the service repositories that possibly contain the matching service indices. The acceptable deviation range in service matchmaking can be represented as the radius, named $R2$, of a circular area *C*, which centered at the destination point mapped from the service request. The forwarding process of a service request consists two steps:

1) The service request is forwarded to all the nodes whose zones overlap with *C*. The set of the forwarding destinations is called *S*.
2) Each node in *S* determines its forwarding destinations for controllable redundancy, as described above. The service request will be forwarded to all of such destinations.

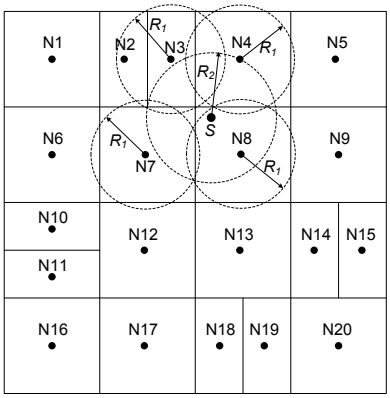

Fig. 5. Forwarding service requests to multiple destinations to ensure that all the matching services are found

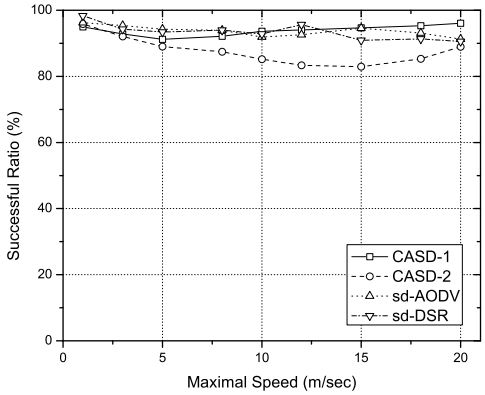

Fig. 6. Successful ratio in different mobility settings

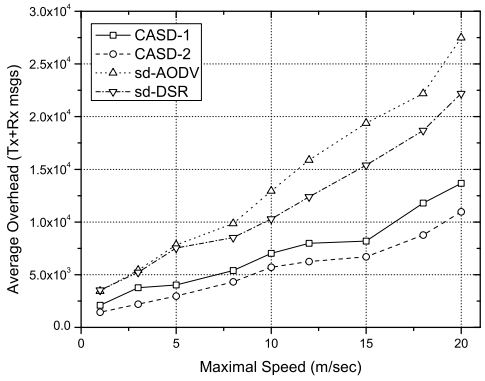

Fig. 7. Average overhead in different mobility settings

Such two steps are illustrated by an example in Fig. 5. When a service request with the virtual point is S is received by N8, it will be forwarded to N3, N4 and N7 in the first step. In the second step, the destinations are N2 (by N3), N5 (by N4), N6 and N12 (by N7), and N9 and N13 (by N8).

VI. Performance Evaluation

We implemented our CASD based on the wireless extensions of ns-2. We compared CASD with service discovery in flat network architecture to illustrate the performance, scalability and robustness of CASD in different network settings. We also showed that CASD can be adaptive to various network applications by adjusting the cluster radius accordingly.

A. Simulation settings

We assume all the nodes use 802.11 MAC mechanisms with idealized features. All the simulations will be conducted as long as T=5000secs to eliminate short-time randomness. The node mobility follows the random-walk mobility model [17], with the node moving speeds normally distributed in a range $[0, v_{max}]$. The communication ranges of nodes are uniformly distributed between $200m$ and $300m$. Basically, we deploy 50 nodes in a $1000 \times 1000m^2$ area, and in the scalability evaluation the area is changed proportionally to the network scale. We assume that every node periodically announces a service and requests for another service. The periods of service announcements and service requests are $T/20$.

We compare the performance of CASD with service discovery in flat network architecture. We have extended two representative ad-hoc routing protocols, AODV [18] and DSR [19], with service discovery functionality by enabling the nodes to process service discovery messages. We refer these two extended protocols as sd-AODV and sd-DSR respectively. We also evaluate the performance of CASD when local CANs are not used. In this case, the dissemination of service discovery messages within a cluster is done by localized broadcast, and inter-cluster service discovery is only conducted on clusterheads. We refer the CASD with local CANs as CASD-1, and the CASD without local CANs as CASD-2.

B. Performance in different mobility settings

In this section, the network scale is fixed to be 50 nodes, and we vary v_{max} from 1m/sec to 20m/sec. 2-hop clusters are constructed in CASD.

The simulation results are shown in Figs. 6 and 7. Service discovery in flat network architecture can maintain constant successful ratio of service discovery above 90% when the network mobility increases, because in such network architecture, service discovery messages are disseminated by global flooding, and thus are guaranteed to arrive the entire network. Fig. 6 shows that CASD can achieve similar successful ratio. Intra-cluster and inter-cluster beaconing mechanisms ensure CASD to detect network topology changes on time, and adjust the network architecture accordingly. Besides, construction of multi-hop clusters in CASD is done in an autonomous manner, which localizes the influence of node mobility.

Service discovery in flat network architecture achieves satisfying successful ratio of service discovery at the cost of the overhead increasing dramatically with the increase of network mobility, which is shown in Fig. 7. Instead, the overhead in CASD can be controlled to increase slowly with the increase of network mobility, because of the dissemination of service discovery messages in a hierarchical manner. Specifically, Fig. 7 also shows that, using local CANs in CASD will bring some extra overhead because of the construction and maintenance of the local CANs. When the network mobility is high, the proportion of such extra overhead can be up to 20%,

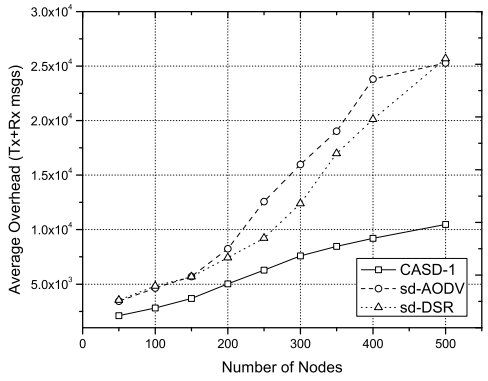

Fig. 8. Average overhead in different network scales (V_{max} = 1 m/sec)

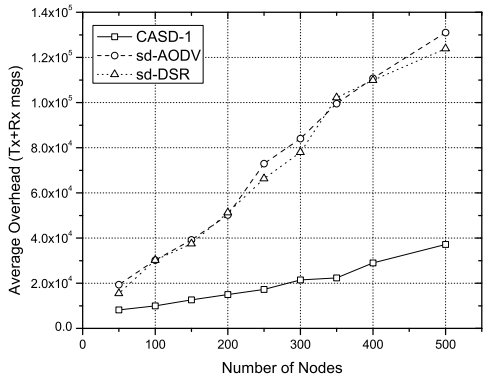

Fig. 9. Average overhead in different network scales (V_{max} = 15 m/sec)

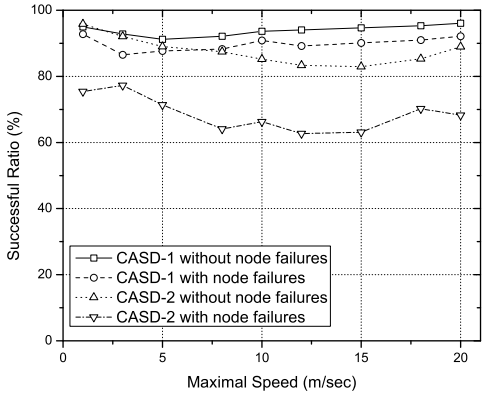

Fig. 10. Successful ratio in cases with and without node failures

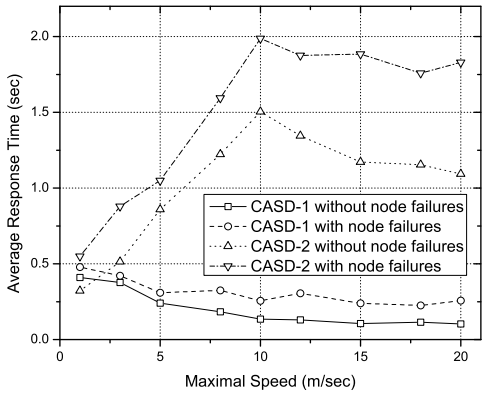

Fig. 11. Response time in cases with and without node failures

because the network topology changes and the corresponding reallocations of local CANs will be very frequent.

C. Scalability evaluation

We evaluated the performance of CASD in different network scales, in which the number of network nodes varies from 50 too 500, with service discovery in flat network architecture. Compared to flat network architecture, CASD can achieve scalable service discovery in ubiquitous computing environments by eliminating broadcast of service discovery messages and distributing storage of service index in local CANs. Fig. 8 and Fig. 9 show that, when the network scale increases, CASD can control the increasing speed of the overhead for service discovery at a low level, compared to flat network architecture, in which such overhead is increasing exponentially. Because every service discovery message in CASD is forwarded to an arbitrary destination only on one route, larger network scale will only increase the length of such forwarding routes, and hence only cause linear increase of the overhead of service discovery. Such advantage is especially apparent in high-mobility network settings, as shown in Fig. 9. When the maximal moving speed of nodes in the network increases to 15 m/sec, the overhead in flat network architecture can increase up to 6 times as it is in low-mobility settings, but the overhead in CASD only increase up to 100%.

D. Robustness evaluation

We evaluate the robustness of CASD by comparing the performance of service discovery in cases with and without node failures in different network mobility settings. In cases with node failures, each node in the network has a probability of 50% to be failed for a fixed length of time. In our simulations, such time is set to 10% of the total simulation length. Once having been into failure, a node will be a “black hole” for all the incoming packets, and all of its local records about the local CAN and virtual backbone are cleaned.

The simulation results are shown in Fig. 10 and Fig. 11. In CASD with local CANs, because of the redundancy storing service indices, the user can still find the desired services in an arbitrary cluster, even if some storage is unavailable due to single points of node failures. Fig. 10 shows that, in cases with node failures, CASD-1 only suffers 5% degradation in the successful ratio of service discovery. Meanwhile, since the number of duplicates for a service index in a cluster is controlled, the unavailability of some of the duplicates will only cuase 10% more delay in service discovery.

The importance of local CANs for the robustness of service discovery is also shown. In the absence of local CANs, all the information of services outside of a cluster will be stored on the clusterhead of that cluster. If this clusterhead becomes unavailable, the performance of CASD will be degraded seriously. In Fig. 10 and Fig. 11, the successful ratio of service discovery can be down to 60%, and the response time can be

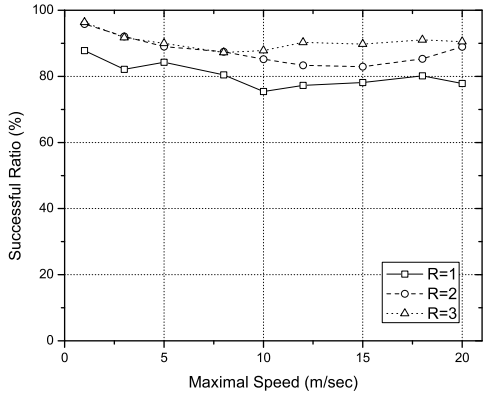

Fig. 12. Successful ratio with different cluster radius

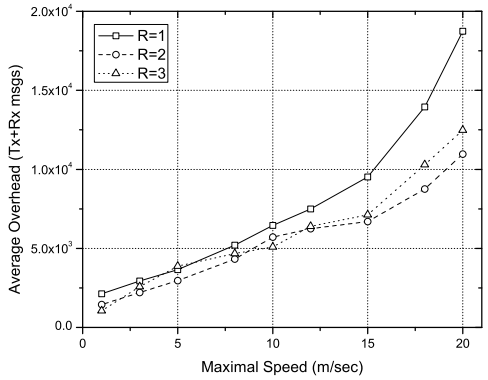

Fig. 13. Average overhead with different cluster radius

50% longer, especially in high-mobility network settings.

E. Selecting different cluster radius

Multi-hop clusters used in CASD enable the user to choose different cluster radius to satisfy the requirements of different ubiquitous computing applications. We set the network scale to 100 nodes, and vary the cluster radius from 1 to 3. The simulation results under different network mobility settings are shown in Figs. 12 and 13. Compared to 1-hop clusters, multi-hop clusters have shown the advantage in all performance metrics we are using. Multi-hop clusters are more advanced in high mobility settings, because 1-hop clusters are more volatile and need to conduct more clusterhead reselection when the mobility is high. Meanwhile, 3-hop clusters in CASD are able to provide higher successful ratio and shorter average response time, but can also lead to larger overhead of service discovery, because of the maintenance of clusters and the local CANs. Hence, cluster radius needs to be selected appropriately according to the network scenario and ubiquitous computing applications, to achieve better performance. Larger clusters are suitable when the network is relatively stable, and the service discovery is required to be highly accurate and timely. Oppositely, smaller clusters should be used in highly dynamic and energy-constrained environments.

VII. Conclusion

In this paper, we presented a multi-hop Cluster-based Architecture for scalable and robust Service Discovery (CASD) in ubiquitous computing environments. CASD organizes the entire network as multi-hop clusters based on the NB scores of mobile nodes, constructs clusters to be extended CANs for service repositories, and connects them together to be a virtual backbone for hierarchical service discovery. Our simulation results show that, compared to approaches based on flat network architecture, CASD is able to provide scalable and robust service discovery in various types of network scenarios, and is able to choose different cluster radius according to the node density and network scale to achieve better performance in different network applications.

References

[1] F. Zhu, M. Mutka and Lionel M. Ni, “Service Discovery in Pervasive Computing Environments”, *IEEE Pervasive Computing*, vol. 4(4), 2005, pp. 81-90

[2] H. Kil, S.-C. Oh, and D. Lee, “On the Topological Landscape of Web Services Matchmaking”, *Proc. Int’l Workshop on Semantic Matchmaking and Resource Retrieval (SMR)*, 2006, pp. 19-34.

[3] S. Ratnasamy, et al., “A Scalable Content-Addressable Network”, *Proc. ACM SIGCOMM*, 2001, pp. 161-172.

[4] Sun Microsystems, “Jini Architecture Specification v1.2”, 2001, http://www.sun.com/software/jini/specs

[5] Microsoft Corporation, “Universal Plug and Play Architecture v1.0”, 2000, http://www.upnp.org/download/UPnPDA10-20000613.htm

[6] U. Kozat and Leandros Tassiulas, “Network Layer Support for Service Discovery in Mobile Ad Hoc Networks”, *Proc. IEEE INFOCOM*, vol. 3, 2003, pp. 1965-1975.

[7] A. Helmy, et al., “CARD: A Contact-based Architecture for Resource Discovery in Wireless Ad Hoc Networks”, *Mobile Networks and Applications*, vol. 10, 2005, pp. 99-113.

[8] I. Stoica, et al., “Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications”, *IEEE/ACM Transactions on Networking*, vol. 11(1), 2003, pp. 17-32.

[9] Q. Xia, et al., “Fully Decentralized DHT based Approach to Grid Service Discovery using Overlay Networks”, *Proc. 5th International Conference on Computer and Information Technology (CIT)*, 2005, pp. 1140-1144.

[10] X. Hong, et al., “Scalable Routing Protocols for Mobile Ad Hoc Networks”, *IEEE Network Magazine*, vol. 16(4), 2002, pp. 11-21.

[11] J. Wu, and H. Li, “On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks”, *Proc. 3rd Int’l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 1999, pp. 7-14.

[12] D. Kim, S. Ha, and Y. Choi, “k-hop cluster-based dynamic source routing in wireless ad-hoc packet radio networks”, *Proc. IEEE Vehicular Technology Conference (VTC)*, 1998, pp. 224-228.

[13] F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, “Connectivity based k-hop clustering in wireless networks”, *Telecommunication Systems*, vol. 22(1-4), 2003, pp. 205-220.

[14] S. Yang, J. Wu and J. Cao, “Connected k-hop Clustering in Ad Hoc Networks”, *Proc. International Conference on Parallel Processing (ICPP)*, 2005, pp. 373-380.

[15] I. I. Er and W. K. G. Seah, “Mobility-based d-hop Clustering Algorithm for Mobile Ad Hoc Networks”, *Proc. WCNC*, 2004, pp. 2359-2364.

[16] B. An and S. Papavassiliou, “A Mobility-based Clustering Approach to Support Mobility Management and Multicast Routing in Mobile Ad-hoc Wireless Networks”, *International Journal of Network Management*, vol. 11(6), 2001, pp. 387-395.

[17] A. McDonald, and T. Znati, “A mobility-based framework for adaptive clustering in wireless ad hoc networks”, *IEEE Journal on Selected Areas in Communications*, vol. 17(8), 1999, 1466-1487.

[18] C.E. Perkins, and E. M. Royer., “Ad-Hoc On-Demand Distance Vector Routing”, *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90-100.

[19] D. Johnson, and D. A. Maltz, “Dynamic Source Routing in Ad-Hoc Wireless Networks,” *Mobile Computing*, Kluwer Academic Publishers, 1996, pp. 153-181.