Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2007, Article ID 25167, 13 pages
doi:10.1155/2007/25167

Research Article

SPIZ: An Effective Service Discovery Protocol for Mobile Ad Hoc Networks

Donggeon Noh and Heonshik Shin

School of Computer Science and Engineering, College of Engineering, Seoul National University, Seoul 151742, Korea

Received 1 February 2006; Revised 19 July 2006; Accepted 16 August 2006

Recommended by Hamid Sadjadpour

The characteristics of mobile ad hoc networks (*MANETs*) require special care in the handling of service advertisement and discovery (*Ad/D*). In this paper, we propose a noble service *Ad/D* technique for *MANETs*. Our scheme avoids redundant flooding and reduces the system overhead by integrating *Ad/D* with routing layer. It also tracks changing conditions, such as traffic and service popularity levels. Based on a variable zone radius, we have combined push-based *Ad/D* with a pull-based *Ad/D* strategy.

1. INTRODUCTION

As computer networks and their applications become central to everyday life, users demand an efficient way to locate services (the services of a network are made up of many kinds of software and hardware components, related to data, information, computational devices, storage, and the network itself) over a network. This is particularly true for self-configurable networks which must be easy to deploy and to reconfigure automatically when they are extended with new hardware and software capabilities. Mobile ad hoc networks (*MANETs*) are a special form of self-configurable network, with their own dynamics, resource constraints at the constituent nodes, and no centralized management mechanism.

Because of these characteristics, the development of service discovery strategies for *MANET*s poses interesting challenges; and existing advertisement and discovery (*Ad/D*) frameworks developed for well-structured networks, such as *Jini*, *SLP*, and *UPnP*, are not suitable. The major challenges in providing a service *Ad/D* for *MANET*s are as follows.

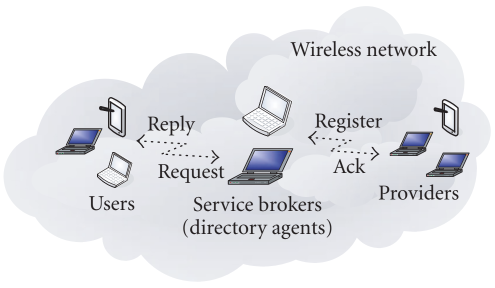
(i) To enable resource-constrained, wireless devices to discover services dynamically, while minimizing both the control traffic and latency.
(ii) Supporting large-scale *MANET*s composed of hundreds of nodes.
(iii) Providing lightweight service discovery for resource-poor constituent nodes.
(iv) Delivering services to a wide spectrum of devices, regardless of their *H/W* and *S/W* platform.

To meet these requirements, we present *SPIZ* (a service *Ad/D* protocol with independent zones). *SPIZ* uses existing network layer control packets, and therefore offers a lightweight implementation of service *Ad/D* and avoids unnecessary overhead. It also incorporates a zone radius determination algorithm for adaptive hybrid service *Ad/D*, which makes allowance for the network characteristics (i.e., mobility and call rate) and the popularity of each service. Additionally, *SPIZ* provides an efficient pull-based service discovery mechanism using bordercasting for on-demand (i.e., pull-based) service finding. These characteristics allow *SPIZ* to support *Ad/D* with a relatively low overhead and latency, making it applicable to large-scale *MANET*s (i.e., those with at least 100 nodes), unlike other directoryless *Ad/D* schemes.

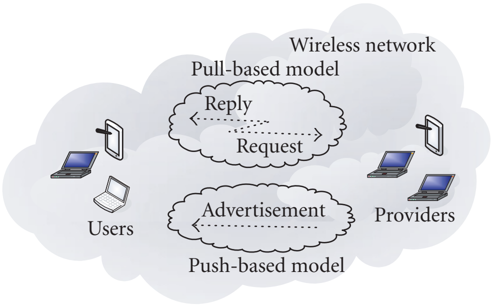
The rest of this paper is organized as follows. Section 2 contains an analysis of existing service *Ad/D* strategies for *MANETs*. Section 3 describes the characteristics of our new lightweight hybrid adaptive service *Ad/D* strategy. We then give an overview of the simulation environment and present an evaluation of our strategy in Section 4. Finally, conclusions are drawn and future work is discussed in Section 5.

2. SERVICE *Ad/D* FOR *MANET*

Existing service *Ad/D* schemes for *MANET*s can be classified into two architectures: one uses a directory model, and the other a directoryless model. The directory model [1, 2] involves service brokers (or directory agents) which are logical entities residing between clients and servers. Clients direct



(a) Directory service *Ad/D* model



(b) Directoryless service *Ad/D* model

Figure 1: Service *Ad/D* architecture.

their requests to well-known service brokers with which servers have registered their services. The brokers then send service reply messages back to the clients and registration acknowledgments to the servers. This scheme is good for scalability and response time, but imposes an extra load on the network because the selection of service brokers is a process that must adapt to topology changes, and the rest of the network must be kept informed about the identities of the broker nodes. If there is high mobility in the network, so that its topology changes very frequently, it is too expensive to maintain service brokers. Therefore, a directory scheme should only be adopted for a *MANET* which is relatively static (e.g., a wireless home network).

In the directoryless model, users actively send out service request messages and each server listens to these messages at a well-determined network interface and port. If the requested service is supported, then a reply is generated and sent back to the clients. Users can also learn about available services in a passive way by listening to service advertisements that are generated by the servers.[1] It can be argued that the directoryless model is more suited to a *MANET* scenario, because it is fully distributed and there is no need for any infrastructure. Figure 1 presents a synopsis of the architectural choices and the control messages required to support each type of model.

[1] We will refer to active service *Ad/D* as the push-based model and to passive *Ad/D* as pull-based. The hybrid service model is a combination of these two approaches.

In several existing directoryless service *Ad/D* implementations, service *Ad/D* models are implemented in the middleware layer [3, 4], with the support of underlying ad hoc routing protocols. However, both the *Ad/D* protocol and the routing protocol can invoke redundant flooding of the network, and this inevitably incurs a large overhead. In any case, these are heavyweight solutions, because they must be implemented as an independent layer. These are serious drawbacks in *MANET*s, in which there is often a shortage of network and computing power.

A consideration of these problems motivates the integration of service *Ad/D* protocols with routing protocols. In this approach, service *Ad/D* information is piggybacked on to extended routing control packets in order to reduce the necessity for flooding. Service *Ad/D* has already been integrated with a proactive routing protocol [5] and also with a reactive routing protocol [6–8]. More recently, a simple hybrid service *Ad/D* protocol [9, 10] has been designed.

Although these protocols represent some progress, none of them includes an adaptation strategy sufficiently suited to a dynamically changing network environment, which is one of the most significant characteristics of a *MANET*. In particular, the radius of the push-based service zone is simply determined by the transmission range [10] or by the popularity of the service [9]. Furthermore, none of the existing protocols is suitable for large-scale *MANET*s.

In summary, the shortcomings of existing directoryless *Ad/D* protocols include a poor ability to adapt to dynamic network changes (e.g., mobility and call rate level in the network, popularity level of the services), low efficiency of service discovery algorithms, and dubious scalability.

We addressed these problems in earlier work [11]. In this paper, we will expand and elaborate on the ideas behind our hybrid and adaptive resource discovery strategy for wireless adhoc networks.

3. *SPIZ*

SPIZ (a service *Ad/D* protocol with independent zones) is an adaptive hybrid service *Ad/D* protocol integrated with an efficient hybrid routing protocol [12]. Figure 2 summarizes distinguished characteristics of *SPIZ*. It allows nodes to adapt their own zone radii dynamically and autonomically to changing conditions, such as mobility and popularity levels. Based on a variable zone radius, *SPIZ* combines push-based *Ad/D* with a pull-based *Ad/D* strategy that uses an efficient bordercasting resolution protocol. Integration with the network layer protocol is intended to result in a lightweight scheme and to reduce the amount of unnecessary network flooding.

3.1. *A hybrid service Ad/D strategy with network layer support*

Redundant flooding operations by the middleware-oriented service *Ad/D* strategies can expose serious deficiencies in *MANET* environments, which are by nature poor in resources. By piggybacking the service information on the

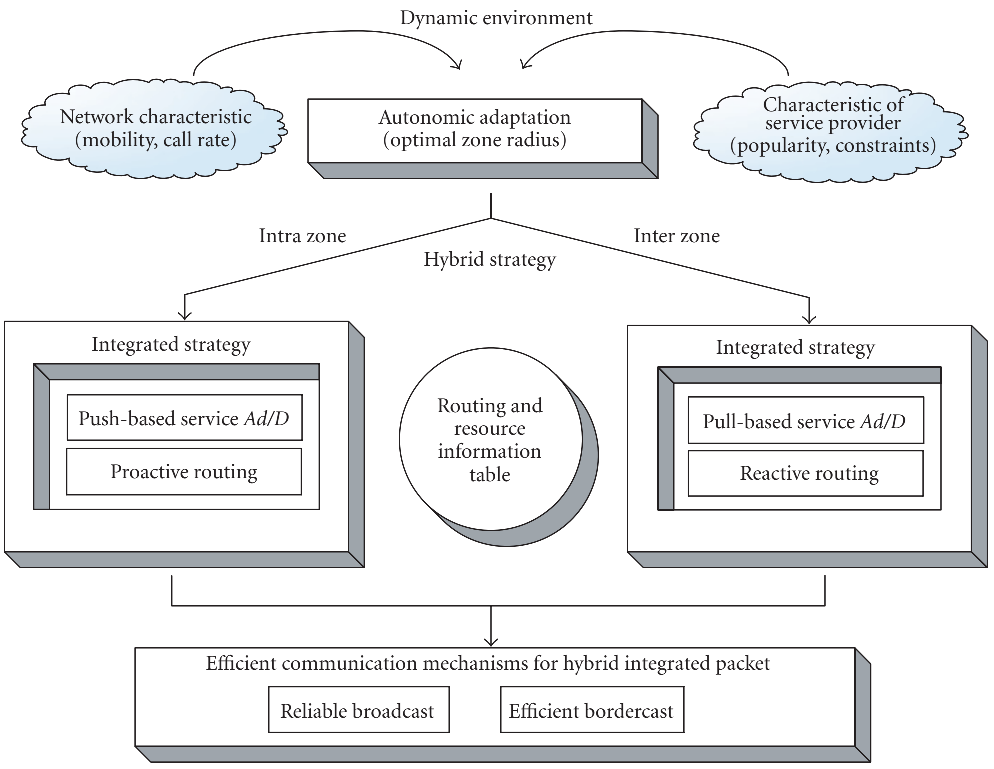

Dynamic environment
Network characteristic (mobility, call rate)
Autonomic adaptation (optimal zone radius)
Characteristic of service provider (popularity, constraints)
Intra zone
Inter zone
Hybrid strategy
Integrated strategy
Push-based service Ad/D
Proactive routing
Routing and resource information table
Integrated strategy
Pull-based service Ad/D
Reactive routing
Efficient communication mechanisms for hybrid integrated packet
Reliable broadcast
Efficient bordercast

FIGURE 2: Overview of the *SPIZ* strategy.

network layer control packet, we can implement a lightweight *Ad/D* scheme which can simultaneously obtain the service and routing information for an expected service provider, thus saving system resources.

In addition, *SPIZ* uses a hybrid service model which allows a node to perform push-based *Ad/D* in its own zone, and pull-based *Ad/D* outside that zone. This is made possible by integrating the service *Ad/D* protocol with a hybrid routing protocol. Previous studies of routing in a *MANET* [12, 13] have shown that a hybrid routing protocol is more efficient than simple proactive or reactive protocols. We therefore expect that integrating the service *Ad/D* protocol with a hybrid routing protocol will achieve a more efficient service *Ad/D* model.

The architecture of the framework for integrating hybrid *Ad/D* and network layer routing is outlined in Figure 3, which introduces an extended version of the IARP (intra routing protocol) packet, that is able to carry service information piggyback, and which we call the *IAIP* (intra integrated protocol) packet. The *IERP* (inter routing protocol) packet is likewise expanded to become the *IEIP* (inter integrated protocol) packet. Query is performed by *IEIP* packets, whereas *IAIP* packets are used for advertising services.

As shown in Figure 3, when an application-layer program wants to find the specific service object which satisfies given constraints, it sends a request to the service *Ad/D* processing module, which checks the service information table. If there is no information about the corresponding service provider, the *Ad/D* processing module asks the routing module to construct an appropriate routing packet and to pass the service information received from the application layer to the integration module. Finally, the integration module assembles an integrated packet (i.e., *IEIP*) which contains both the service and routing information. When a node receives an integrated packet, it sends it to the integration module. This separates the routing-related and service-related information, which are then managed by the routing and service *Ad/D* modules, respectively.

3.2. *Autonomous adaptive zone radius*

In *SPIZ*, each node determines the radius of its own zone independently, while taking into account the state of the network (i.e., call rate, mobility). Service providers must also consider popularity (i.e., the number of service invocations). For example, if the network has a high call rate and low mobility, most nodes should have relatively large zone radii, in order to perform effective routing and service *Ad/D* with minimum impact on the network overhead and latency. But when the network has a relatively low call rate and high mobility, smaller zones are more effective. Additionally, the provider of a more popular service operates more efficiently with a larger zone.

Since *SPIZ* integrates the service *Ad/D* protocol with the network layer protocol, the zone of a service provider node is now the push-based *Ad/D* zone as well as the proactive routing zone. But the zone of a service user node is only the proactive routing zone, and does not include push-based

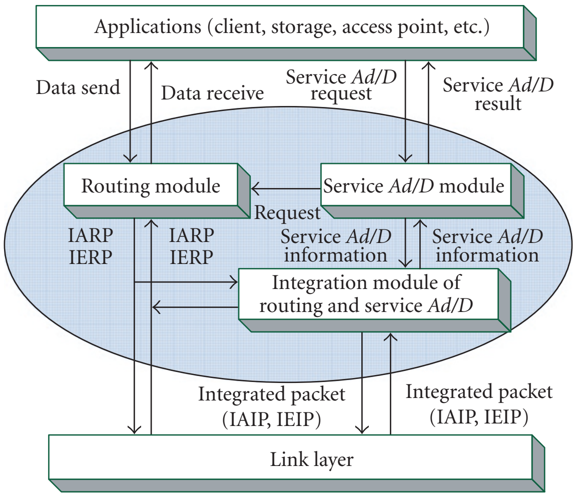


FIGURE 3: Hybrid service *Ad/D* framework with network layer support.

Ad/D, since a user has nothing to advertise. Nevertheless, the zone of a user node is still significant in the service *Ad/D* process, because it determines the base from which we can perform effective on-demand service discovery, as explained in Section 3.4.

3.2.1. Determination of zone radius

We determine zone radii using a hybrid of the *Min_Searching* and *Adaptive_Traffic_Estimation (ATE)* algorithms [14], tailored to integrated *Ad/D*. The proactive traffic and push-based *Ad/D* traffic of a node is a nondecreasing function of its zone radius, and the reactive traffic and pull-based *Ad/D* traffic is a nonincreasing function of the same radius. Hence, the total integrated control traffic, which is a sum of these two components, is a convex function. Figure 4(a) shows how the *IAIP*, *IEIP*, and total control traffic vary with zone radius. In this figure, the total traffic is a minimum when the zone radius is 3.

At each node, the *Min_Searching* algorithm can find the minimum point on the integrated control traffic curve by repeated refinement of the zone radius (ρ) in increments and decrements of one hop ($\Delta\rho = \pm 1$), as illustrated in Figure 4(a).[2] More specifically, each node estimates the integrated control packet traffic ($T\rho(k)$) at each time step, k. If the amount of traffic has fallen ($T\rho(k) < T\rho(k-1)$) such as step (1), step (2), and step (4), the next change to the radius will be in the same sense ($\rho(k+1) = \rho(k) + \Delta\rho$); if the traffic has increased such as step (3) and step (5), the radius is adjusted in the opposite direction ($\Delta\rho = -\Delta\rho$). *Min_Searching* will find the minimum ($T\rho_{\text{opt}}$), provided that the network behavior does not change substantially while the algorithm is running. If the minimum is found correctly by *Min_Searching*, the cost incurred is

$$C_{\text{success}} = \sum_{\rho=1}^{\rho_{\text{opt}}+1} \left[T\rho - T\rho_{\text{opt}}\right], \qquad (1)$$

where $T\rho$ is the traffic during a period of time when the zone radius is ρ. This cost occurs because *Min_Searching* cannot determine the optimal radius immediately.

However, *min_searching* becomes less effective if the network conditions change while it is running. Therefore, the length of a time step must be determined prudently. If it is too short, then accurate measurement of the traffic is difficult; if it is too long, the probability of changes occurring in the network is increased.

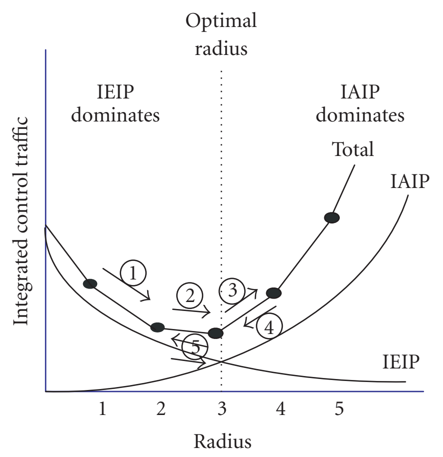
Once the lowest point on the control traffic curve has been found, the ratio of the *IEIP* component to the *IAIP* component at the optimal zone radius is set to Γ_{thres}, which is periodically used by the *ATE* algorithm to tune the zone radius. Let $\Gamma(R)$ be the ratio of *IEIP* traffic to *IAIP* traffic, measured at a network node during an estimation interval when the zone radius is R. Simplistically, we could now compare $\Gamma(R)$ with Γ_{thres} to determine whether the zone radius should shrink or grow. If $\Gamma(R) < \Gamma_{\text{thres}}$, the current status corresponds to *IAIP* domination status as illustrated in the right-hand of Figure 4(a). So the zone radius must be decreased by one hop to decrease *IAIP* traffic. Contrastively, if $\Gamma(R) > \Gamma_{\text{thres}}$, the zone radius must be increased by one hop to decrease the dominance of *IEIP* traffic. However, since frequent changes of zone radius can make the network unstable, a delayed triggering mechanism is introduced by the use of a multiplicative hysteresis term, δ ($\delta \geq 1$). As illustrated in Figure 4(b), if $\Gamma(R) > \Gamma_{\text{thres}} \bullet \delta$ such as region (3), then the zone radius is increased by one hop; if $\Gamma(R) < \Gamma_{\text{thres}}/\delta$ such as region (1), the zone radius is decreased by one hop. The larger value of δ makes a larger margin of changing zone radius, which makes region (2) larger. It means a slow change of zone radius.

When the *ATE* algorithm is being run by a service provider, the popularity of the service must be considered, in addition to the network state. For this reason, a service provider periodically monitors the frequency of invocation of its service. As shown in Figure 4(c), if P_{new} (the invocation frequency during the current period) is higher than P_{old} (the invocation frequency during the last period) such as region (3), then the zone radius is increased by one hop to decrease the pull-based *Ad/D* traffic, and vice versa. Again, a delayed triggering mechanism is used to prevent frequent changes of zone radius. The multiplicative hysteresis term is ε ($\varepsilon \geq 1$) determining the width of region (2). The appropriate values of δ and ε for a particular environment can be obtained from several experiments.

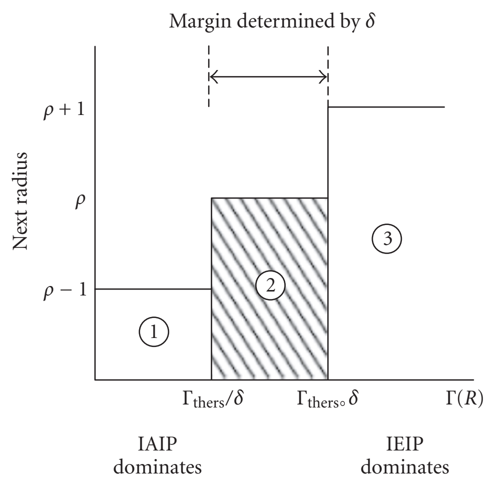
Note that service providers and service users use an identical *ATE* algorithm. However, for a service user, the popularity parameter is fixed since it is not meaningful.

This hybrid algorithm of *Min_Searching* and *ATE* allows each node to adapt to dynamic changes in the network environment with little computational overhead. Algorithm 1

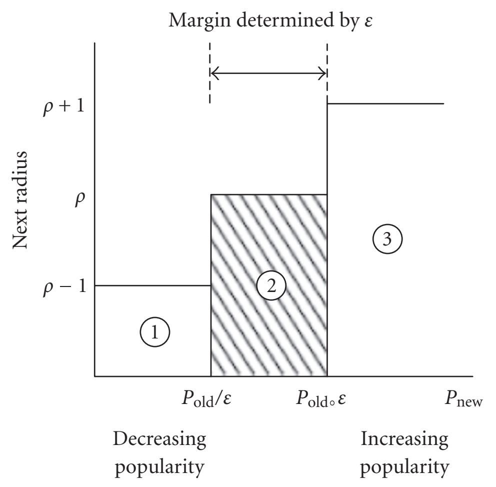
[2] The initial values of ρ and $\Delta\rho$ are both set to 1.



(a) *Min_searching*


Margin determined by δ
$\rho + 1$
ρ
$\rho - 1$
Next radius
$\Gamma_{\text{thers}}/\delta$
$\Gamma_{\text{thers}\circ}\delta$
$\Gamma(R)$
IAIP dominates
IEIP dominates

(b) *Adaptive_traffic_estimation (ATE)*



(c) Additional part of *ATE* for service provider

Figure 4: The hybrid zone determination algorithm used by *SPIZ*.

```
Min_Searching ()

if Estimate_Traffic (Period) == REDUCED then
    if Prev_Radius < Radius then
        Change_Zone_Size (Radius + 1)
    else
        Optimal_Zone_Radius = Radius
        $\Gamma_{\text{thres}} = \Gamma$ (Radius)
        Invoke (Adaptive_Traffic_Estimation ())
    end if
else Estimate_Traffic (Period) == INCREASED then
    if Prev_Radius < Radius then
        Change_Zone_Size (Radius - 1)
    else
        Change_Zone_Size (Radius + 1)
    end if
end if

Adaptive_Traffic_Estimation ()

if $\Gamma(Radius) > \Gamma_{\text{thres}} * \delta \,\|\, p_{\text{new}} > p_{\text{old}} * \varepsilon$ then
    Change_Zone_Size (Radius + 1)
else if $\Gamma(Radius) < \Gamma_{\text{thres}}/\delta \,\|\, p_{\text{new}} < p_{\text{old}}/\varepsilon$ then
    Change_Zone_Size (Radius − 1)
end if
```

Algorithm 1: Simplified core pseudocode for the zone radius determination algorithm.

contains simplified core pseudocode for the modified *Min_Searching* and *ATE* algorithms.

3.3. *Push-based service Ad/D*

In *SPIZ*, each service provider performs push-based service advertisement in its dynamic zone. The provider periodically broadcasts an advertisement message to all nodes within its zone, and service users within that zone learn passively about the service by receiving these advertisements.

3.3.1. Message format for push-based Ad/D

In order to implement integrated push-based service advertisement, we designed the *IAIP* packet format. We will refer to an *IAIP* packet used in the *Ad/D* mechanism as an *SAM* (service advertisement message). As Figure 5 illustrates, an *SAM* is composed of two parts. One is the routing control part used by the IARP. The other is the service information part which includes the service type, the service lifetime, and additional information about the service, such as its functional interface and *QoS* (quality of service) level.

This service information part can be modified as required. Reserved space can be used for that purpose. If the target service architecture is service-oriented and based on web services, then *WSDL* (web services description language) can be used to describe the service. In this paper, we focus on the *Ad/D* architecture and not on the device-level or service-level interoperability. Therefore, we use the simple service information description shown in Figure 5.

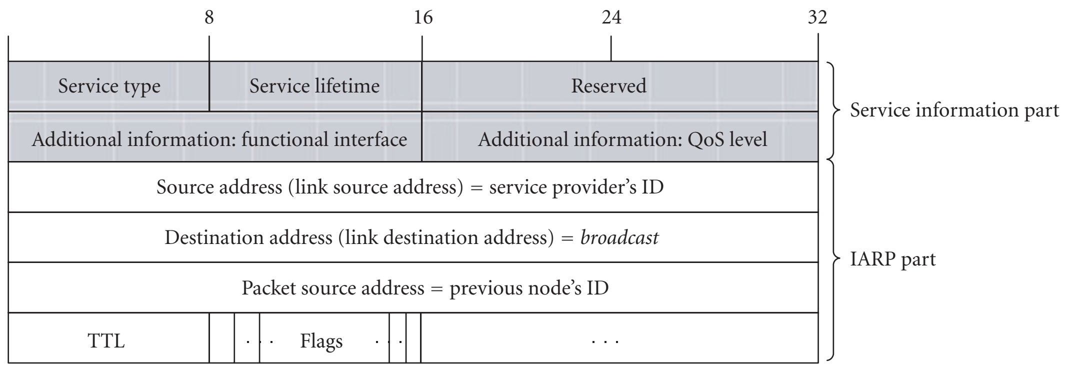

8
16
24
32
Service type
Service lifetime
Reserved
Additional information: functional interface
Additional information: QoS level
Service information part
Source address (link source address) = service provider's ID
Destination address (link destination address) = broadcast
Packet source address = previous node's ID
IARP part
TTL
Flags

Figure 5: Push-based *SAM* (service advertisement message) format.

The service type is predefined across all the nodes, and the service lifetime field is used to support the renewal cycle of the service provider. If a node which receives an *SAM* does not receive it again during the lifetime of that service, the node invalidates that service information. This allows the network to accommodate quickly to the disappearance of a service provider. The additional information field includes the functional interface and dynamic *QoS* attributes of the service. The functional interface provides the information about how to interface the service (e.g., port number, parameter, etc.). The *QoS* specification includes: (i) scalability information, which specifies the capacity of the service to serve additional requests over a specified period of time; (ii) the performance and capacities of the host, including its memory size, available energy, computation speed, and network bandwidth. This specification of *QoS* parameters is optional but, for each parameter that is provided, the following attributes must be specified: name, value, and expiration. Each attribute may have one of the following values: not supported, null, bronze, gold, platinum, or infinite.

The source address field in the IARP part of the *SAM* includes the address of the service provider and a *TTL* (time to live) field, which is initially set to its own zone radius.

With use of the *SAM*, each node maintains the essential information for the push-based service discovery and for the intrarouting, which is shown in Table 1.

Table 1: Service and routing table of each node.

Service provider (from which service Ad was received)	Service information
Nearby provider 1	Type
	Lifetime
	Additional info
⋮	⋮

Destination (all nodes within one's zone)	Routing information (Node ID list)
Member node 1	Node ID_1
	Node ID_2
	⋮
⋮	⋮

3.3.2. Analysis of the traffic required to maintain a zone

The traffic that is incurred in maintaining a zone can be divided into traffic for the push-based service and traffic for updating the intrarouting information. However, these two jobs can be done at a time in the *SPIZ*, because service information is piggybacked on the routing control packet. Therefore, the rate of total traffic for maintaining zone can be expressed as follows:

$$C_{\text{intrazone[traffic/node/s]}} = C_{\text{push\_based}} + C_{\text{IARP}} = C_{\text{SAM}}, \tag{2}$$

where $C_{\text{push\_based}}$ is the rate of traffic flow for push-based *Ad/D*, C_{IAPR} is the rate of traffic flow for intrarouting, and C_{SAM} is the rate of *SAM* traffic flow.

SAM updates of route and service information can be triggered periodically (i.e., time-driven triggering) or when there is a change in the node's connectivity with a neighbor (i.e., event-driven triggering). When *SAM* updates are triggered, the node broadcast a portion of its service and routing information to all nodes within its zone. If we assume that the link cost remains constant, the rate of *SAM* traffic can be expressed as follows:

$$\begin{aligned} C_{\text{SAM[traffic/node/s]}} &= T_{\text{time\_driven}} + T_{\text{event\_driven}} \\ &= F_{\text{update}} \bullet \frac{UT_{\text{SAM}}}{N_{\text{zone}}(\rho,\sigma)} \\ &\quad + \nu \bullet \frac{UT_{\text{SAM}}}{N_{\text{zone}}(\rho,\sigma)}, \end{aligned} \tag{3}$$

where $T_{\text{time\_driven}}$ is the rate of time-driven traffic, $T_{\text{event\_driven}}$ is the rate of event-driven traffic, F_{update} is the update frequency [1/s], UT_{SAM} is the update traffic of *SAM*, ρ is the zone radius [hop], σ is the node density [neighbors/node], $N_{\text{zone}}(\rho, \sigma)$ is the number of nodes within the zone, *and* v is the node speed [neighbors/s].[3]

Note that the amount of *SAM* traffic per node does not depend on the total number of nodes, and a higher velocity or update frequency will cause heavier *SAM* traffic.

3.4. *On-demand pull-based service discovery*

When a node wants to find a service object, but has no information about the specific service object which meets the required constraints, it actively sends a query using the on-demand (pull-based) service discovery mechanism.

3.4.1. *Efficient bordercasting*

SPIZ uses the *BRP* (bordercast resolution protocol) [15] as a pull-based service discovery method. The bordercasting provided by *BRP* is much more efficient than simple flooding. It provides efficient mechanisms for sending a query to rebordercasting nodes, and for routing the query outward beyond the zone of the originating node. Additionally, it provides a query detection mechanism to prevent query overlap.

Since the zone radii in *SPIZ* are variable, *BRP* can be used more effectively. Since the zone radii are independent, the zone of one node may be completely included in the zone of another. In this case, the first node cannot explore any new zone when it receives a query from the second node, and processing such a query wastes the resources of the first node. *BRP* avoids this situation by assigning query processing to nodes which are able to explore new zones.

Figure 6 illustrates the example of bordercasting by a node with a zone radius of 3. To start bordercasting, the *BRP* constructs a bordercast tree that connects the source node to all peripheral nodes whose minimum distance to the source node is exactly equal to the zone radius. Then it chooses rebordercasting nodes on the basis of their zone radius and the query detection mechanism. In Figure 6, Nodes B and D are chosen as rebordercasting nodes, since they are closest to the source node, out of all the nodes which are able to access the outside of the zone and which have not previously received the current query. It means that the sum of Node B's (or Node D's) radius and the distance between the source and Node B (or Node D) is larger than the source node's radius. The service user unicasts a service query message to these rebordercasting nodes via forwarding nodes[4] such as Nodes A and C. Lastly, the rebordercasting node processes the responses to its queries; but if it still has no information about the target service, it performs bordercasting again in order to explore new zones. In this manner, *SPIZ* floods the network with the query, but in an efficient way.

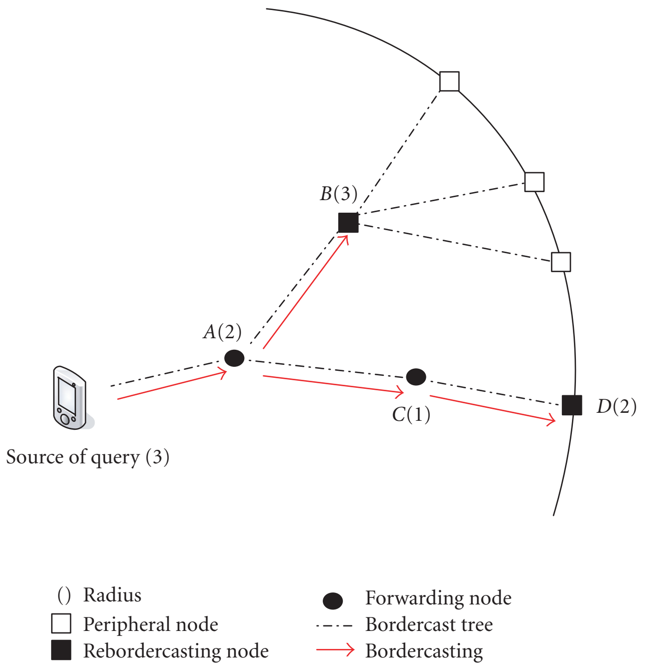


FIGURE 6: Example of bordercasting using *BRP*.

3.4.2. *Message format for pull-based Ad/D*

In order to achieve pull-based service discovery using bordercasting, we need an *SQM* (service query message) and an *SRM* (service reply message), which add service information to the general *IERP* request packet (*IERP_REQ*) and to the *IERP* reply packet (*IERP_REP*), respectively. The format of a pull-based *SQM* is shown in Figure 7. The lifetime and service provider address fields of the *SQM* are initially empty, and service provider address field is used for temporary data before an *SRM* is finally generated. The service type and additional information fields should initially be filled with data identifying the service information that the user wants to find. The destination address field of the *SQM* contains the address of a bordercasting node supplied by the source node.

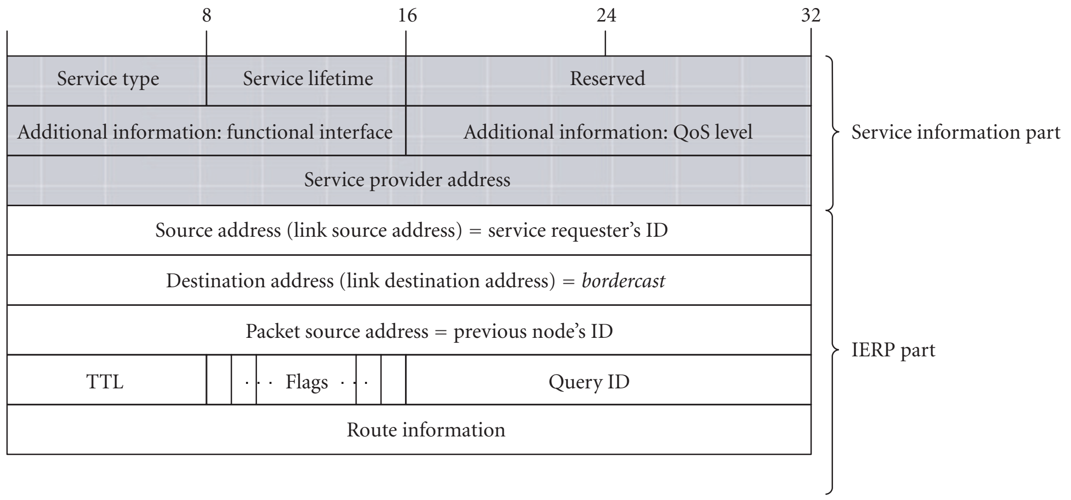
The *SRM* has similar format to the *SQM*. It contains the service information which the *SQM* has found. After an *SRM* has been created by a node which has the necessary information, it is sent to the node from which the *SQM* was received, as part of a backtracking process that leads back to the node that initiated bordercasting.

3.4.3. *Analysis and correctness proof*

The traffic generated by on-demand requests is made up of two parts: traffic for pull-based service *Ad/D* and traffic for reactive routing requests. The amount of on-demand traffic

[3] Node speed can be expressed in terms of the rate of new neighbor acquisition instead of the physical measure of distance traveled/unit time.

[4] A forwarding node is a node that lies on the path between the source node and a rebordercasting node.



8
16
24
32
Service type
Service lifetime
Reserved
Additional information: functional interface
Additional information: QoS level
Service provider address
Source address (link source address) = service requester's ID
Destination address (link destination address) = bordercast
Packet source address = previous node's ID
TTL
Flags
Query ID
Route information
Service information part
IERP part

FIGURE 7: Pull-based *SQM* (service query message) format.

per node can be expressed as

$$\begin{aligned} &C_{\text{on-demand[traffic/node/s]}} \\ &\quad = C_{\text{pullbased\_sd}} + C_{\text{reactive\_routing}} \\ &\quad = \text{SQM}_{\text{traffic[traffic/query/node]}} \bullet \frac{N_{\text{sp}}}{MTNR} \\ &\qquad + \text{IERP}_{\text{traffic[traffic/query/node]}} \bullet N_{\text{total/MSID}}, \end{aligned} \tag{4}$$

where $C_{\text{pullbased\_sd}}$ is traffic generated by pull-based *Ad/D*, $C_{\text{reactive\_routing}}$ is traffic generated by reactive routing, SQM_{traffic} and $IERP_{\text{traffic}}$ are the traffic of each node per service *Ad/D* query and routing query, respectively, N_{sp} is the number of service providers, N_{total} is the total number of nodes, *MTNR* is the mean time between service requests, and *MSID* is the mean session interarrival delay, which is the inverse of the mean call rate.

Our focus is on reducing the value of $C_{\text{pushbased\_sd}}$. In a traditional service *Ad/D* protocol, implemented as an independent layer which is separated from the routing layer, SQM_{traffic} incurs much more traffic than *SPIZ* because additional activity is needed to find routing information for the service provider. In addition, a lower value of SQM_{traffic} can be expected with *SPIZ* because it employs efficient query processing and bordercasting.

The above traffic calculation does not take into account problems in routing the *SQM* due to link failure. To include link failure in the analysis, the *NLFF* (normalized link failure frequency of nodes) must be considered, but we will omit this additional complication. Note that the amount of on-demand traffic per node depends on the total number of nodes and it will become heavier if there are increases in the number of service providers or the popularity of a service provider.

In addition, the following lemmas show that the query distribution for service discovery provides full coverage within finite time. To simplify the proof, we will assume that the network topology remains static during operation of the service discovery mechanism. The definitions used in this proof are explained in Table 2.

TABLE 2: Definitions used in correctness proof.

Symbol	Symbol description
$Y_{(t)}$	The set of reachable nodes that belong to the zones of all bordercast recipients, at time *t*.
$Y^c_{(t)}$	The complementary set of $Y_{(t)}$, which represents the set of unexplored, at time *t*.
$Z_{(t)}$	A subset of $Y_{(t)}$ such that each node in $Z_{(t)}$ has a new unexplored region; thus it is expected to invoke bordercasting, in time T for $t < T < \infty$.

Lemma 1. *If there exists a node* $n \in Z_{(t1)}$ *such that* $n \notin Z_{(t2)}$, *then* $|Y^c_{(t1)}| > |Y^c_{(t2)}|$ *for* $t1 \leq t2$.

Proof. $n \in Z_{(t1)}$ and $n \notin Z_{(t2)}$ mean that node n has invoked bordercasting and explored a new region at time $t2$. Therefore, $|Y_{(t1)}| < |Y_{(t2)}|$. From basic set theory, it also follows that $|Y^c_{(t1)}| > |Y^c_{(t2)}|$. □

Lemma 2. *SPIZ permits at least one node in* $Z_{(t1)}$ *to receive a query for service discovery by time* $t1 < t2 < \infty$.

Proof. For each node $n \in Z_{(t1)}$, there must be at least one node, m, which will launch a bordercast to node n, because $n \in Z_{(t1)}$ also implies $n \in Y_{(t1)}$. The bordercasting algorithm explained in Section 3.4.1 is such that node n will receive the service query packet from node m by time $t1 < t2 < \infty$. □

Lemma 3. *SPIZ provides full coverage.*

Proof. Based on Lemma 2, at least one node $n \in Z_{(t1)}$ can receive a service query and rebordercast it within finite time $t1 < t2 < \infty$. By exploring the zone of node n, we achieve the condition $n \notin Z_{(t2)}$. Therefore, following Lemma 1, $|Y^c_{(t1)}| > |Y^c_{(t2)}|$, thus $|Y^c|$ decreases gradually, ultimately reaching zero. □

3.5. Query processing

The simplified query processing algorithm is shown in Algorithm 2. When a rebordercasting node receives a service query, it executes the query processing algorithm. If a node has information about the service provider which matches the *SQM*, and the corresponding routing information, then that node creates an *SRM* which contains this service and routing information, and sends it back by the reverse path. But if the node has only service information about the provider, and no routing information, it only fills the service information fields of the *SQM* (including the service provider address field) before rebordercasting it. If a node has no service information that matches the *SQM*, it simply rebordercasts the *SQM*.

Now, suppose that a node receives an *SQM* with the service provider address field already filled in. We can infer from this kind of *SQM* that service information about a provider has already been located, but the routing information is still missing. If the node has the required routing information, it can create an appropriate *SRM* and send it back. However, if it has no routing information about that service provider, but it does have service and routing information about an alternative service provider, then the node creates an *SRM* with information about that alternative provider and sends it back.

Figure 8 provides a simple example of service discovery using our algorithm. When a service user wants to find the tablet *PC* service, the service user creates an *SQM* and uses the bordercasting mechanism to explore outside its own zone. The *SQM* is routed to rebordercasting nodes, one of which is Node A. When Node A receives the *SQM*, it executes the query-processing algorithm explained in Algorithm 2. As Node A does not have any information about the tablet *PC* service, it invokes the bordercasting mechanism again, thus re-routing the query to bordercasting nodes, including Node D. Then Node D executes query processing. As Node D has service and routing information for the tablet *PC* node, it creates an *SRM* and sends it back to the service user along the path taken by the *SQM*, in the reverse direction.

The service user can use the laptop service in a similar way. It creates an *SQM* and bordercasts it. When a rebordercasting node, such as Node B, receives the *SQM*, it executes the query-processing algorithm. Node B is included in the zone of laptop node, but the laptop node is not included in the zone of Node B. That means that Node B has service information, but not routing information, for the laptop node. Therefore, Node B can only fill the service information fields of the *SQM* with cached information about the laptop node. After that, it bordercasts the query again. Node C now receives the *SQM* sent by Node B, and performs query processing. Node C has the routing information for the laptop node whose address has been written in the service provider address field of the *SQM*, so it creates an *SRM* and sends it back.

```
Query-processing (SQM)
1   if Check_SPA (SQM) == NULL then //SPA
    (Service Provider Address)
2     if Have_Service_Info (ST(SQM)) then //
      ST (Service Type)
3       if Have_Route_Info (SPA(SQM)) then
4         Make_SRM (Service_Info, Route_Info)
5         Reply (SRM)
6       else
7         Fill_SQM (Service_Info) //SPA is filled
8         Bordercasting (SQM)
9       end if
10    else
11      Bordercasting (SQM)
12    end if
13  else
14    if Have_Route_Info (SPA(SQM)) then
15      Make_SRM (Service_Info, Route_Info)
16      Reply (SRM)
17    else
18      if Have_Alter_Service_Info_With_Routing_Info
        (ST(SQM)) then
19        Make_SRM (Service_Info, Route_Info)
20        Reply (SRM)
21      else
22        Bordercasting (SQM)
23      end if
24    end if
25  end if

Bordercasting (SQM)
1   tree T
2   node[] Rebordercasting_Nodes
3   T = Construct_Bordercasting_Tree
     (Root_Node, Peripheral_Nodes)
4   Rebordercasting_Nodes
       = Choose_Rebordercasting_Nodes (T)
5   for (∀node ∈ Rebordercasting_Nodes)
6         Fill_SQM (Destination_Address)
7         Send (Rebordercasting_Nodes, SQM)
8   end for
```

ALGORITHM 2: Simplified pseudocode for the query processing and bordercasting algorithm.

4. PERFORMANCE EVALUATION

We designed a simulation to evaluate the performance of *SPIZ*, which we implemented using an extended version of *NS2* from Cornell University.[5] On top of the *IEEE* 802.11 *MAC* protocol, *OLSR* [16] was used as the proactive routing

[5] http://wnl.ece.cornell.edu/Software/zrp/ns2.

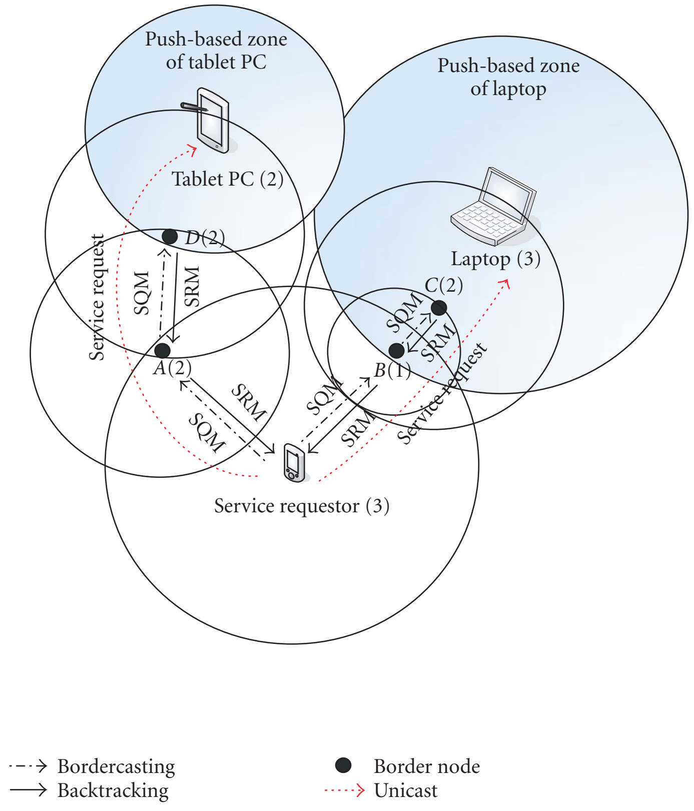


Figure 8: An example of an on-demand service discovery algorithm.

protocol integrated with a push-based service *Ad/D* protocol, and *AODV* [17] was used as the reactive routing protocol integrated with a pull-based service *Ad/D* protocol.

4.1. Simulation model

We created networks containing different numbers of nodes (50, 100, 150, and 200), spread randomly over an area of 1000 m × 1000 m. Five nodes are service providers. All nodes in the network have advance knowledge of the service types. Each simulation ran for 500 seconds and there were 30 runs in total.

There are several parameters that we can use to characterize a network. The first is the mean speed of the nodes. The faster their relative speed, the more dynamic the network is. The second parameter is the mean pause time, which controls how long a node can remain in one place before moving. The longer the pause time is, the more stable the network is. The third parameter is the *MSID* (mean session interarrival delay) which corresponds to the call rate of the nodes. The smaller the *MSID*, the more frequent calls are. From the service provider's point of view, there is one further parameter, which is the *MTNR* (mean time to next request); it represents the popularity of the service.

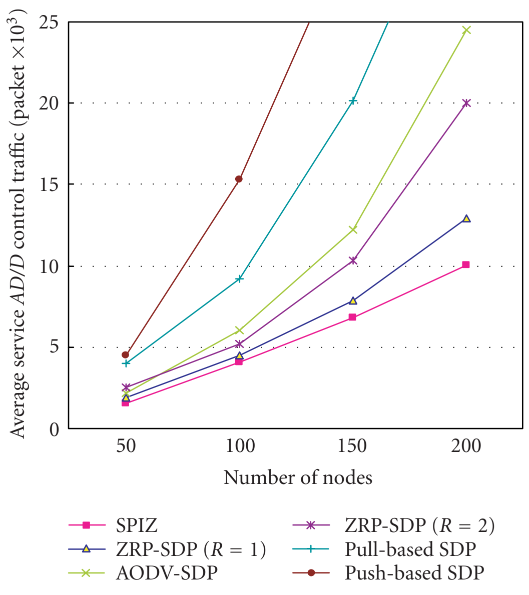
In order to simulate the service *Ad/D* traffic, a randomly chosen node sends a service query message to one service provider. The interarrival times between queries to each provider are exponentially distributed with a given *MTNR* (1 s, 10 s). Since *SPIZ* is integrated with the routing protocol, we need to simulate routing traffic as well as the service *Ad/D* traffic. We therefore make each node send a certain number of data packets to a randomly chosen destination. The number of data packets per session follows a Poisson distribution with an average of 10 packets. The interarrival time between sessions at each node is exponentially distributed with a given *MSID* (3 s, 150 s). Each source of a particular session generates 1 Kbit data packets at a constant rate of 16 packets per second.

4.2. Simulation results

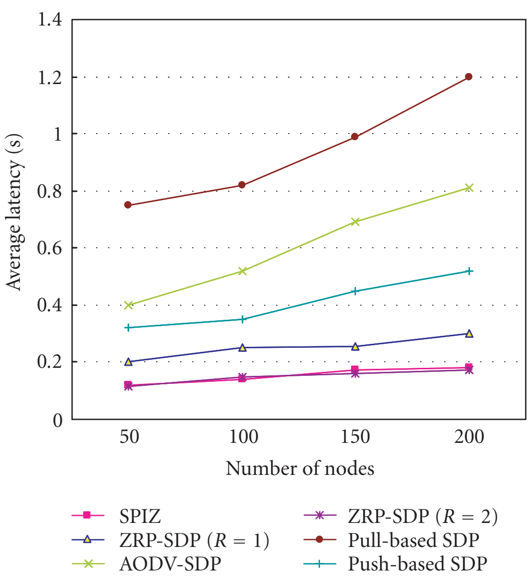
To evaluate the performance of *SPIZ*, we implemented five different service *Ad/D* strategies and simulated each strategy. *ZRP-SDP* is the service *Ad/D* protocol integrated with *ZRP*, and *AODV-SDP* is the *Ad/D* protocol integrated with *AODV*. We will also refer to pull-based *SDP* and push-based *SDP*, which are the service *Ad/D* protocols separated from the routing protocol.

4.2.1. Result of service traffic model

Figure 9 shows comparative results for average traffic and latency for different service *Ad/D* strategies. In this experiment, we simulated only the service *Ad/D* traffic and not the



(a) Average service *Ad/D* control traffic



(b) Average *Ad/D* query latency

Figure 9: Performance of *SPIZ* with only service *Ad/D* traffic.

routing traffic. The mean speed of the nodes is 10 m/s, the pause time is 10 s, and the *MTNR* of each service node is 1 s. The value of δ, ε for delayed triggering is 10 and 1.5, respectively. As Figure 9(a) indicates, *SPIZ* saves between 20% and 65% of the control traffic related to service *Ad/D* when the number of nodes is 50. The average control traffic plotted in this figure is the number of control packets passing through each node during the simulation. Therefore, we see that the total number of control packets in the network can be reduced substantially by using *SPIZ*. Moreover, the larger the number of nodes, the more definite the difference in traffic overhead between *SPIZ* and the other strategies. Among those other strategies, *ZRP-SDP* shows the best performance when the zone radius is 1 hop, but this pre-defined uniform radius may not be suited to other environments. The traffic overhead of *SPIZ* does not increase exponentially as the number of nodes increases, which shows that *SPIZ* is suitable for large-scale ad hoc networks. We can also infer that the nodes have found an approximately optimal radius from the fact that, using *SPIZ*, the traffic is less than it is for *ZRP-SDP*, whether the latter has a zone radius of 1 or 2. The average zone radius of all nodes was about 1.25 and the average zone radius of the service provider was 2.21. Figure 9(b) shows that *SPIZ* also has the best performance in term of latency.

4.2.2. *Result of realistic traffic environment*

To observe the performance of *SPIZ* in a more realistic environment, in which service *Ad/D* and routing traffic coexist, we simulated service *Ad/D* traffic and routing traffic simultaneously. We also changed the network environment 250 seconds after the start of the simulation in order to assess the adaptability of *SPIZ*. The network characteristics and traffic model that we simulated are set out in Table 3.

Table 3: Realistic traffic model.

		Period 1 (0 s ∼ 250 s)	Period 2 (250 s ∼ 500 s)
Network Environment	Average speed	15 m/s	0.5 m/s
	Pause time	10 s	100 s
Service *Ad/D* traffic	MTNR	10 s	1 s
Routing traffic	MSID	150 s	3 s

Figure 10 shows the effect on the average control traffic of varying the service *Ad/D* strategy and the number of nodes. Again, *SPIZ* gives the best performance among the six strategies, and the differential in performance grows with the number of nodes. Moreover, *SPIZ* has much more effect on the traffic overhead in this realistic scenario than in a static environment in which there is only service *Ad/D* traffic.

4.2.3. *Study of adaptability in more detail*

In order to study the performance of *SPIZ* in more detail, we analyzed the traffic for each strategy at each period, in a 50-node network. Figure 11(a) shows the results. In Period 1, *SPIZ*, *ZRP-SDP* with a zone radius of 1 and *AODV-SDP* produce relatively little traffic, while *ZRP-SDP* with a zone radius of 2 and push-based *SDP* generate much more. We suggest that this occurs because a high level of traffic is incurred by zone maintenance when there is a rapidly changing network topology and a high probability of link failure. In Period 2, however, *SPIZ* and *ZRP-SDP* with a zone radius of 2 show

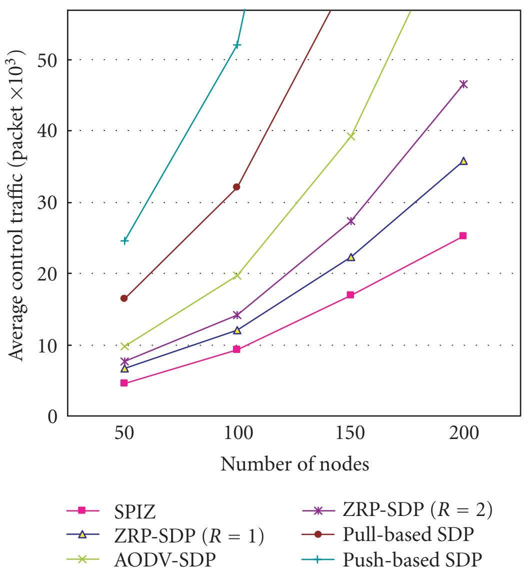
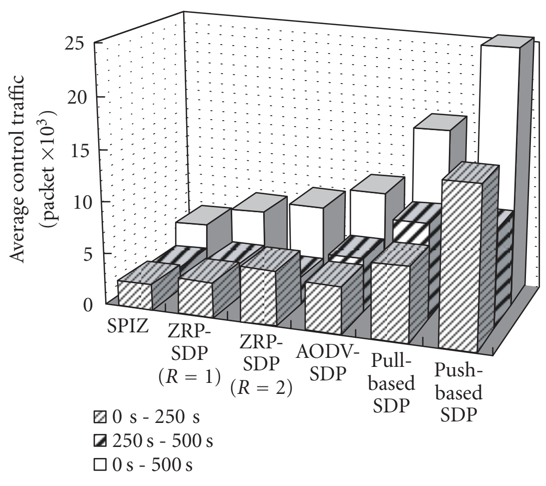
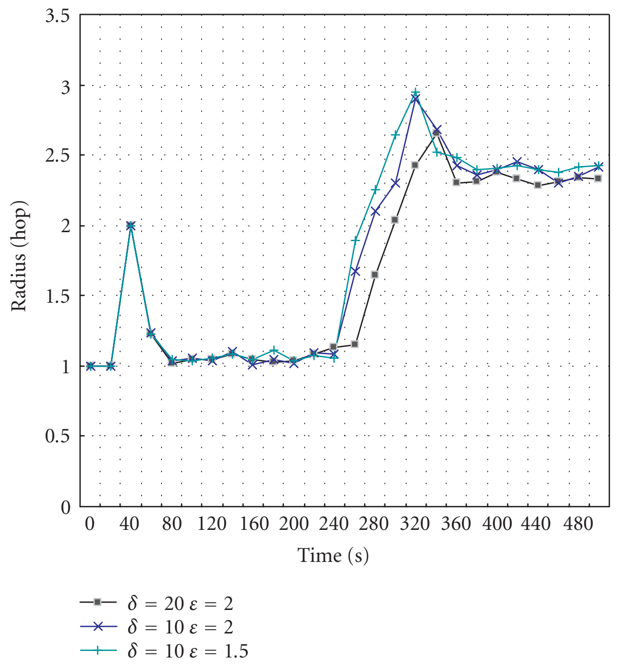


Figure 10: Average control traffic with both service *Ad/D* and routing traffic.



(a) Average control traffic



(b) Average zone radius

Figure 11: Adaptability of *SPIZ* with both resource *Ad/D* and routing traffic (50-node network).

relatively little traffic, while *AODV-SDP* and pull-based *SDP* are now much busier. This result indicates that it is more efficient to maintain larger zones when there is a relatively stable network environment and a high popularity, which are the characteristics of Period 2. As we can see from Figure 11(a), *SPIZ* has better performance than all the other schemes, in terms of the total number of control packets, over both periods.

We also plotted the average zone radius of the nodes over time while varying the δ, ε. As we can see from Figure 11(b), the average zone radius of a node is now about 1 in Period 1, growing to 2.4 in Period 2. And the average zone radius of the five service providers is 1.9 in Period 1, growing to 3.3 in Period 2. In addition, the results suggest that a high value of δ, ε means that adaptation to changing network characteristics is slow.

From these results, we can argue that the zone radius determination algorithm used by *SPIZ* can track a changing network environment while maintaining approximately optimal zone radii. We are confident of the validity of this assertion because of the strong performance of *ZRP-SDP* with a zone radius of 1 during Period 1, and with a zone radius of 2 during Period 2, as shown in Figure 11(a).

5. CONCLUSIONS

The characteristics of *MANET*s, such as a potentially highly dynamic topology and the inclusion of heterogeneous wireless nodes, which need to save energy to enhance their autonomy, require special care in the handling of distributed resource provisioning. In particular, the discovery of services must encompass the whole *MANET*, to ensure availability, while limiting resource consumption. But existing directoryless discovery protocols designed for *MANET*s are short of adaptability, efficiency, and thus also of scalability.

SPIZ is a new lightweight adaptive service *Ad/D* strategy integrated with the network layer protocol. It avoids the use of redundant flooding mechanisms and reduces the system overhead by piggybacking the *Ad/D* information on the routing packet, and can perform more effective service *Ad/D* by applying a hybrid service *Ad/D* model which combines the pull-based and push-based approaches. In addition, each node can have its own zone radius to facilitate local autonomic adaptation to dynamic changes in the network environment. Maintaining an optimal zone around each node allows *SPIZ* to provide an efficient query routing and processing mechanism. The reduced overheads of *SPIZ* can translate to lower power consumption, less congestion, and reduced memory and processing requirements. Due to these advantages, *SPIZ* has the scalability necessary for large-scale *MANET*s containing several hundreds of nodes, unlike previous directoryless service *Ad/D* strategies.

ACKNOWLEDGMENTS

This work is a part of a research project supported by Korea Ministry of Construction and Transportation (MOCT) through Korea Construction Engineering Development Collaboratory Program Management Center (KOCED PMC) at Seoul National University. The authors wish to express their gratitude for the financial support.

REFERENCES

[1] U. C. Kozat and L. Tassiulas, "Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues," *Ad Hoc Networks*, vol. 2, no. 1, pp. 23–44, 2004.

[2] F. Sailhan and V. Issarny, "Scalable service discovery for MANET," in *Proceedings of 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom '05)*, pp. 235–244, Kauai Island, Hawaii, USA, March 2005.

[3] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark—a service discovery and delivery protocol for ad-hoc networks," in *Proceedings of IEEE Wireless Communications and Networking (WCNC '03)*, vol. 3, pp. 2107–2113, New Orleans, La, USA, March 2003.

[4] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "DEAPspace—transient ad hoc networking of pervasive devices," *Computer Networks*, vol. 35, no. 4, pp. 411–428, 2001.

[5] U. C. Kozat and L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, pp. 1965–1975, San Francisco, Calif, USA, March-April 2003.

[6] W. Ma, B. Wu, W. Zhang, and L. Cheng, "Implementation of a light weight service advertisement and discovery protocol for mobile ad hoc networks," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 2, pp. 1023–1027, San Francisco, Calif, USA, December 2003.

[7] L. Cheng, "Service advertisement and discovery in mobile ad hoc networks," in *Proceedings of on the ACM Conference on Computer Supported Cooperative Work (CSCW '02)*, New Orleans, La, USA, November 2002.

[8] R. Koodli and C. E. Perkins, "Service discovery in on-demand ad hoc networks," Internet-Draft, IETF MANET Working Group, draft-koodli-manetservicediscovery-00.txt, September 2002.

[9] C.-S. Oh, Y.-B. Ko, and Y.-S. Roh, "An integrated approach for efficient routing and service discovery in mobile ad hoc networks," in *Proceedings of 2nd IEEE Consumer Communications and Networking Conference (CCNC '05)*, pp. 184–189, Las Vegas, Nev, USA, January 2005.

[10] R. Harbird, S. Halies, and C. Mascolo, "Adaptive resource discovery for ubiquitous computing," in *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing*, pp. 155–160, Toronto, Canada, October 2004.

[11] D. Noh and H. Shin, "An adaptive and scalable resource advertisement and discovery strategy for mobile ad hoc networks," in *Proceedings of 1st International Conference on Grid and Pervasive Computing (GPC '06)*, pp. 237–249, Taichung, Taiwan, May 2006.

[12] P. Samar, M. R. Pearlman, and Z. J. Haas, "Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 595–608, 2004.

[13] Z. J. Haas, M. R. Pearlman, and P. Samar, "The zone routing protocol (ZRP) for ad hoc networks," Internet-Draft, IETF MANET Working Group, July 2002.

[14] M. R. Pearlman and Z. J. Haas, "Determining the optimal configuration for the zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1395–1414, 1999.

[15] Z. J. Haas, M. R. Pearlman, and P. Samar, "The bordercast routing protocol (BRP) for ad hoc networks," Internet-Draft, IETF MANET Working Group, July 2002.

[16] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, IETF MANET Working Group, October 2003.

[17] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, IETF MANET Working Group, July 2003.