Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks

by

Honghui Luo

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfilment of the requirements for the degree of

Master of Science

Information and Systems Science

School of Computer Science

Carleton University

Ottawa, Ontario

October 2003

The undersigned hereby recommend to

The Faculty of Graduate Studies and Research

Acceptance of the thesis,

Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks

submitted by

Honghui Luo

in partial fulfilment of the requirements for the degree of

Master of Science

Information and Systems Science

Dr. Douglas Howe

(Director, School of Computer Science)

Dr. Michel Barbeau

(Thesis Supervisor)

Carleton University

October 2003

Abstract

Service discovery is an important and necessary component of ad hoc networks. To fit within the context of ad hoc networks, a *Post-query model* with several service discovery strategies named *Post-query strategies* have been proposed, by Barbeau and Kranakis, which focus on locating services over an ad hoc network. Each strategy consists of a sequence of *Post-query protocols* executed in rounds. They proposed four types of Post-query strategies: the *greedy*, *incremental*, *uniform memoryless* and *with memory* strategies. Inspired by these proposals, we define the *conservative* Post-query strategy. We give the detailed design and implementation of these five strategies. We also conduct a simulation of these five strategies in combination with two types of ad hoc routing protocols: *Dynamic Source Routing (DSR)* (a source initiated on-demand routing protocol) and *Destination-Sequenced Distance Vector (DSDV)* (a table-driven routing protocol). Through simulation, the performance of the five Post-query strategies when combined with the DSR protocol and DSDV protocol are evaluated, in terms of success rate, number of transmitted messages and average waiting time.

Our main conclusions are as follows. The greedy strategy achieves a very high success rate with the lowest average waiting time and highest number of transmitted messages of all the strategies. The conservative strategy can only achieve a high success rate in a high dynamic ratio ad hoc network. The incremental strategy can also achieve a high success rate with a low number of transmitted messages, while it has the longest average waiting time of all the strategies. The uniform memoryless strategy uses a relatively low amount of network bandwidth if the sizes of the posting and querying sets are relatively low. The *with memory* strategy can achieve a high success rate, while each node builds a cache of previously visited nodes. Compared with the uniform memoryless strategy, the success rate of this strategy is improved by 10.34% when combined with the DSR protocol, and by 16.5% when combined with the DSDV protocol.

Acknowledgements

It took me a long trip to find my own passion. I sincerely thank my thesis supervisor Dr. Michel Barbeau, especially for his wisdom, guidance, advice and support, which gave me more prompting whenever I needed it, his great patience which never gave up encouraging me has helped me throughout the whole thesis.

During the process of my thesis, I also received warm-hearted assistance and support from many other friends. I wish to express my deep gratitude to our project group members, especially Huan Qi, Maoning Wang , and Zheyin Li for their generous help, invaluable discussions and kindly encouragement. I also wish to acknowledge Tao Wan and Yufang Zhu for their in-depth and constructive comments on my thesis. I also received a lot of help on the LATEX tutorial from Miguel Vargas Martín and Hua Guo.

Finally, I leave my special thanks to my family: my husband Qing, my parents Yuzhen Zhang and Jixian Luo, my sisters Hongmei Luo and Feng Zhang for their endless love. Their great courage to overcome any kind of difficulty inspired me to accomplish this work.

Contents

List of Tables

List of Figures

List of Acronyms

AODV	Ad hoc Distance Vector Routing
AWT	Average Waiting Time
DA	Directory Agent
DCF	Distributed Coordination Function
DR	Dynamic Ratio
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
LC	Link Change
MAC	Medium Access Control
NS	Network Simulator
NTM	Number of Transmitted Messages
ODMRP	On-Demand Multicast Routing Protocol
OLSR	Optimized Link State Routing
RF	Radio Frequency
SA	Service Agent
SDP	Service Discovery Protocol
SLP	Service Location Protocol
SR	Success Rate
SREP	Service Reply
SREQ	Service Request
TORA	Temporally-Ordered Routing Algorithm
UA	User Agent
ZRP	Zone Routing Protocol

Chapter 1

Introduction

This chapter starts by introducing the context in which this thesis has been written, followed by a history of the research topic. Then, the contributions are summarized and finally, an outline of the thesis is presented.

1.1 Research context

1.1.1 Service Discovery in ad hoc networks

An *ad hoc network* is a multihop wireless network consisting of a set of mobile nodes not requiring any preexisting network infrastructure. It is used in situations where temporary network connectivity is needed. Some examples of the possible uses of ad hoc networks include emergency disaster relief and students using laptop computers to exchange course information. Within this kind of

network, every node is free to move randomly and get connected arbitrarily to other nodes. Every node plays the role of router as well as host. Thus, such a network's wireless topology may change rapidly and unpredictably. In contrast to traditional wired networks, ad hoc networks have lower bandwidth and are more subject to the influences of interference, fading and noise. Highly dynamic changes of topology imply an increased control message overhead to maintain the connectivity. Bandwidth should be used efficiently, and battery power should be conserved. All the aforementioned constraints bring new challenges not only to network routing, but also to application layer protocols such as the service discovery protocol. Over the past few years, much effort has been put into inventing effective ad hoc routing protocols under these constraints. These routing protocols can generally be categorized as on-demand routing protocols or table-driven routing protocols. They offer good support for application layer protocols.

Service discovery is defined as the problem of automatically locating different services within a network. Services are entities available for use by a network node. For example, a service could be a printer, an image scanner, or a file server. Currently there is a variety of service discovery protocols emerging in the network community. The most well known so far are the *Service Location Protocol (SLP)* defined by IETF [Gut99]; *Jini*, defined by Sun Microsystems [Sun99]; and the *Service Discovery Protocol (SDP)*, defined by Bluetooth Forum [Blu01].

SLP is designed for TCP/IP networks and is scalable up to large enterprise

networks. The SLP architecture contains three main components: *User Agent (UA)*, which locates services on behalf of a client; *Service Agent (SA)*, which provides information about a service; and *Directory Agent (DA)*, which is optional. A DA caches service information received from SAs and replies to service requests from UAs. For small networks, it is more efficient to deploy SLP without a DA. SLP can operate in a distributed manner (without a DA) or in a centralized manner (with one or more DAs).

The Jini technology is written in the programming language Java. The architecture of Jini consists of three main components: *service provider*, which is the front-end of a service available on a network; *client*, which is a user of services; and *lookup service*, which is a central node where service providers register their services (similar to the DA of SLP).

Bluetooth is a short range ad hoc network technology. Bluetooth networks are also named piconets or scatternets. A piconet has one device called the master which can support up to seven slave Bluetooth devices. A set of interconnected piconets forms a scatternet. In a piconet, all of the devices use the same radio channel. Bluetooth employs SDP for locating services provided by or available through other Bluetooth enabled devices. SDP defines the SDP server and SDP client. To find a service, an SDP client unicasts an SDP request to SDP servers one by one. SDP servers either send back SDP responses or error responses. In contrast to SLP and Jini, SDP is designed specifically for the Bluetooth networks,

where the set of services that are available change dynamically based on the Radio Frequency (RF) proximity of devices in motion.

With the rising number of Internet services and increased use of wireless devices, service discovery will be a very important component in self-organizing ad hoc networks. Nevertheless, it is very challenging to design and implement service discovery protocols in such networks because of their characteristics. SLP and Jini do not specifically target ad hoc networks.

Efficient service discovery on ad hoc networks requires a balance between performance (success rate) and cost (number of transmitted messages and average waiting time). The following question has been asked: How fast and cheaply can a service be located under a dynamic changing topology which results in unreliable network communication? An attempt to answer this question has been proposed by Barbeau and Kranakis [Bar03]. They introduced a *Post-query model* focused on locating services in an ad hoc network.

1.1.2 History of the Post-query model

The Post-query model of Barbeau and Kranakis is based on the *distributed match-making*, put forward by Mullender and Vitányi [Mul88] and Kranakis and Vitányi [Kra92]. This distributed match-making paradigm refers only to traditional networks. In network computing, the client/server model is used. The client sends a request to the server about a service and the server replies to the client about

that service. Each node in the network is either a client or a server. Most distributed systems have been designed to use this client/server model. Client and server processes residing in different nodes may need to find each other, without knowing the each other's host address in advance. For each server s in the network, there is a set of nodes $P(s)$ to which s posts its service. For a client c in the network which looks for a particular service provided by a server s, there is a set of nodes $Q(c)$. Let $1, 2, \ldots, k$ be the k different types of services the network has and let $K = \{1, 2, \ldots, k\}$. A *Post-query protocol* is a pair of functions $(P(s),\ Q(c))$, where $P(s)$ is the *posting protocol*, and $Q(c)$ is the *query protocol*.

Barbeau and Kranakis [Bar03] adapt and extend the distributed match making model to the context of ad hoc networks. They define algorithms for posting and querying services named Post-query protocols, and Post-query strategies which are time dependent Post-query protocols, executed in rounds modeling post-query interactions between clients and servers in an ad hoc network. They assume that routes to services are discovered before or at the same time as services. In an ad hoc network, at each round r, the servers first post services they can offer to other nodes according to a *posting protocol* P_r, and the clients then query other nodes in the network according to a *querying protocol* Q_r. A Post-query strategy consists of a sequence $(P_1, Q_1), \ldots, (P_r, Q_r), \ldots, (P_R, Q_R)$ of Post-query protocols that adapt to the topological changes over time in an ad hoc network. In each round, posting is executed first and then querying is conducted.

The Post-query strategies include:

1. The greedy post-query strategy: This is the post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter. All the servers post to all the nodes, and all the clients query all the nodes of the network using a network broadcast mechanism based on a flooding algorithm. It is called greedy because it consumes significant network resources.

2. Based on the greedy strategy of Barbeau and Kranakis, we define another strategy called the conservative Post-query strategy. This is a post-broadcast-h, query-broadcast-h strategy with h equal to one. It uses a network one-hop broadcast mechanism. Compared with the greedy strategy, this strategy is conservative by eliminating many of the transmitted messages on the network.

3. The incremental post-query strategy: This is the post-incremental, query-incremental strategy. Servers and clients post to and query from a small set of nodes in the first round and increase their set size gradually. This strategy can save network resources.

4. The uniform memoryless post-query strategy: This is a post-to-l, query-l' strategy consisting of uniform and memoryless rounds. Servers post all the services they can offer to a random set of size l and clients query a random set of size l'.

5. The *with memory* post-query strategy: This is a post-to-l, query-l' strategy combined with a memory usage strategy. The memory is used to store the addresses of the nodes posted to or queried from before. Each new round only involves posting (querying) previously unused nodes. Each node in the ad hoc network builds a cache to store the addresses of the nodes it has already contacted.

With the proposition of these five Post-query strategies, a uniform model needs to be constructed to evaluate their performance. As application layer protocols, Post-query protocols cannot work well without the support of the network layer routing protocols. How those strategies perform when they are combined with some typical ad hoc routing protocols is an interesting open question.

1.2 Contribution

This thesis focuses on Post-query strategies and their design, implementation and evaluation when combined with the DSR protocol and DSDV protocol. The contributions of this thesis are summarized as follows:

- The proposal of the conservative Post-query strategy;
- The design and implementation of five types of Post-query strategies, including the greedy, conservative, incremental, uniform memoryless and with memory strategies, combined with the DSR protocol and DSDV protocol;

- The design of a simulation model that covers constructing network topologies, selecting the parameters used in the simulation and designing simulation scenarios for different situations;

- A performance evaluation of these five strategies combined with the DSR protocol and DSDV protocol.

1.3 Outline

The remainder of this thesis is structured in the following manner:

- Chapter 2 reviews the Post-query protocols in detail, and introduces recent related work on combining service discovery with ad hoc routing protocols.

- Chapter 3 describes in detail the design of the Post-query strategies, including architecture design, major activities and the adopted algorithms.

- Chapter 4 gives an overview of the simulation environment and describes the simulation scenarios. It also covers how the parameters which will affect performance are chosen. In addition, it shows the implementation of these five Post-query strategies in the simulation tool.

- Chapter 5 presents performance metrics and simulation results, followed by a quantitative performance evaluation of the five Post-query strategies combined with the DSR protocol and DSDV protocol. Then it summarizes

each strategy's performance. In addition, it explains the design choices that account for these performances.

- Chapter 6 puts forward conclusions on the thesis. Furthermore, some suggestions for the direction of the future work are advanced.

Chapter 2

Review of Literature

In this chapter, we briefly introduce two variations of ad hoc routing protocols, focusing on the DSDV protocol and DSR protocol. The original Post-query model is then described in detail. Subsequently, a new approach, combining service discovery with suitable ad hoc routing protocols is presented. Finally, we study some related work.

2.1 Ad hoc routing protocols

Numerous routing protocols have been developed for ad hoc networks over the last few decades. Generally, these protocols can be categorized as either *table-driven* or *source initiated on-demand*. Table-driven protocols attempt to maintain consistent, up-to-date routing information from each node to all the other nodes in the network. Each node of the network requires the maintenance of one or more

tables to store routing information. All the nodes propagate routing updates throughout the network periodically in order to keep a consistent network view respondent to changes in the network topology. In source initiated on-demand routing, all up-to-date routes are not maintained at every node; instead, the routes are created when required. A node invokes a route discovery process within the network when it requires a route to a destination node. Once a route has been established, it is maintained through a route maintenance process. The route remains valid until the destination is unreachable or until the route is no longer needed. In the following section, we introduce one well-known table-driven protocol and one source initiated on-demand protocol.

2.1.1 Destination Sequenced Distance Vector Routing Protocol

Destination Sequenced Distance Vector Routing (DSDV) [Per94] is derived from the Distributed Bellman-Ford algorithm [Jub87]. In the DSDV protocol, each node maintains a routing table that stores the next-hop and number of hops for all reachable destinations. To avoid routing loops, each route table entry is tagged with a sequence number. The sequence number enables the nodes to distinguish stale routes from new ones. To maintain consistency in the routing tables, each node periodically broadcasts routing updates. When a node receives a new route update packet, it compares the packet to the information already available. Route

updates are selected based on the following rules. Routes with larger sequence number are always preferred. If two update packets have identical sequence numbers, the DSDV protocol chooses the shortest path based on the hop count to the destination. There are two possible types of route update packets. The first is called a *full dump* and the second is known as an *incremental packet*. The former type of packet carries all available routing information from the sender's routing table, while the latter type of packet is used to relay information which has changed since the last full dump, reducing the generated traffic.

The DSDV protocol is dependent on periodic broadcasts; it needs some time to converge before a route can be used. In an ad hoc network, the topology is expected to be very dynamic, resulting in a slow convergence of routes as packets are dropped and nodes move around. Periodic and triggered broadcasts also add a significant overhead to the network, especially those exhibiting high mobility. When the number of nodes in the network grows, the size of the routing tables and the bandwidth required to update them also grows, which could result in excessive communication overhead. This overhead is nearly constant with respect to the mobility rate. The simulation results in [Bro98] show that, the DSDV protocol delivers virtually all data packets when the node mobility rate and speed are low, and it has a lower packet delivery ratio as node mobility increases.

2.1.2 Dynamic Source Routing Protocol

Dynamic Source Routing (DSR) [Joh96] is a source initiated on-demand routing protocol. Source routing means that the sender of the packet determines the complete ordered list of the nodes through which the packet should be forwarded. In contrast to the DSDV protocol, the DSR protocol uses no periodic routing messages, thus greatly reducing network overhead and avoiding a large number of routing updates. Instead, the DSR protocol requires that each node in the network maintains a route cache which saves the found active routes.

The main procedures in the DSR protocol are *route discovery* and *route maintenance*. When a source node wants to send a packet to a destination node in the network, it first looks up its route cache to determine if a route to the destination has already been stored. If an unexpired route to the destination exists, the source node then adopts this route to send the packet. However, if such a route is not found in its route cache, the source node then invokes a route discovery procedure. It broadcasts a *route request* packet which contains the address of the source and destination nodes and a unique identification number. Each intermediate node checks if a route to the destination exists in its route cache. If such a route is not found, the intermediate node appends its address to the route record of the packet and forwards the packet to its neighbors. A *route reply* packet is generated when either the destination node or an intermediate node which has an unexpired route to the destination node receives the route request packet. When

the destination node generates the route reply, it copies the route information from the route request packet into the route reply packet. When an intermediate node sends the route reply, it appends the found route to the route information of the request packet and puts it into the route reply packet. In order to send the route reply packet, the responding node must have a route to the source node. If symmetric links are supported, the reverse of the route information can be used to send this route reply packet. In situations where symmetric links are not supported, the destination node or the intermediate node can initiate route discovery to the source node and piggyback the route reply onto this new route request. The DSR protocol uses *route error* packets and acknowledgements for the route maintenance procedure. Each node residing along the route is responsible for detecting whether the link to its next hop is broken when it wants to transmit a packet to the next hop. The link breakage is detected using a wireless MAC layer retransmission and acknowledgement mechanism or passive acknowledgements as described in [Mal99]. When a broken link is detected, the node returns a *route error* packet to the source node of the packet. When a *route error* packet is received, the broken link and the links after it are then removed from any route cache which contains this hop. The source node can use any other route to the destination node if there is such an unexpired route in its route cache. Otherwise, it can initiate a *route discovery* procedure to find a new route. In the DSR protocol, the network overhead grows if the packet has to go through more hops

to reach the destination node. It does not use periodic routing advertisements, thereby saving bandwidth and reducing power consumption. On the other hand, as the network becomes larger, control packets and data packets also become larger, because they need to carry addresses for every node in the path. Moreover, aggressive uses of the route cache will cause delays and affect throughput performance.

2.2 Existing approaches

In Section 2.1, the representative ad hoc routing protocols are introduced. These protocols provide the routing support for the service discovery protocols. It is challenging to design and implement service discovery protocols in ad hoc networks because of their characteristics. Several attempts have been made to enable service discovery in ad hoc networks. In the following sections, we focus on reviewing two of the existing approaches.

2.2.1 The Post-query strategies

Barbeau and Kranakis [Bar03] define several Post-query strategies that can be used to locate a service in an ad hoc network, including the: greedy, incremental, uniform memoryless and with memory strategies. Service discovery protocols are abstracted as Post-query protocols. The Post-query strategies are time dependent

Post-query protocols that are executed in rounds. Their proposals are based on the assumption that routes to services are discovered before or at the same time as services. They aim at maximizing the probability that in the Post-query strategy, a given client succeeds in locating a service and minimizing the waiting time and total number of postings and queryings.

A *Post-query protocol* is defined as a pair of functions (P, Q), where $P(s)$ is the *posting protocol* and $Q(c)$ is the *querying protocol*. Let $1, 2, \ldots, k$ be the k types of services offered in the network and let $K = \{1, 2, \ldots, k\}$. Each server s posts a service of type k_s to a set of nodes N_s according to the algorithm defined in $P(s)$. Similarly, each client c queries a service type k_c from a set of nodes N_c based on the algorithm defined in $Q(c)$. A client successfully locates a service if this service has been posted by some server to a node that it queries. In the context of ad hoc networks, the topology changes as time passes, so a node may be unreachable during the execution of a Post-query protocol. Hence, a Post-query protocol is not by itself sufficient to efficiently adapt to the dynamic changes of the network topology. To solve this problem, Post-query strategies are proposed. They are executed in a sequence of rounds; at each round r a pair of Post-query protocols $(P(s), Q(c))$ are employed. The maximum round R is the upper bound of the number of rounds, after which the nodes terminate their strategies. A Post-query strategy is defined as a sequence $(P_1(s), Q_1(c)), (P_2(s), Q_2(c)), \ldots, (P_R(s), Q_R(c))$ of Post-query protocols. To model the influence of dynamic topology changes, a

connection probability p is introduced with $0 \leq p \leq 1$ to indicate the probability of a communication path existing between each pair of nodes in the network.

The greedy, incremental, uniform memoryless and with memory strategies are defined. They use a common algorithm. For each round r, a server s first posts its services according to the posting protocol $P_r(s)$, and a client then queries services based on the querying protocol $Q_r(c)$. In each round, all nodes follow the posting-querying order. The algorithm is executed until the maximum round index R is reached or the queried services are successfully located, whichever comes first. The proposals of each Post-query strategy are as follows:

1. The greedy Post-query strategy: In this strategy, all nodes post to all nodes and all nodes query all nodes in the network using the network broadcast mechanism. This strategy is *non-adaptive* because the execution in each round remains the same. There are two alternatives in the greedy strategy. One is called *post-greedy*, in which the servers post to all nodes and the clients query only one node in the network. The other is called *query-greedy*, in which the servers post to one node and the clients query all the nodes in the network.

2. The incremental Post-query strategy: In order to conserve valuable resources, servers and clients post to and query from a small set of nodes in the first round and increase the sizes of the posting and querying sets gradually. Two variants of this strategy, *post-incremental* and *query-incremental*

are defined. In the former, only the size of the posting set is increased while in the latter case, only the size of the querying set is incremented.

3. The uniform memoryless Post-query strategy: This strategy is called *uniform* because the same Post-query protocol is executed in each round, and the posting and querying protocols are identical for all the nodes of the network. Servers post all the services they can offer to a random set of nodes. Clients query a random set of nodes. The chosen sizes of the posting and querying sets affect the waiting time, network overload and the probability that a given client $c \in N$ succeeds in finding a service.

4. The *with memory* Post-query strategy: In this strategy, each client builds a cache to store the addresses of the nodes posted to or queried from in previous rounds. Each new round only involves posting (querying) previously uncontacted nodes.

The simulation results show that, the probability of success within a low number of rounds increases as the connection probability p increases. The probability of success within a low number of rounds increases with the number of nodes since the number of service offers increases as well. The greedy strategy maximizes the probability of succeeding while requiring the greatest network resources. The waiting time for the greedy strategy is the shortest of all the strategies. The incremental strategy involves the longest waiting time, while it consumes the least

network resources. The waiting time and cost of the uniform memoryless strategy lie between that of the greedy and incremental strategies. When the connection probability p is greater than 0.5, the probability of succeeding is higher than with the greedy strategy. Simulation also shows that in dense ad hoc networks, Post-query strategies are more useful.

2.2.2 Service discovery in on-demand ad hoc networks

Koodli and Perkins [Koo02] define another work in progress approach for service discovery in ad hoc networks. They add extensions to suitable ad hoc routing protocols in order to provide support for service discovery along with routes to those services. In their approach, the association between a service and the IP address of the node hosting the service is known as *service binding*.

For table-driven routing protocols, a Service Reply extension is added to a topology updating packet. By this means, service information as well as the information about which links are available for creating routes, can be made available immediately. The service discovery operation is combined with the operation of processing new topology update packets.

For source initiated on-demand routing protocols, the basic service discovery process adopts the same operations and message format as the *route discovery* process, while it adds a *service request* extension to the route request packet. There are two types of service request extensions: the *Serivce Port Request* and

Service URL Request extensions. When a node wants to locate a service, it initiates a service request by including an extension in the route request and then floods this route request. A node that receives a route request with a service request extension processes this service request according to Figure 2.1.

A node that receives a route request with a service request extension (such a message is called an SREQ) first checks whether it has a valid service binding with a valid lifetime. If it has no such service binding, it should rebroadcast the original SREQ packet. Next, it verifies whether there is a valid route to the resolved IP address of the service binding. If the node has the service binding as well as a valid route, it constructs a service reply extension to a route reply (such a message is called an SREP). Then, it sends the SREP packet back to the source node. If the node only has a valid service binding but no valid route to the resolved IP address, it sets the Destination IP address to the resolved IP address, and rebroadcasts the modified SREQ packet.

2.3 Related Work

In the previous section, we described two approaches for service discovery in ad hoc networks. Besides these two approaches, we review related work on the design and implementation of service discovery protocols in such networks.

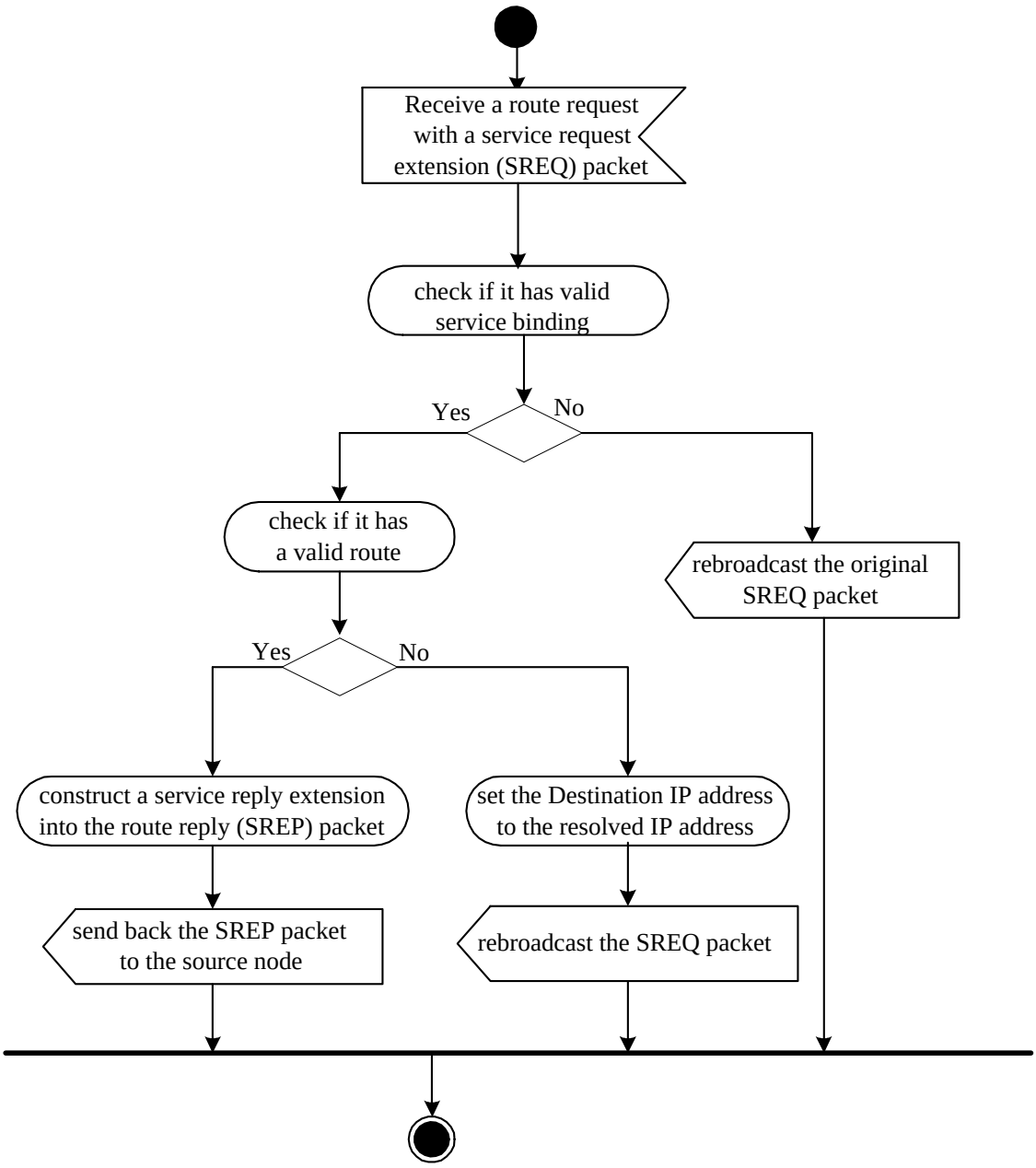

Figure 2.1: Activity diagram of receiving an SREQ packet of an intermediate node

2.3.1 Service discovery in single-hop ad hoc networks

IBM has developed DEAPspace [Nid01] [Her01] that addresses the service discovery problem in wireless ad hoc single-hop networks. Its primary goal is faster convergence of service information available in the network. Considering that in an ad hoc network, the maintenance of the centralized node which stores all the service information of the network is difficult and complicated, DEAPspace selects a distributed approach: the *push model* (in which servers send unsolicited service advertisements).

Each node in DEAPspace maintains a cache to store all the services offered in the network. Nodes participate in the advertising of services through a broadcast mechanism. Periodic broadcasts are scheduled in a proactive way using an adaptive backoff mechanism. If a node which receives the service advertisement finds that the services that it offers are absent or about to time out, it increases its chances of broadcasting more frequently. Each node adds new service advertisements into its internal cache, if in these advertisements, the services offered in the network have longer lifetime values than the ones in its internal cache.

A performance evaluation of DEAPspace shows that the bandwidth required for the DEAPspace algorithm is about the same as other push model solutions. However, the time for the discovery of available services is better with DEAPspace than with other push model solutions which also use a broadcast mechanism, since more service information is being propagated when one node broadcasts its entire

cache than when each node of the network only broadcasts the service information which it offers. However, we noticed that unnecessarily repeated broadcasts occur in DEAPspace, consuming network bandwidth. Moreover, maintaining such a tight convergence in a highly dynamic ad hoc network may not be worth the effort.

2.3.2 Konark - A service discovery delivery protocol for ad hoc networks

Konark [Hel03] is a middleware designed specifically for the discovery and delivery of services in multihop ad hoc networks. It comprises two major operations - service discovery and service delivery. We focus on reviewing the service discovery aspect. Konark supports the *push model* and *pull model*. In the former, the servers send unsolicited service advertisements, while in the latter model the clients send out service requests. The Konark project assumes the multicast support of underlying ad hoc routing protocols.

Each node adopts a Konark *SDP Manager* which is responsible for the discovery of the required services and the registration and advertisement of its local services. All the nodes join a locally-scoped multicast group. The *SDP Manager* of each node maintains a cache called a *service registry* to store the node's local services, as well as service information that it has discovered or received via service advertisements. The service registry has a tree-based data structure which

classifies the service information. This structure facilitates the service query and advertisement processes. To discover services in the network, servers employ a mechanism called *passive push* and clients use an *active pull* mechanism. First, a client initiates a service discovery message and sends out a discovery message on a fixed multicast group. After receiving such a service discovery message, each server checks the corresponding service in its service registry. If such a service is found, the server constructs a service advertisement message and replies back to the client. The service advertisement process is similar to the service discovery process. Upon receiving a service advertisement message, the clients store it according to the structure of their service registries.

The Konark project has recently paid more attention to an efficient service discovery protocol to exploit the nature of highly dynamic ad hoc networks. Based on its original design, it has developed a new algorithm which attempts to balance the convergence time and network bandwidth use. Compared with the DEAPspace algorithm we reviewed in Section 2.3.1., when a node receives a service message, it only multicasts the difference between its own relevant services and the service information in the received service message. Hence, network overload is reduced. The new algorithm also avoids the storm of concurrent multicasts by randomly assigning the multicast time interval. Moreover, if the node which receives the service message verifies that its service registry has the same service information as that in the received service message, it then remains

silent and no multicast occurs.

The performance evaluation of this project is a work in process. From the algorithm, we can conclude that this approach still uses a considerable amount of multicast messages to achieve faster convergence.

2.3.3 Service discovery protocol based on On-Demand Multicast Routing Protocol

A lightweight service advertisement and discovery protocol for ad hoc networks, based on the On-Demand Multicast Routing Protocol (ODMRP) [Lee99], is proposed by Liang Cheng [Che02]. The service advertisement and discovery information is piggybacked onto ODMRP routing packets in this protocol. It supports both the pull model and push model described in Section 2.3.2.

In the push model, each server first constructs a service advertisement message. It then combines this message with the *join query* packet header by appending the advertisement message as an extension. Next, the server multicasts the revised ODMRP *join query* packet. When a client receives the service advertisement, it first saves the service information into its cache. Then it sends back a *service awareness reply*, which is an ODMRP *join reply* packet without any extension. Once a server receives service awareness replies sent back from some clients, the server sends its updated service advertisement messages using an ODMRP packet with the server advertisement extension. In the pull model,

if a client wants to locate a service, it first sends a query to a well-known multicast address. The service query message is similar to the service advertisement message, and is also combined with an ODMRP *join query* packet. Once a node receives a service query message sent by a client, it first waits for a random period of time to determine whether it has already had a corresponding service awareness reply message. If it is found, the node keeps silent. Otherwise, it checks its cached service information. If the queried service can be obtained locally, the node constructs a service awareness reply message and sends it back to the client.

The design and implementation of this protocol benefit from the design of the ODMRP protocol. This protocol only sends update advertisements to avoid periodically sending multicast messages, thus, reducing the network overhead.

Chapter 3

Detailed Design of the Post-query Strategies

In Chapter 2, we reviewed several approaches to service discovery in ad hoc networks. We also detailed the greedy, incremental, uniform memoryless and *with memory* Post-query strategies. In this chapter, in addition to presenting our detailed design of these proposed Post-query strategies, we define another Post-query strategy called the conservative strategy, based on the greedy strategy, and describe its design in detail. This chapter first describes the design architecture of our protocol stack, and then offers a detailed design for each Post-query strategy. The flow of messages in the main processes are shown using message sequence diagrams according to the Message Sequence Chart 96 standard (see Appendix A) [Msc99].

3.1 Architecture design

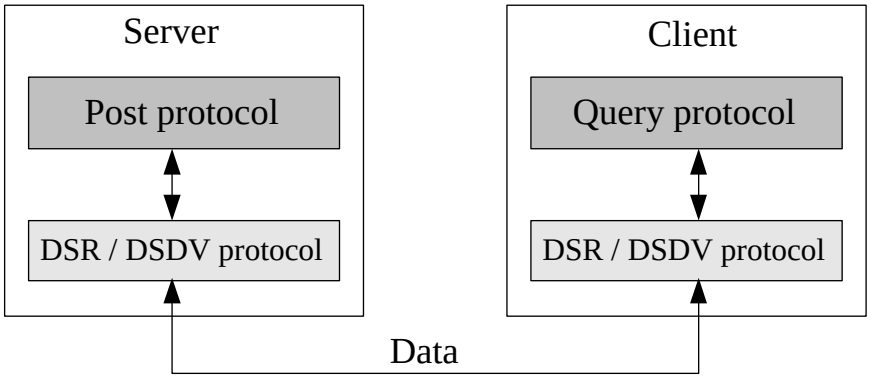

Figure 3.1: Architecture of protocol stack

The architecture design is illustrated in Figure 3.1. Each node in an ad hoc network is either a server or a client. For each server, a Post protocol is employed and for each client a Query protocol is employed. We choose the DSR protocol and DSDV protocol as the underlying ad hoc routing protocols.

3.2 The message format of the Post-query protocol

Figure 3.2 illustrates the message format of the Post-query protocol. Here we have three types of messages transmitted over the network: ServicePost, ServiceQuery and ServiceReply messages. The lifetime field indicates how long the message will be active. A list of the services in the message format covers the situation in

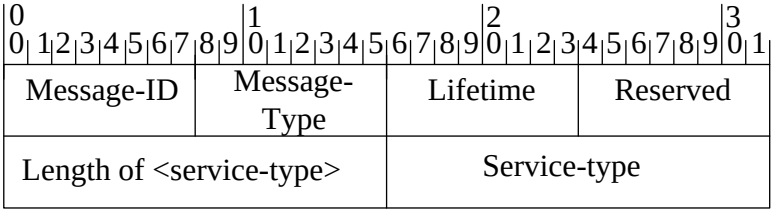

Figure 3.2: The Post-query protocol message format

which a server may offer several different kinds of services, a client may request various kinds of services and a client may receive several diverse services from one or more servers.

3.3 Common design features of these five Post-query strategies

Before we go deep into the detailed design of each strategy, we synthesize the common design features of these five strategies:

1. Our main concern is how to locate services in ad hoc networks. We focus on the design of efficiently locating services.

2. We require from each node of the network the maintenance of a cache to store the service information from previous rounds. Thus, after each round, the nodes which receive other servers' postings become potential *servers* for the following rounds. This means that, in the following rounds when these

nodes are queried for the services which they cached from previous rounds, they are capable of sending back replies.

3. Although the servers and clients have a peer-to-peer relationship, from an application layer protocol view they have different functions. A server may send ServicePost messages to other nodes, or ServiceReply messages to other clients. It can also receive ServiceQuery messages from other clients and ServicePost messages from other servers. A client may send Service-Query messages to other nodes, or ServiceReply messages to other clients. It can also receive ServicePost messages from other servers, ServiceQuery messages from other clients and ServiceReply messages from other clients.

4. When a node receives a query from a client about a service, it sends all the matched service information back to this client. Hence, if this client wants to use this service information later, it has more choices. If this node does not have the desired service, it keeps silent and does nothing.

3.4 The greedy Post-query strategy

The underlying link layer protocol treat the application data as only a one-hop broadcast if the destination IP address is set to IP_BROADCAST (-1). The role of the DSR protocol and DSDV protocol is to deliver service reply messages to the senders. The greedy strategy requires that all servers post their services

to all the nodes, and that all clients query their services from all the nodes in the network in each round. To achieve this goal, a flooding algorithm is used. The idea of a flooding algorithm is that each node tries to forward every message to every one of its neighbors. This results in every message eventually being delivered to all reachable parts of the network. However, some precautions have to be taken to avoid duplicate deliveries and infinite loops. Each node should maintain a cache in which to save the identifiers of received messages. The nodes can discard those messages which they have already seen. For example, when a node receives a ServicePost or a ServiceQuery message, it checks the message identifier in its cache. If the message identifier is new, the node saves the message identifier in its cache and resends the message.

For each server s of the network, a Post protocol is adopted. A Post protocol maintains two caches. One is called the message cache, and it saves the message identifiers. The other is called the service cache, and it stores the service information. Before sending a ServicePost message, the type of service offered by this server is added into its service cache. For each client c of the network, a Query protocol is adopted. In addition to the attributes of the Post protocol, a Query protocol also has a *sent time* attribute which memorizes the time the message was sent. This attribute is used for calculating the time a client waits to receive a ServiceReply message.

Figure 3.3 demonstrates the message flows between a server and a client.

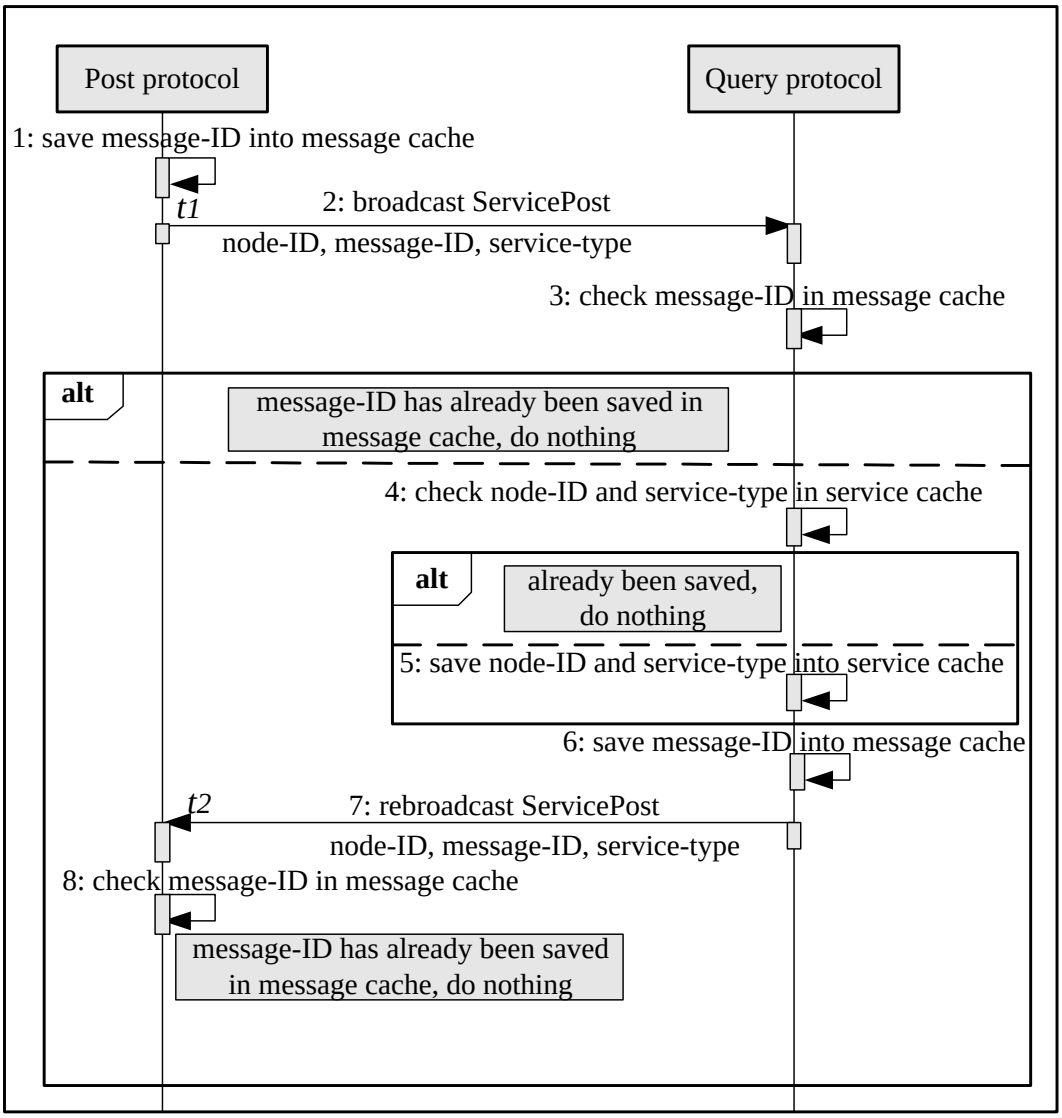

Figure 3.3: Sequence diagram of the greedy strategy: server broadcasts ServicePost message

A Post protocol and a Query protocol are employed in the server and client respectively. In each round, before sending out a ServicePost message, the server stores the message ID into its message cache. At time t_1, the server broadcasts a ServicePost message containing its node ID, message ID and the service type it is offering. When a client which resides within the communication range receives this ServicePost message, it first checks for the message ID in its message cache. If the message ID is found, indicating that the client has already received that same message from the same server, it then does nothing. If the message ID is new, the client first checks the service type and the server's node ID in its service cache. If the client has already saved the same service information before, implying that the client has received the same ServicePost message in a previous round, it ignores this service information. It then saves the message ID into its message cache and rebroadcasts the same ServicePost message. If the node ID and service type are new to the client, it saves them into its service cache, stores the message ID into its message cache and rebroadcasts the same ServicePost message. At time t_2, the server receives the rebroadcast ServicePost message which it had originated by itself. It checks its message cache and finds that the message ID has already been saved before. Then, the server keeps silent. By this means, infinite loops are avoided. As discussed in Section 3.3, the same message flows could occur between two servers. A server could also receive other servers' ServicePost messages and save the service information into its service

cache. However, this server can only post the services which it can offer to the other nodes, while it can be queried for other cached services.

Figure 3.4 illustrates the message flows between a client and a server. A Query protocol and a Post protocol are employed in the client and server respectively. In each round, before sending out a ServiceQuery message, the client stores the message ID into its message cache. At time t_1, the client broadcasts a ServiceQuery message, which consists of its node ID, message ID and the service type which it is querying. When a server which resides within the communication range receives this ServiceQuery message, it first checks the ID of this message in its message cache. If it is found, implying that the server has already received that same message from the same client, it then does nothing. If the message ID is new, the server first checks the service type queried by the client in its service cache. If the server does not have the queried service type, it then saves the message ID into its message cache and rebroadcasts this ServiceQuery message. If the required service is found, the server first unicasts a ServiceReply message to the client, and then stores the message ID into its message cache and rebroadcasts this ServiceQuery message. The waiting time of the client is t_2 minus *sent time* t_1 if the client receives a ServiceReply message, or the interval of the round if the client receives no ServiceReply message during this round. At time t_3, the client receives the rebroadcast ServiceQuery message which it had originated by itself. It checks its message cache and finds that the message ID has already been

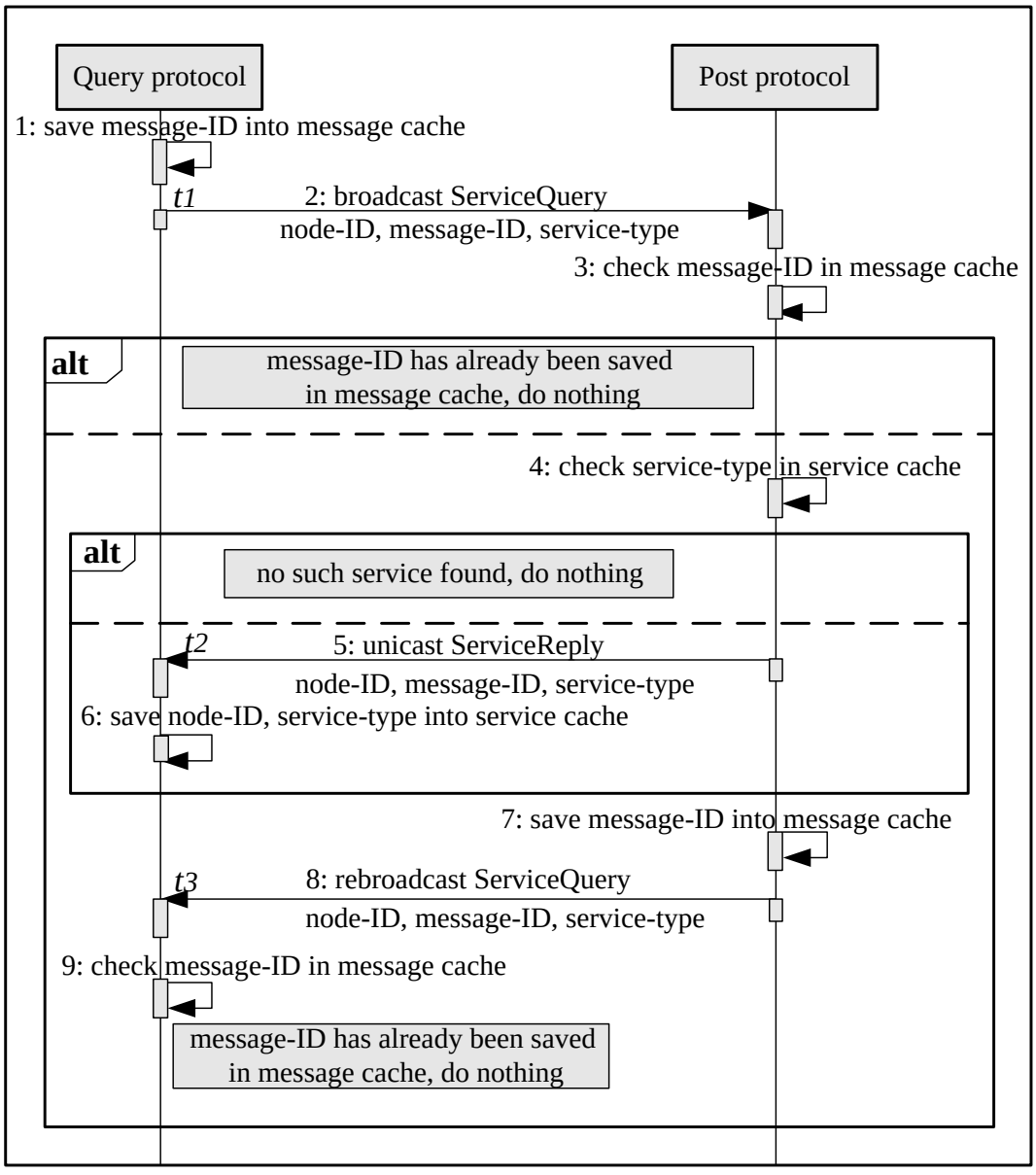

Figure 3.4: Sequence diagram of the greedy strategy: client broadcasts Service-Query message

saved before. Then the client keeps silent. As interpreted in Section 3.3, the same message flows could occur between two clients. A client could also receive other clients' ServiceQuery messages. They may have cached the required service information. Furthermore, a client could also send a ServiceReply message to another client.

3.5 The conservative Post-query strategy

Based on the greedy strategy, we define another strategy called the conservative Post-query strategy. This conservative strategy requires that in each round, all the servers (clients) post to (query from) their one-hop neighbors in the network. As was mentioned in Section 3.4, this strategy can directly use the one-hop broadcast mechanism provided by the underlying link layer protocol. It uses the DSR protocol and DSDV protocol to deliver service reply messages to the senders. Unlike the greedy strategy, in the conservative strategy a Post protocol does not need a message cache to store all the IDs of the received messages, since the conservative strategy requires each node to do one-hop broadcast and no rebroadcast. A Post protocol employed in a server using the conservative strategy has a service cache attribute and a Query protocol employed in a client of this strategy has a service cache and a sent time attribute.

Figure 3.5 shows the message flows between a server and a client. The server and the client adopt a Post protocol and a Query protocol respectively. At time t_1,

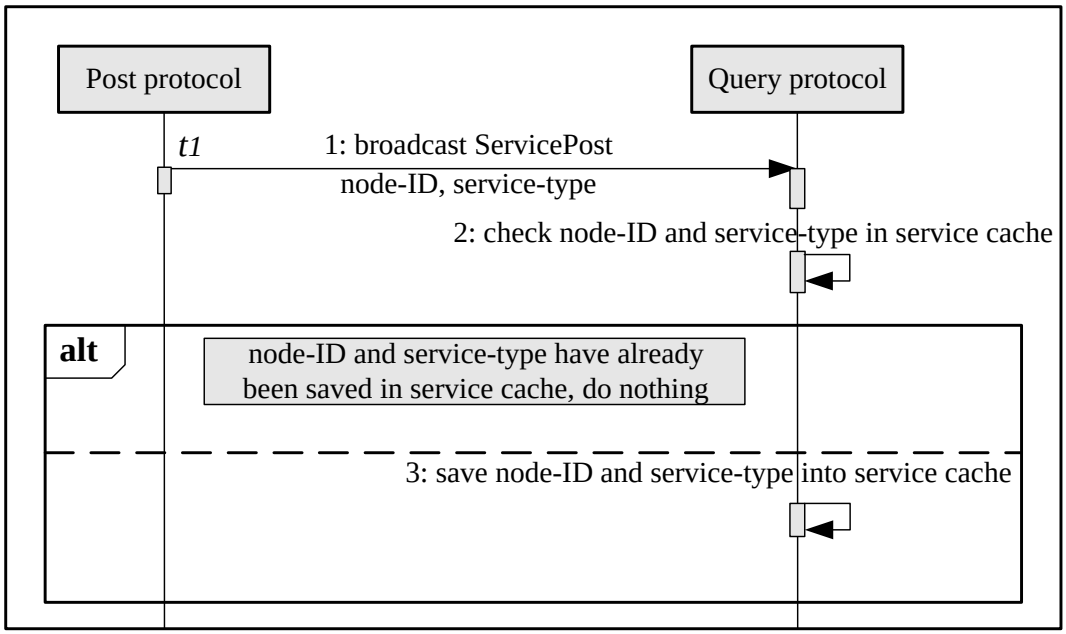

Figure 3.5: Sequence diagram of the conservative strategy: server broadcasts ServicePost message

the server and the client are within communication range. The server broadcasts a ServicePost message containing its node ID and offered service type. When the client receives this ServicePost message, it first checks the node ID and service type in its service cache. If the same service information has been saved before, then the client does nothing. Otherwise, it stores the node ID and service type into its service cache. The same message flows could occur between two servers, as described in Section 3.4.

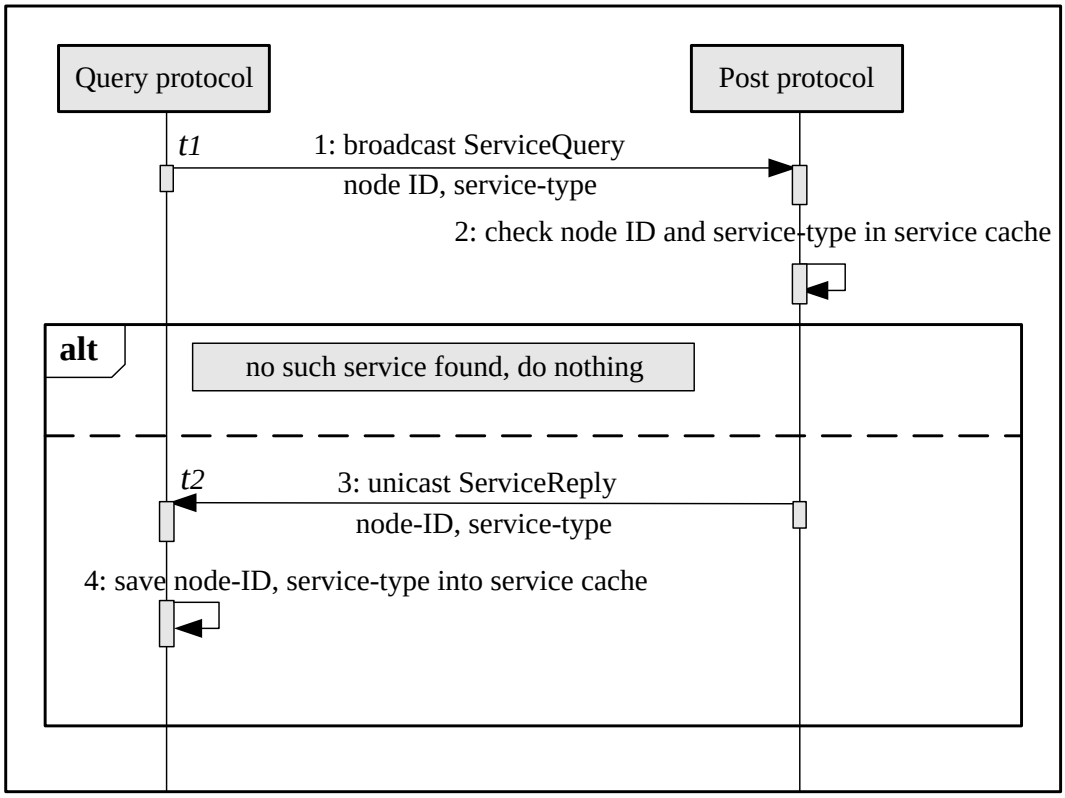

Figure 3.6: Sequence diagram of the conservative strategy: client broadcasts ServiceQuery message

Figure 3.6 illustrates the message flows between a client and a server. A

Query protocol and a Post protocol are employed in the client and the server respectively. At time t_1, the client and the server are within communication range. The client broadcasts a ServiceQuery message which consists of its node ID and the service type which it is querying. When the server receives this ServiceQuery message, it first checks in its service cache to see if it has the same service information. If it cannot find the queried service, then it remains quiet. If the service which the client queries is found in its service cache, at time t_2 the server unicasts a ServiceReply message to the client. At that time, they may still be within communication range or may have moved multiple hops away from each other. The waiting time of the client t is equal to t_2 minus t_1 if the client receives a ServiceReply message, or the interval of the round if the client receives nothing. The same message flows could occur between two clients, as noted in Section 3.4.

3.6 The incremental and uniform memoryless Post-query strategies

We put these two strategies together because they have the same message flows and both use the network unicast communication; however, the size of posting and querying sets are different. In the incremental strategy, all the servers (clients) in the network begin posting to (querying from) a small set of randomly chosen

nodes in the first round. As the number of rounds increases, the size of this set gradually increases too. In the uniform memoryless strategy, all the servers in the network post their services to l nodes, and all the clients query l' nodes, where l and l' are less than or equal to the total number of nodes in the network and are positive integers. This consists of rounds of uniform repetitions of the $(l,\ l')$ post-query protocol. A Post protocol and a Query protocol in these two strategies have the same attributes as those in the conservative strategy. In Figures 3.7 and 3.8, we omit the part common with Figures 3.5 and 3.6.

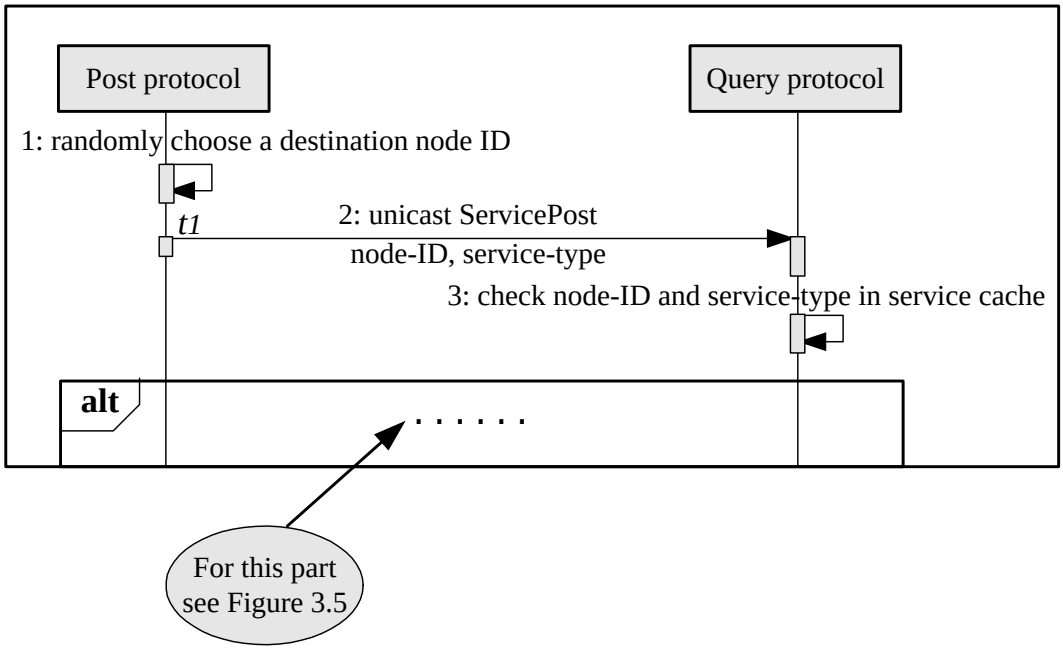

Figure 3.7: Sequence diagram of the incremental and uniform memoryless strategies: server unicasts ServicePost message

Figure 3.7 reviews the message flows between a server and a client. A Post protocol is adopted in the server and a Query protocol is employed in the client.

The server randomly chooses a client ID as its destination. At time t_1 the server and the client could be either within communication range or multiple hops away from each other. When the client receives the ServicePost message sent by the server, it checks for the service information in its service cache. If the service information is found, the client then does nothing. If not, the client stores this service information in its service cache. The rest of the figure is identical to Figure 3.5. The same message flows could occur between two servers as were described in Section 3.4.

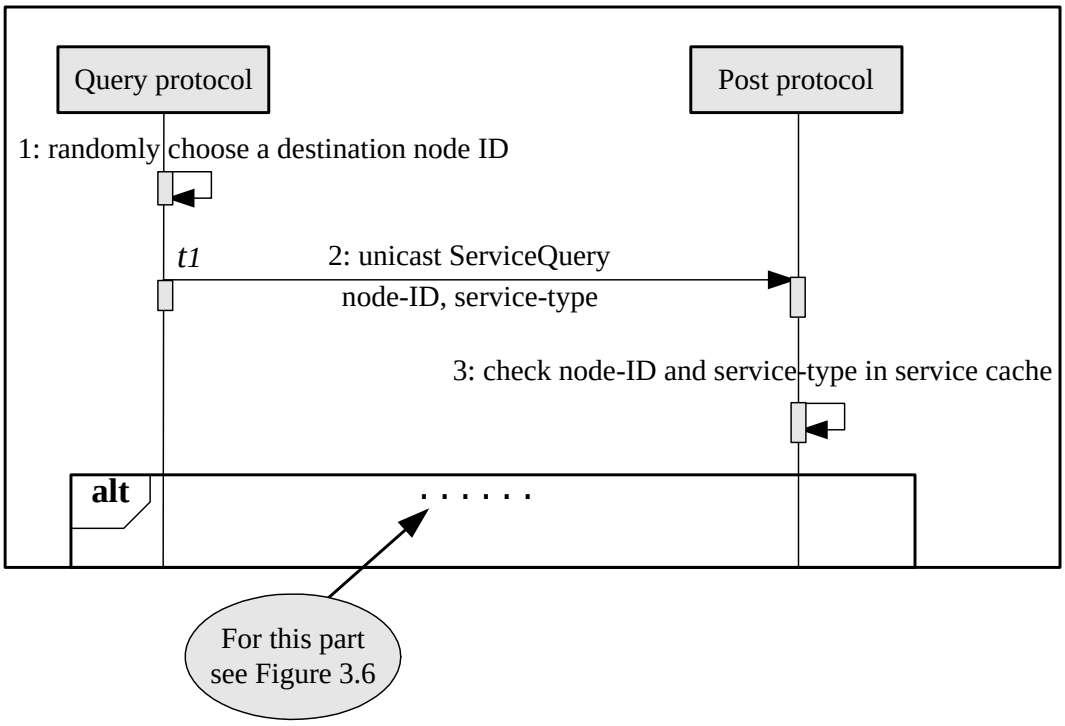

Figure 3.8: Sequence diagram of the incremental and uniform memoryless strategies: client unicasts ServiceQuery message

Figure 3.8 demonstrates the message flows between a client and a server. A

Query protocol is carried by the client and a Post protocol is adopted by the sever. The client randomly chooses a server ID as its destination. At time t_1 the client and the server could be either within communication range or multiple hops away from each other. When the server receives the ServiceQuery message sent by the client, it checks for the service information in its service cache. The rest of this figure is identical to Figure 3.6. The same message flows could occur between two clients as discussed in Section 3.4.

3.7 The with memory Post-query strategy

For a service discovery strategy, our main concern is that a greater number of nodes find their services. Obviously the aforementioned two strategies, the incremental and uniform memoryless strategies, are not sufficient to resolve this concern, if posting and querying sets are relatively small. To solve this problem, we can introduce another cache to each node to store the IDs of the nodes which have not been visited before. In this strategy, in each round all the servers in the network post their services to l nodes, and all the clients query l' nodes, where l and l' are positive integers and are less than or equal to the total number of nodes in the network. Unicast communication is used. Each round only involves the nodes which have not been visited before. Compared with the uniform memoryless Post-query strategy, Post and Query protocols in the *with memory* strategy have one more cache. In Figures 3.9 and 3.10, we also omit those sections in

common with Figures 3.5 and 3.6.

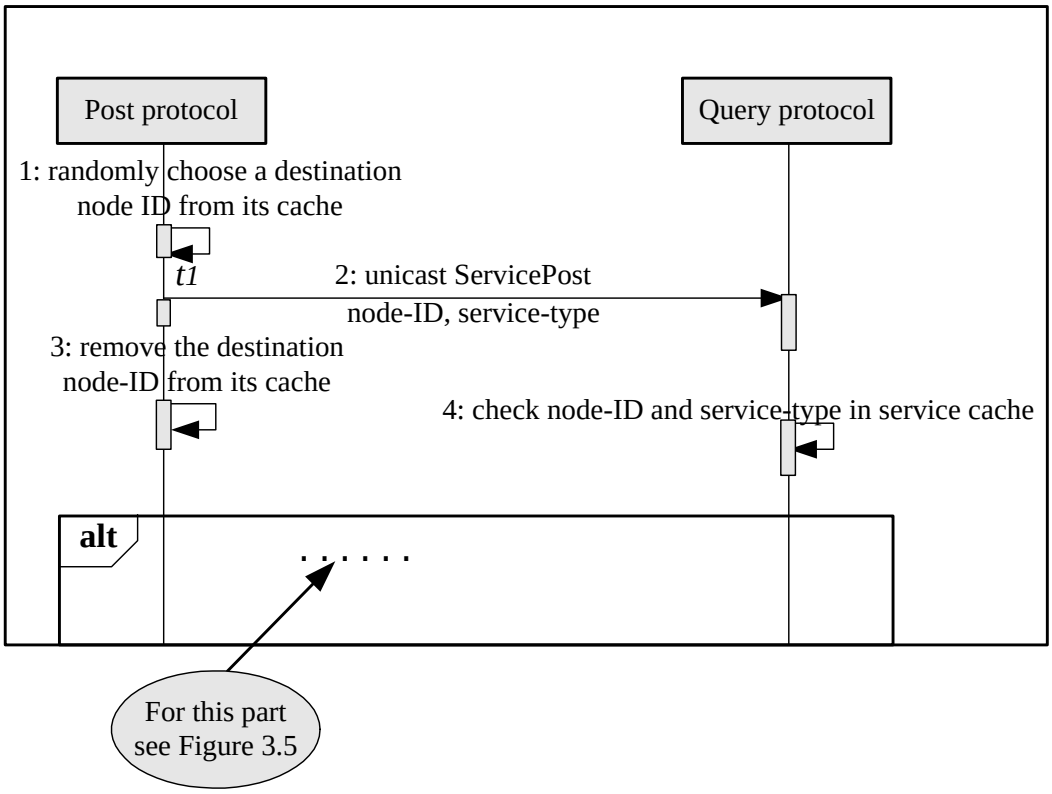

Figure 3.9: Sequence diagram of the *with memory* strategy: server unicasts ServicePost message

Figure 3.9 depicts the message flows between a server and a client. A Post protocol is adopted in the server and a Query protocol is employed in the client. The server randomly chooses a client ID as its destination from its untouched cache. At time t_1 the server and the client could be either within communication range or multiple hops away from each other. After the server unicasts a ServicePost message to the chosen client, the server removes the client ID from its cache so that the next time the server randomly chooses a destination node

ID, only those nodes which have not been contacted with will be involved. When the client receives the ServicePost message sent by the server, it checks for the service information in its service cache. If the service information is found, the client then does nothing. If not, the client stores this service information in its service cache. The rest of the figure is identical to Figure 3.5. The same message flows could occur between two servers as described in Section 3.4.

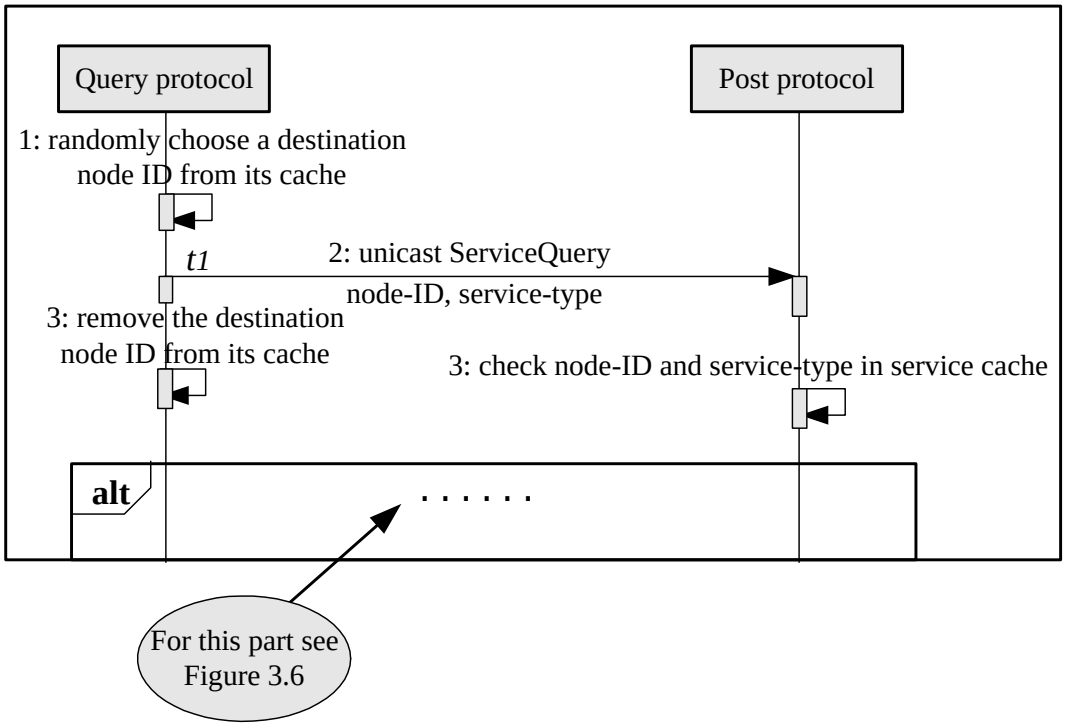

Figure 3.10: Sequence diagram of the *with memory* strategy: client unicasts ServiceQuery message

Figure 3.10 presents the message flows between a client and a server. A Query protocol is carried by the client and a Post protocol is adopted by the sever. The client randomly chooses a server ID as its destination from its untouched cache.

At time t_1 the client and the server could be either within communication range or multiple hops away from each other. After the client unicasts a ServiceQuery message to the chosen server, the client deletes the server ID from its cache. Thus, when the server wants to randomly choose another destination node, only those nodes which have not been visited may be chosen. When the server receives the ServiceQuery message sent by the client, it checks for the service information in its service cache. The rest of this figure is identical to Figure 3.6. The same message flows could occur between two clients as discussed in Section 3.4.

Chapter 4

Simulation and Implementation

In this chapter we introduce the simulation environment for these five Post-query strategies. We use a well known simulation tool called the Network Simulator (NS). We then present some information on how the scenarios are chosen, followed by a description of the selection of all the simulation parameters. According to the designs from the previous chapter, we show the implementation of these five Post-query strategies in NS.

4.1 Simulation environment

In our research, the functional requirements for a simulation tool to evaluate the greedy, conservative, incremental, uniform memoryless and with memory Post-query strategies, combined with the DSR protocol and DSDV protocol are as follows. First, the simulation tool should support our architecture design as

depicted in Section 3.1. This means an ideal simulation tool should permit the addition of new application protocols easily. Moreover, underlying ad hoc routing protocols such as the DSR protocol and DSDV protocol should be available. The integration of the application protocol and routing protocols should not require significant effort. Second, our context is ad hoc networking, so the simulation tool should be capable of generating scenarios which can model different ad hoc network scenarios. Various simulation scenarios should also be created for the evaluation of the performance of the five Post-query strategies. Finally, from the design of these five strategies in Chapter 3, it can be noted that they either use a network broadcast mechanism or a network unicast mechanism. This requires that the routing protocols and underlying link layer protocols in the simulation tool support these two communication mechanisms.

The *Network Simulator* (NS) is chosen as our simulation tool because it meets our functional requirements. The current version is NS-2 [Fal97]. It is an object-oriented, discrete event driven network simulator developed at UC Berkeley, written in the C++ and OTcl languages. The Monarch research group at Carnegie-Mellon University developed support for the simulation of multihop wireless networks complete with the physical, data link, and medium access control (MAC) layer models in NS-2 [Bro98]. NS-2 has an approach which separates the implementation of basic network component objects from the implementation of the control objects. With respect to the incremental and uniform memoryless

Post-query strategies, they allow for common network component objects and common data flows. Only the objects which are responsible for controlling postings and queryings are different. Our implementation of these two strategies benefited from this aspect of NS-2. In NS-2, to achieve the separation of data and control components while maintaining the communication between them, C++ objects are available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects. The control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. For C++ objects that have an OTcl linkage forming a hierarchy, there is a matching OTcl object hierarchy very similar to that of C++.

In NS-2, the Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs is used as the MAC layer protocol. The wireless interface functions similar to the Lucent WaveLAN radio interface [Tuc93]. The transmission range is about 250 meters. The signal propagation model combines both a free space propagation model and a two-ray ground reflection model.

4.2 Movement scenarios

The main goal of this research is to measure how network topology changes in an ad hoc network affect the performance of the five Post-query strategies, when they are combined with the DSR protocol and DSDV protocol. Each ad hoc network

in our simulation consists of 50 wireless nodes, moving over a square (640 m x 640 m) flat space. The *random waypoint mobility model* [Joh96] is used to generate different node movement scenarios. There are two parameters in this model: one is *pause_time* and the other is the maximum node speed *max_speed*. The model works as follows. Each node of the ad hoc network begins the simulation by remaining fixed for *pause_time* seconds, and then moves to a randomly chosen destination within our simulation area at a speed v distributed uniformly between zero and *max_speed*. When it reaches the destination, the node stops moving and remains still for up to *pause_time* seconds. It then randomly chooses another destination within our simulation area, and repeats the procedure as previously described for the duration of the simulation process. We can note that the pause time and maximum node speed determine the dynamic topology changes of an ad hoc network. A high pause time and a low maximum node speed result in a low number of topology changes while an ad hoc network with a low pause time and a high maximum node speed has a high number of topology changes. The mobile nodes are initially uniformly distributed around the simulation area, but as the simulation time elapses, the uniform distribution can not be assumed. In our simulation, the movement scenario files are generated for three different pause times: 30, 100 and 300 seconds. We use three different maximum node speeds for node movement: one meter per second, 10 meters per second and 30 meters per second. We choose three different mobility models for our simulation:

Scenario 300-01 which has a 300 second pause time and one m/s maximum node speed; Scenario 100-10 which has a 100 second pause time and 10 m/s maximum node speed; and Scenario 30-30 which has a 30 second pause time and 30 m/s maximum node speed. To characterize the mobility of the nodes of an ad hoc network, a parameter called *dynamic ratio* (DR) is introduced. To calculate it, we denote the total number of nodes in the network as N and the total number of link changes of all the nodes during the entire simulation time as LC. The dynamic ratio is then defined as:

$$DR = LC/N$$

Scenario	**Link changes**	**Route changes**	**Dynamic ratio**
300-01	403	905	8.06
100-10	3074	7080	61.48
30-30	9536	21259	190.72

Table 4.1: The link changes, route changes and dynamic ratios of the three mobility models

The link changes and route changes of these three different mobility models, as well as the dynamic ratios, are shown in Table 4.1. Hereafter, we use the *dynamic ratio* to indicate the degrees of the topology changes of an ad hoc network. The scenario 30-30 has the highest number of route changes and link changes and the highest dynamic ratio. The scenario 300-01 has the fewest route changes and link changes of the three scenarios, and the lowest dynamic ratio. The scenario 100-10 is in-between scenario 30-30 and scenario 300-01.

4.3 Parameter selection

Many parameters are involved in our simulation, some of which are described in the previous section. Before we describe the implementation of these five Post-query strategies, certain parameter values need to be selected. We synthesize these parameters in Table 4.2.

Parameter name	**Parameter value**
Number of nodes	50
Simulation area	640 $m \times 640$ m
Node density	1/8192 m^2
Transmission range	250 m
Simulation time	600 s
Maximum node speed	1 m/s, 10 m/s, 30 m/s
Pause time	300 s, 100 s, 30 s
Background traffic	null
Number of servers	10
Number of clients	40
Number of services	10
Number of cases	10
Number of runnings	10
Maximum round	10
Round interval	15 s
Posting(querying) interval	15 s
Routing protocols	DSR / DSDV protocol

Table 4.2: Simulation and implementation parameters

There are 50 nodes in the network. Ten of them act as servers and 40 of them perform the role of clients. Ten types of services are provided. We assume that each server provides one type of service and that all network services follow a uniform distribution, such that each kind of service is requested with equal probability. For each Post-query strategy, we have two different ad hoc routing

protocols and three different mobility models. For each mobility model, we generate ten different scenario cases, each of which is run ten times. Therefore, the total number of simulations is $5 \times 2 \times 3 \times 10 \times 10 = 3000$. The Post-query strategies belong to the application layer protocols. Hence, background traffic is not necessary to evaluate the performance of these Post-query strategies. We denote the round interval and posting (querying) interval as 15 seconds. We assume that the lifetime of all three types of messages is equal to the duration of a round times the maximum number of rounds. To achieve a fair comparison, we configure these five Post-query strategies with the same simulation and implementation parameters.

4.4 Implementation of the five Post-query strategies

In Chapter 3, the designs of the greedy, conservative, incremental, uniform memoryless and *with memory* Post-query strategies are described in detail. In Section 4.1, we introduced the NS-2 simulation environment. The implementation of these strategies, combined with the DSR protocol and DSDV protocol in NS-2, should succeed in separating of the control mechanism from the message processing mechanism. In the next sections, we briefly present how we achieve this separation using the C++ and OTcl languages. Our implementation has a data

processing component in C++ and a control component in OTcl. We show that how they are combined together in Figure 4.1.

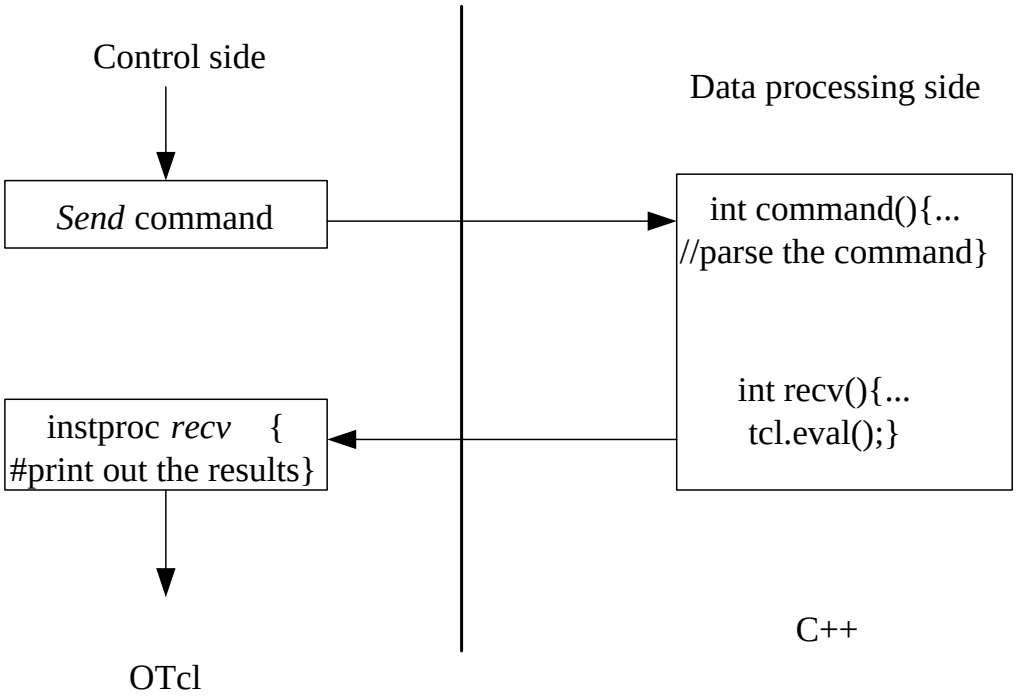

Figure 4.1: Combination of the components of the data and control sides

4.4.1 The implementation of the Post and Query agents

Packet sending and receiving in NS-2 are handled by a set of protocol agents. For the Post-query protocol, the Post agent and Query agent are coded in the C++ language. Instances of the Post (Query) agent run on server (client) nodes. Considering the functional requirements of these five Post-query strategies, we developed two sets of Post agents and Query agents. One set of agents is for the greedy strategy. The Post and Query agents in this strategy have the command(),

recv(), hasService() and hasMessageID() C++ methods. The second set of agents is for the conservative, incremental, uniform memoryless and *with memory* Post-query strategies. The Post and Query agents in these four strategies have the command(), recv() and hasSerivce() C++ methods.

- The command() method is common to all the agents. It plays the role of an interface between OTcl and C++. This method is a transmission operation. For a Post agent, it generates ServicePost messages and broadcasts them. For a Query agent, it creates SerivceQuery messages and sends them out.

- The hasService() method is a helper operation. The Post agent and Query agent use this method to determine if a given posted (queried) service type has been saved in the service cache.

- The hasMessageID() method is another helper operation specifically used in the greedy strategy for infinite loop avoidance in the flooding algorithm. It checks the IDs of the received ServicePost or ServiceQuery messages in the message cache. If the message ID has already been saved, then the Post and Query agents in the greedy strategy do not rebroadcast the received messages. The hasMessageID() method is not necessary for the Post and Query agents of the conservative, incremental, uniform memoryless and *with memory* strategies because they do not need to rebroadcast the messages.

- The recv() method is mainly responsible for processing the incoming Post-

query messages. In the greedy strategy, the recv() method in a Post agent checks the service information obtained from the received ServicePost and SeviceQuery messages. It unicasts ServiceReply messages to other Query agents based on the results of the hasService() method. It rebroadcasts these messages according to the results of the hasMessageID() method. In addition, for a Query agent, the recv() method also processes received ServiceReply messages from other Post agents or Query agents. In the other four Post-query strategies, rebroadcast is not necessary, so the recv() method in their Post and Query agents only takes charge of dealing with the received messages and sending out ServiceReply messages. For the sake of performance evaluation, when a ServiceReply message is received by a Query agent, the recv() method calculates the waiting time .

The method *command()* is called after an OTcl *send* command is executed on the control side to send a Post-query message between a Post agent and Query agent. The *send* command is then parsed by the *command()* method in C++. During the execution of the *recv()* method in C++, an OTcl *recv* procedure on the control side is called from the data processing side using *tcl.eval()*. It prints out the result if there is a ServiceReply message received.

4.4.2 The implementation of the message control objects

NS-2 splits the C++ and OTcl objects, which allows us to avoid changes at the C++ level if new functionality is needed at the OTcl level. We developed the control mechanism for the five Post-query strategies according to the control functional requirements. The greedy and conservative strategies have the same control mechanisms because they both use network broadcast communication. As for the remaining three Post-query strategies, although each strategy uses a network unicast mechanism, either the sizes of posting and querying sets of each round or the ways of selecting the posting and querying destinations are different. The different control functional requirements result in different message control mechanisms even if they share common C++ components. The control mechanisms are pictured in Figure 4.2.

First, all the nodes in the network are created. The predefined network scenario file and service type pattern are then loaded. The network scenario files are generated using the random waypoint algorithm. We define the service type pattern which indicates the service type each server can offer and the service type each client wants to locate. A Post agent is attached to each server s and a Query agent attached to each client c using the *attach-agent* command. The *connect* command builds a communication channel between each server and each client. We detail four types of sending and receiving mechanisms in the five Post-query strategies.

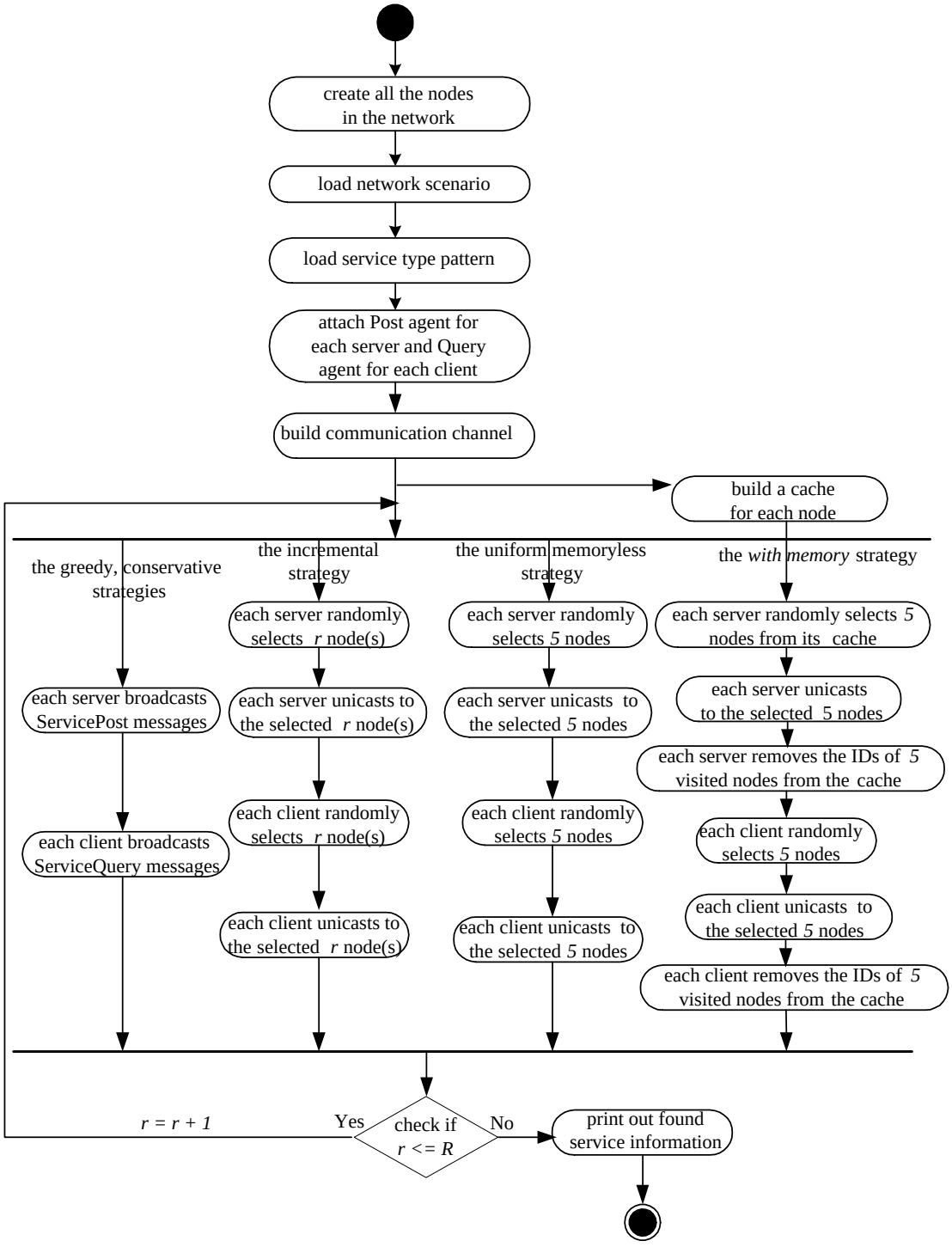

Figure 4.2: Activity diagram of the control mechanism of the five Post-query strategies

- The greedy and conservative strategies require that in each round, ServicePost messages are first broadcast by each server and ServiceQuery messages are then broadcast by each client. The rebroadcast mechanism designed in the greedy strategy is implemented in a data processing object because in NS-2 the dynamic control is not supported in OTcl. The greedy strategy is implemented as a Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter. The conservative strategy is implemented as a Post-broadcast-h, query-broadcast-h strategy with h equal to one, where h is the number of broadcast hops.

- The incremental strategy requires that in a round with index r, each server randomly chooses r node(s) to be unicast the ServicePost messages. Each client randomly chooses r node(s) to be unicast the ServiceQuery messages. As the index of rounds r increases, the number of nodes to be selected increases too. This strategy is implemented as a Post-incremental, query-incremental strategy.

- The uniform memoryless strategy requires that in a round with index r, each server randomly chooses and unicasts to five nodes the ServicePost messages and each client randomly selects and unicasts to five nodes the ServiceQuery messages. As the index of rounds r increases, the number of chosen nodes in each round remains the same (five). This strategy is

implemented as a Post-to-l, query-l' strategy where l and l' are positive integers and are equal to five.

- The *with memory* strategy has the same l and l' as the uniform memoryless strategy. A fresh cache for each node in the network is introduced. This cache stores all the IDs of the nodes in the network. At first, each server randomly selects five nodes from its cache. After these selected five nodes are unicast with the ServicePost messages, their IDs are removed from the cache so that they will not be selected in the next round. Then, each client randomly chooses five nodes from its cache to unicast ServiceQuery messages to. After unicast, the IDs are deleted from the cache to make sure that in the next round only the uncontacted nodes are involved. This strategy is implemented as a Post-to-l, query-l' strategy with visited nodes, a modified Post-to-l, query-l' strategy.

Finally, the service information from the ServiceReply messages is printed out, including the ID of the client which sends ServiceQuery message, the service type which the client queries, the ID of the server which offers the corresponding service and the waiting time of the client.

Chapter 5

Performance Evaluation

In the previous chapters, we introduced the design, implementation and simulation of the greedy, conservative, incremental, uniform memoryless and with memory Post-query strategies combined with the DSR routing protocol and DSDV routing protocol. This chapter investigates the performance of these five strategies. The performance evaluation is carried out through three performance metrics: success rate, number of transmitted messages and average waiting time. Furthermore, a performance comparison of all these five strategies combined with the DSR protocol and the DSDV protocol is given.

5.1 Performance metrics

To compare the performance of the greedy, conservative, incremental, uniform memoryless and with memory Post-query strategies combined with the DSR pro-

tocol and DSDV protocol, the following three metrics are used. We denote the number of total nodes as N, the number of total clients as N_c, and the number of total successful clients as N_{succ}. For each round r, we denote the number of ServicePost messages as M_{post}, the number of ServiceQuery messages as M_{query}, and the number of ServiceReply messages as M_{reply}. Each successful client c receives 1 to m ServiceReply messages of waiting time $t_1,\ t_2,\ \ldots,\ t_m$.

- Success rate(SR): The ratio (as a percentage) of the number of clients which successfully locate the services, over the total client number. It is calculated using the following formula:

  $SR = N_{succ}/N_c \times 100\ (\%)$

- Number of transmitted messages(NMT): The number of messages transmitted in each round by all the nodes in the network, including ServicePost, ServiceQuery and ServiceReply messages. It is calculated using the following formula:

  $NTM = \sum_{n=1}^{N} \sum(M_{post},\ M_{query},\ M_{reply})$

- Average waiting time(AWT): The minimum time period in seconds, averaged over all the clients, starting from the sending of a ServiceQuery message and ending with the receiving of a ServiceReply message. A client which cannot find the service after r rounds has the waiting time r times round interval seconds. It is calculated using the following formula:

$$AWT = \frac{\sum_{n=1}^{N_c} min(t_1,t_2,...,t_m)}{N_c} \ (s)$$

For a Post-query strategy, good performance means a high success rate, a short average waiting time and a low number of transmitted messages.

5.2 Results and discussion

5.2.1 Greedy: Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter

In this strategy, all the servers post to all the nodes and all the clients query all the nodes in the network using a network broadcast mechanism based on a flooding algorithm. Once a node in the network receives a ServicePost or a ServiceQuery message, it checks the message identifier in its cache. If the message identifier is new, the node rebroadcasts the message and saves the message identifier in its cache. The number of rebroadcast hops h is the diameter of the network. This strategy is greedy because it consumes significant network resources, in terms of the number of transmitted messages, to achieve a high success rate. It has the highest number of transmitted messages and the lowest average waiting time of all the strategies.

From Figures 5.1 and 5.2, we can observe three features of this strategy. First, this strategy can achieve a very high success rate, up to 100%, when it is combined

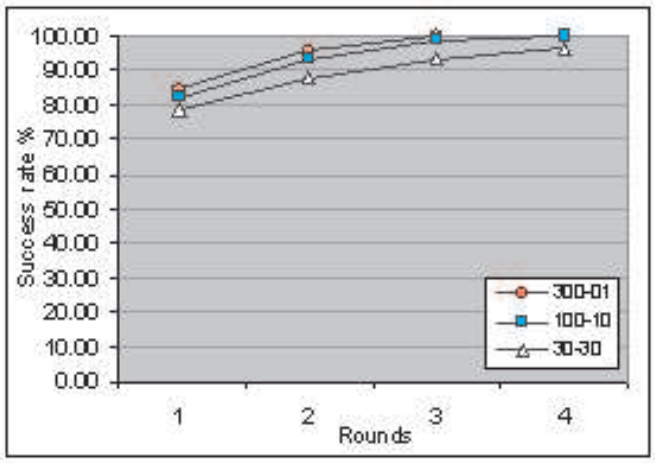

Figure 5.1: Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter DSR success rate

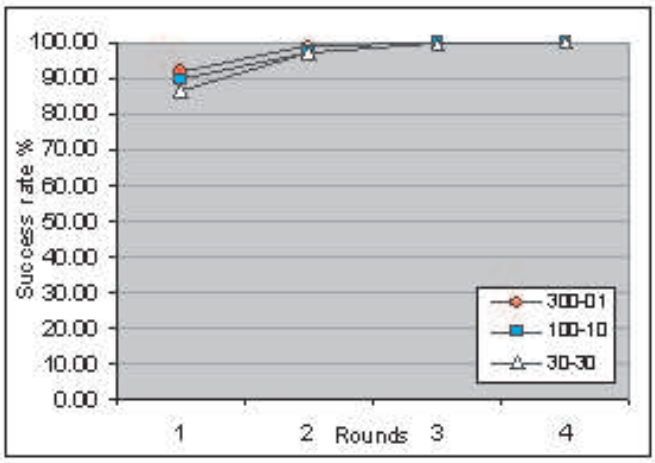

Figure 5.2: Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter DSDV success rate

with the DSR protocol or DSDV protocol. The flooding algorithm used in this strategy results in a large number of nodes being posted to or queried from within a low number of rounds.

Second, we find that the higher the dynamic ratio of the network is, the lower the success rate of this strategy is. This feature can be explained by studying the characteristics of these two routing protocols. As a table-driven routing protocol, the DSDV protocol attempts to maintain consistent, up-to-date routing information from each node to every other node in the network. It requires that each node maintain one or more tables to store routing information and to respond to changes in network topology by periodically propagating updates throughout the network. For an ad hoc network with a constant number of nodes, e.g. $N = 50$, the routing overhead in the three dynamic ratios is very similar. A network with a higher dynamic ratio has slightly more routing overhead than a network with a lower dynamic ratio. However, the packet delivery ratio drops greatly as the dynamic ratio of the network becomes higher, as Broch et al. [Bro98] found in their simulations. So when combined with the DSDV protocol, the success rate of this strategy decreases when the dynamic ratio of the network becomes higher. As a source-initiated on-demand ad hoc routing protocol, the DSR protocol creates routes only when they are needed by source nodes. There are no periodic routing advertisements in the protocol. Instead, when a node needs a route to another node, it dynamically determines one based on cached

information, or on the result of a route discovery process. For an ad hoc network with a constant number of nodes, the packet delivery ratio is very high regardless of the dynamic ratio of the network. However, the routing overhead of the network significantly increases when the dynamic ratio of the network becomes higher. Table 4.1 shows that the higher the dynamic ratio of the network, the higher the number of route changes and link changes in the network, which causes greater routing overhead. Hence, when this strategy is combined with the DSR protocol, the success rate decreases as the dynamic ratio of the network increases.

Finally, we notice that when this strategy is combined with the DSDV protocol, it can achieve a higher success rate than when combined with the DSR protocol. This strategy's success rate with the DSDV protocol is 0.42% higher than the one with the DSR protocol. We can synthesize the features of this greedy strategy and the two routing protocols to clarify this observation. This strategy uses a flooding algorithm which requires that all nodes post to all nodes and all nodes query all nodes in the network. Thus, the underlying routing protocols are required to accommodate the heavy routing demands. When this strategy is combined with the DSDV protocol, the routing overhead remains constant regardless of the amplitude of the routing demand. Alternatively, under the DSR protocol, the routing overhead increases with the number of routing demands and is larger than when using the DSDV protocol.

From Figures 5.3 and 5.4, we can summarize the following points. First, this

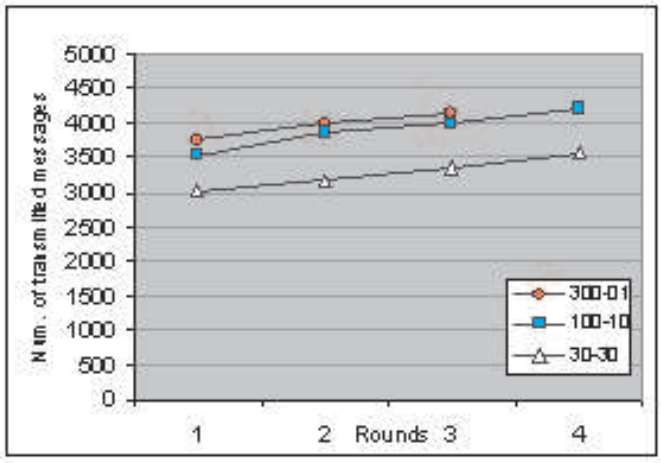

Figure 5.3: Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter DSR num. of transmitted messages

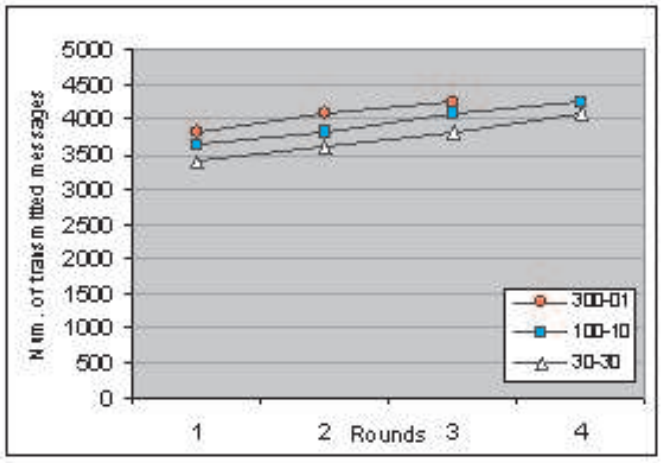

Figure 5.4: Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter DSDV num. of transmitted messages

strategy is very costly in terms of the total number of transmitted messages when it is combined with the DSR protocol or DSDV protocol. From the flooding algorithm used in this strategy, we can compute that the complexity of the number of transmitted messages for each round is $O(N^2)$, where N is the total number of nodes in the network.

Second, we find that in the highest dynamic ratio network scenario (i.e. scenario 30-30), this strategy has the lowest number of transmitted messages of the three dynamic ratios. From our discussion of Figures 5.1 and 5.2, as the network dynamic ratio increases, the DSR protocol has greater routing overhead, and the DSDV protocol has a lower packet delivery ratio. Therefore, the number of transmitted messages decreases.

Finally, when this strategy is combined with the DSR protocol, it has 4.9% fewer transmitted messages than when using the DSDV protocol. In this strategy, the former has more routing overhead than the latter.

From Figure 5.5, we can extract some characteristics of this strategy. First, this strategy has the lowest average waiting time of all the strategies when combined with the DSR protocol or DSDV protocol. The flooding algorithm used here results in a large number of nodes posted to or queried from within a low number of rounds.

Second, when this strategy is combined with the DSR protocol, the average waiting time increases with the dynamic ratio since a higher dynamic ratio in the

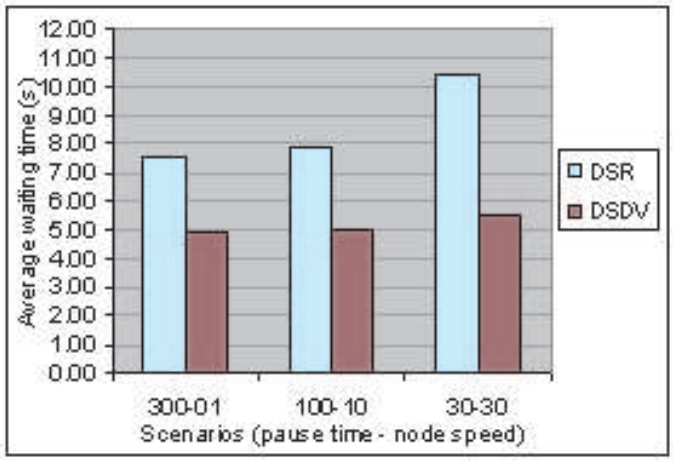

Figure 5.5: Post-broadcast-h, query-broadcast-h strategy with h equal to the network diameter DSR and DSDV average waiting time

network scenario results in more routing overhead in the network, which lengthens the average waiting time. When it is combined with the DSDV protocol, the average waiting times of the three dynamic ratio scenarios are very similar. In scenario 30-30, the average waiting time is slightly longer than that of the other two scenarios, in scenario 100-10, this strategy has a slightly longer average waiting time than the one in the scenario 300-01. The explanation of Figures 5.1 and 5.2 also indicates that as the dynamic ratio of the network scenarios increases, the DSDV protocol has a lower packet delivery ratio; this results in fewer successful clients and those clients which cannot find their services have longer waiting times.

Finally, when this strategy is combined with the DSDV protocol, it has a 40% shorter average waiting time than the one when combined with the DSR

protocol. As the aforementioned characteristics of the DSR protocol and DSDV protocol indicate, the DSR protocol has to accommodate the heavy demands of establishing the routes of all the nodes in the network, which is very time-consuming. Yet the routing demands have little effect on the average waiting time of the strategy when combined with the DSDV protocol. A route is prepared before messages are sent with the DSDV protocol, which saves time when sending a large number of messages in the network.

5.2.2 Conservative: Post-broadcast-h, query-broadcast-h strategy with h equal to one

In this strategy, all the servers in the network post to their one-hop neighbors, and all the clients query their one-hop neighbors using a network local broadcast mechanism. The number of broadcast hops h is equal to one. In contrast to the aforementioned greedy strategy, this strategy is conservative because it significantly reduces the number of transmitted messages. The strategy can also achieve a high success rate in a high dynamic ratio ad hoc network. For ad hoc networks with a lower dynamic ratio, the success rate is low because numerous nodes may not be posted to or queried from at all. This strategy has a low number of transmitted messages and a short average waiting time.

Figures 5.6 and 5.7 highlight the following features of this strategy. First, we observe that this strategy can achieve a very high success rate of 100% when

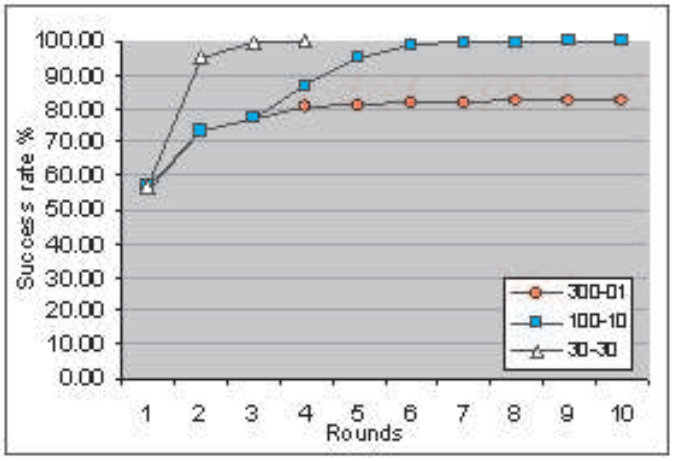

Figure 5.6: Post-broadcast-h, query-broadcast-h strategy with h equal to one DSR success rate

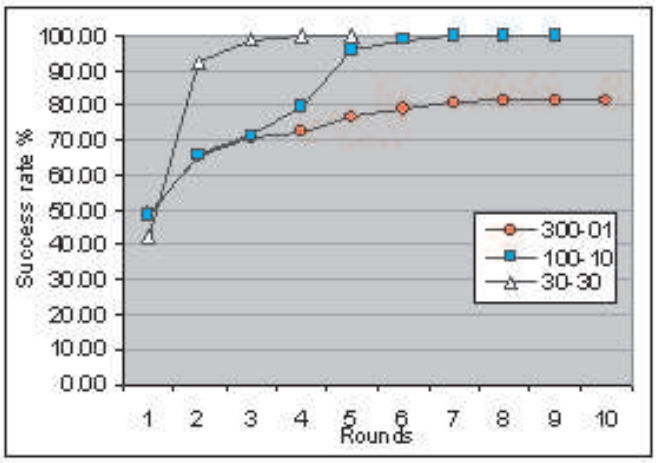

Figure 5.7: Post-broadcast-h, query-broadcast-h strategy with h equal to one DSDV success rate

the dynamic ratio is the highest (i.e. scenario 30-30). When the dynamic ratio is lower (i.e. scenario 100-10), this strategy can obtain a high success rate of 99.75% when combined with the DSR protocol and a very high success rate of 100% with the DSDV protocol. However, in the lowest dynamic ratio network scenario (i.e. scenario 300-01), this strategy can only achieve a success rate of 82.25% with the DSR protocol and 81.5% with the DSDV protocol. When the one-hop broadcast mechanism is used, the higher the dynamic ratio of the network is, the more nodes a server (client) can post to (query from), increasing the success rate of the network. In a very low dynamic ratio scenario or in a low node density scenario, numerous nodes may not be posted to or queried from at all, which explains why the success rate may not reach 100%.

Second, when this strategy is combined with the DSR protocol, it has a 0.33% higher success rate than the one when combined with the DSDV protocol. Using the one-hop broadcast mechanism, the number of routing demands is relatively low, which results in lower routing overhead when using the DSR protocol than that of the DSDV protocol. Under this circumstance, the DSR protocol also has a higher packet delivery ratio than the DSDV protocol.

Figures 5.8 and 5.9 show the following results. First, we observe that this strategy has a low number of transmitted messages when it is combined with the DSR protocol or DSDV protocol in a low dynamic ratio scenario (i.e. scenario 300-01). We also find that in the highest dynamic ratio scenario (i.e. scenario

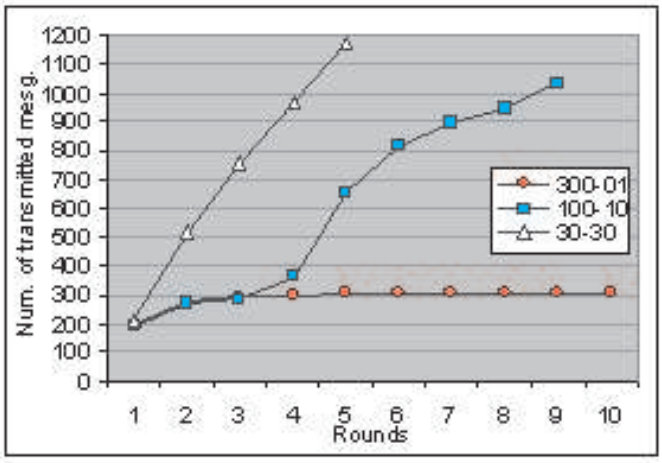

Figure 5.8: Post-broadcast-h, query-broadcast-h strategy with h equal to one DSR num. of transmitted messages

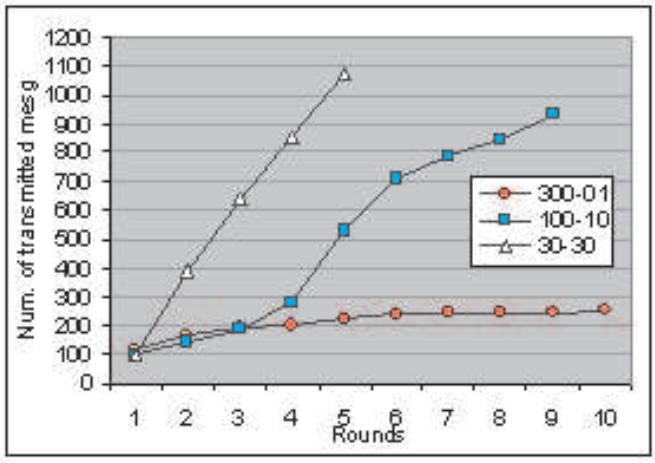

Figure 5.9: Post-broadcast-h, query-broadcast-h strategy with h equal to one DSDV num. of transmitted messages

30-30), this strategy has the highest number of transmitted messages, as a node in this scenario may encounter more nodes than in the other two scenarios. For the same reason, the number of transmitted messages in scenario 100-10 is higher than that in scenario 300-01.

Second, these two figures show that in the lowest dynamic ratio scenario 300-01, the number of transmitted messages remains almost unchanged as the number of rounds increases. This can be explained by noting that when the dynamic ratio of the network is low, the topology of the network is static, and the nodes in the network post to or query from almost the same one-hop neighbors in each round and receive almost the same number of ServiceReply messages.

Finally, when this strategy is combined with the DSR protocol, we notice that it transmits 23.68% more messages than with the DSDV protocol. As we clarified concerning Figures 5.6 and 5.7, the DSR protocol has a lower routing overhead and a higher packet delivery ratio than the DSDV protocol.

The average waiting times of the DSR protocol and the DSDV protocol are shown in Figure 5.10. First, this strategy has a short average waiting time when combined with the DSR protocol or the DSDV protocol. Second, as the dynamic ratio of the scenario becomes higher, the average waiting time becomes shorter. In a lower dynamic ratio network scenario (i.e. scenario 300-01), nearly 20% of clients cannot find their services after this strategy is executed for the maximum number of rounds. Thus, the waiting time is the number of rounds times the

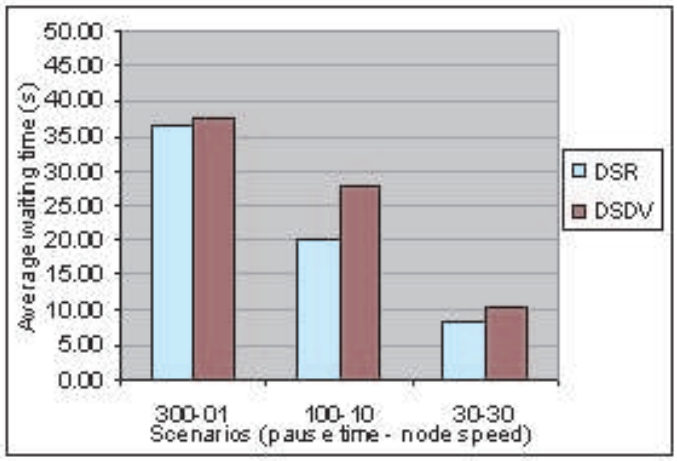

Figure 5.10: Post-broadcast-h, query-broadcast-h strategy with h equal to one DSR and DSDV average waiting time

round interval, which results in a longer average waiting time. In a higher dynamic ratio network scenario (i.e. scenario 30-30), it only takes all the clients four rounds to locate all the services they queried, so the average waiting time is short. Finally, when this strategy is combined with the DSR protocol, it has a 13.79% shorter average waiting time than the one with the DSDV protocol. As noted concerning Figures 5.6 and 5.7, the DSR protocol has lower routing overhead and a higher packet delivery ratio than the DSDV protocol.

5.2.3 Incremental: Post-incremental, query-incremental strategy

In this strategy, all the servers (clients) in the network start posting to (querying from) a small set of randomly chosen nodes in the first round. As the number of

rounds increases, the size of this set gradually increases too. It uses a network unicast mechanism. This strategy has a low number of transmitted messages. It can also achieve a high success rate, while the tradeoff is a long average waiting time. It has the longest average waiting time of all the strategies.

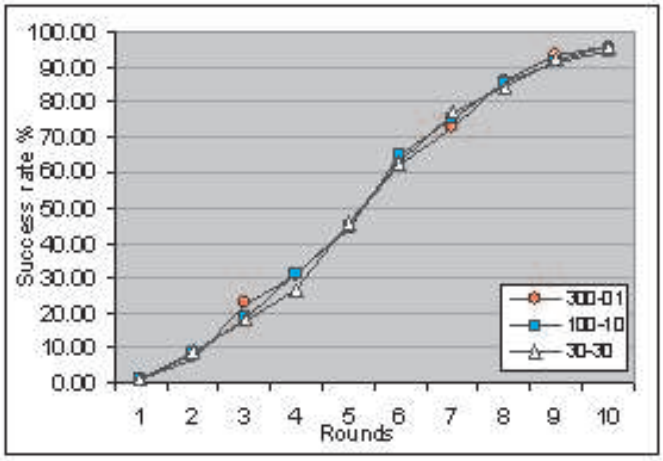

Figure 5.11: Post-incremental, query-incremental strategy DSR success rate

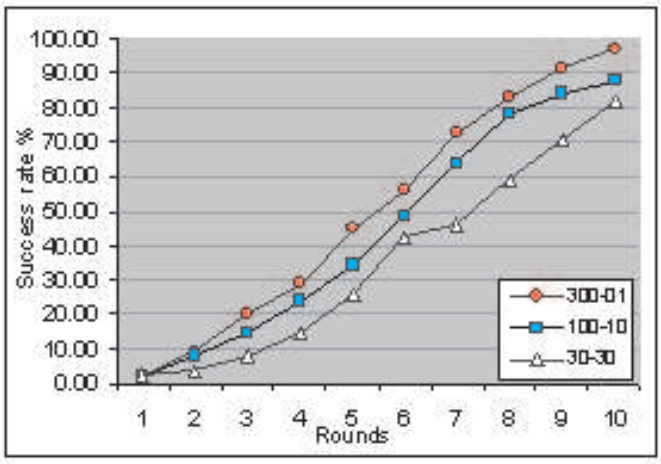

Figure 5.12: Post-incremental, query-incremental strategy DSDV success rate

From Figures 5.11 and 5.12, we can elicit the following observations. First,

although this strategy can hardly achieve a 100% success rate when combined with the DSR protocol or DSDV protocol, it can still achieve a relatively high success rate of 94.75% with the DSR protocol and 81.75% with the DSDV protocol. Second, when this strategy is combined with the DSR protocol, the dynamic ratio of the network scenario has a minimal effect on the success rate. In the three dynamic ratio scenarios, it has a very similar success rate for each round, and it can achieve high success rates of from 94.75% to 96% after ten rounds. However, when the strategy is combined with the DSDV protocol, the dynamic ratio affects the success rate significantly. The higher the dynamic ratio of the network is, the lower the strategy's success rate. For example, it achieves a high success rate of 97% in the lowest dynamic ratio (i.e. scenario 300-01), but only obtains a success rate of 87.75% in scenario 100-10, and gets the lowest success rate of 81.75% in the lowest dynamic ratio scenario (i.e. scenario 30-30). This strategy uses a network unicast mechanism. The size of the posting (querying) set is small, so the routing demands in the underlying routing protocols are very low. Under these circumstances, the DSR protocol has lower routing overhead than the DSDV protocol, which has a relatively constant routing overhead regardless of the routing demands in the same dynamic ratio scenario. With low routing demands, the DSR protocol performs well even in a high dynamic ratio network scenario. As the dynamic ratio of the network increases, the DSDV protocol's packet delivery ratio decreases, causing the success rate to drop. When

this strategy is combined with the DSR protocol, it has a 6.75% higher success rate than the one when combined with the DSDV protocol.

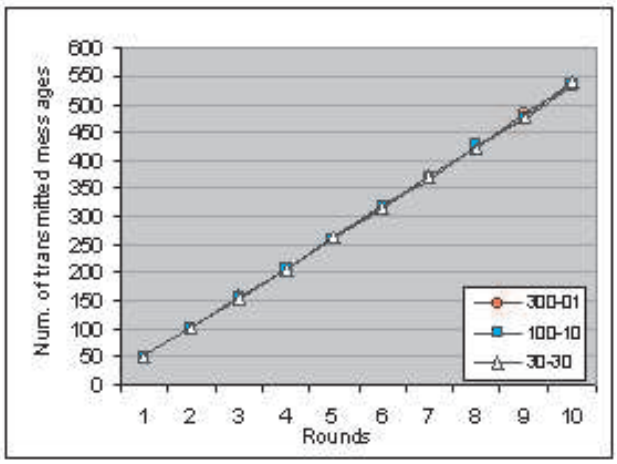

Figure 5.13: Post-incremental, query-incremental strategy DSR num. of transmitted messages

Figures 5.13 and 5.14 show several characteristics of this strategy. First, we observe that this strategy has a low number of transmitted messages when it is combined with the DSR protocol or the DSDV protocol. The number of transmitted messages is in direct ratio with the number of rounds, because we gradually increase the size of the posting (querying) set. Second, the dynamic ratios of the network scenarios have a minimal effect on the number of transmitted messages when this strategy is combined with the DSR protocol. In these three dynamic ratio scenarios, this strategy has nearly the same number of transmitted messages. However, when it is combined with the DSDV protocol, the network with the higher dynamic ratio has slightly fewer transmitted messages than the

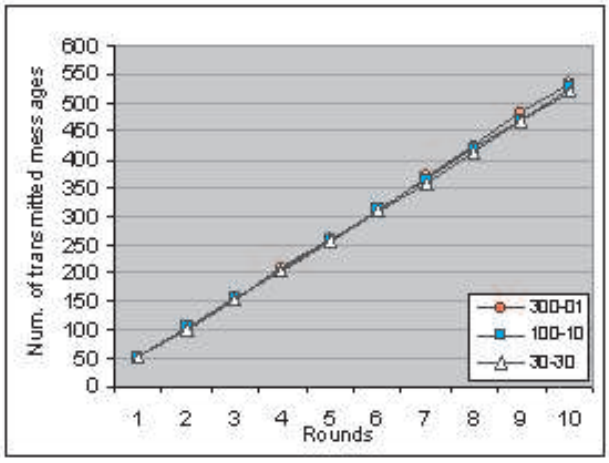

Figure 5.14: Post-incremental, query-incremental strategy DSDV num. of transmitted messages

network with the lower dynamic ratio. Finally, the number of transmitted messages is 1.29% more when the strategy is combined with the DSR protocol than when combined with the DSDV protocol. As the explanation of Figures 5.11 and 5.12, when the routing demand is very low, the DSR protocol has a very high packet delivery ratio and low routing overhead, which results in a similar number of transmitted messages among the three dynamic ratio scenarios. The DSDV protocol has a lower packet delivery ratio as the dynamic ratio of the network increases, which reduces the number of transmitted messages.

Figure 5.15 reviews the following. First, this strategy has the longest average waiting time of all the strategies. For a node in the network, it randomly chooses one node to post to or query from in the first round. As the number of rounds increases, the number of nodes which are randomly chosen also increases. These

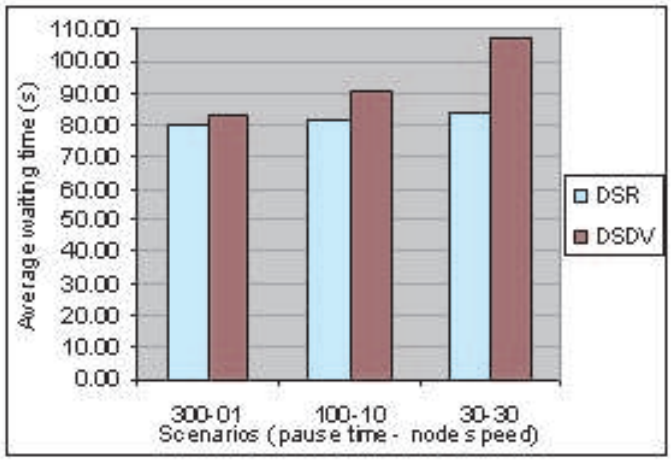

Figure 5.15: Post-incremental, query-incremental strategy DSR and DSDV average waiting time

nodes may be either within the same transmission range or multiple hops away from the node. This strategy causes a large number of clients to receive ServiceReply messages within a high number of rounds, which lengthens the average waiting time. Second, when this strategy is combined with the DSR protocol, the dynamic ratio of the network has a minimal effect on the average waiting time. A network with a higher dynamic ratio has a slightly longer average waiting time than the network with a lower dynamic ratio. As the number of rounds increases, the routing demands increase as well. For example, in the ninth round, the number of total postings and queryings is 450, and in the tenth round it increases to 500. With such routing demands, the DSR protocol in a high dynamic ratio scenario has higher routing overhead in those rounds than in a low dynamic ratio scenario. However, when the strategy is combined with the DSDV protocol, the

dynamic ratio has a more significant effect on the average waiting time. As the dynamic ratio increases, the average waiting time becomes longer. The discussion of Figures 5.11 and 5.12 reveals that the DSDV protocol has a lower packet delivery ratio as the dynamic ratio of the network increases, which results in more clients waiting for a long time to receive their ServiceReply messages. Finally, when this strategy is combined with the DSR protocol, it has a 12.94% shorter average waiting time than with the DSDV protocol.

5.2.4 Uniform memoryless: Post-to-l, query-l' strategy

In this strategy, all the servers in the network post their services to l nodes, and all the clients query l' nodes, where l and l' are less than or equal to N and are positive integers. This strategy uses a network unicast mechanism. It consists of rounds of uniform and memoryless repetitions of the $(l,\ l')$ post-query protocol. Thus, the number of transmitted messages is low and remains constant in each round. Hence, this strategy uses a relatively low amount of network bandwidth in each round if l and l' are relatively low. It has a high success rate and a long average waiting time.

The success rate of this strategy when it is combined with the DSR protocol and the DSDV protocol is illustrated in Figures 5.16 and 5.17. We extract two features of this strategy. First, although this strategy cannot achieve a 100% success rate, it can still obtain a success rate of 92.5% when it is combined with

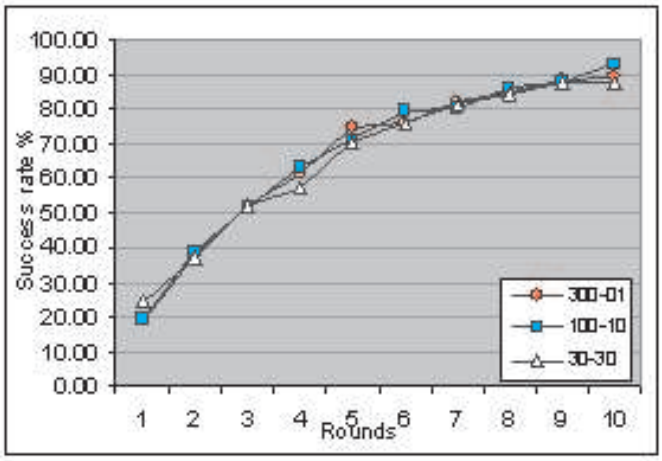

Figure 5.16: Post-to-l, query-l' DSR success rate

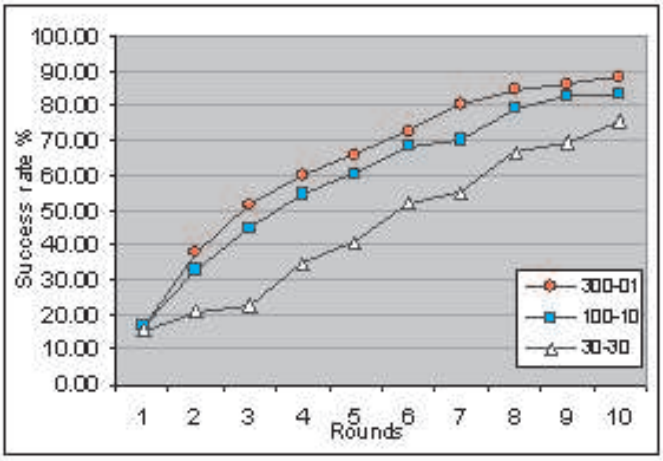

Figure 5.17: Post-to-l, query-l' DSDV success rate

the DSR protocol and of 88.25% with the DSDV protocol. The strategy requires that for each round, all the servers post their services to l nodes and all the clients query l' nodes. The size of the posting set l and the querying set l' influences the success rate. In this strategy we choose l and l' to be equal to five which is relatively low. A randomly chosen node from the previous round may be chosen again in the same round or in following rounds, so that after ten rounds of posting (querying), some nodes may not be posted to or queried from at all, which explains why the success rate may not reach 100%.

Second, we find that the dynamic ratio of the network has a minimal effect on the success rate when this strategy is combined with the DSR protocol. In all three scenarios, this strategy has a very similar success rate of each round, and they can all achieve high success rates of from 87.5% to 92.5% after ten rounds. However, it has a more significant effect on the success rate when combined with the DSDV protocol. For example, the strategy can achieve a success rate of 88.25% in the lowest dynamic ratio scenario (i.e. scenario 300-01), but it only obtains a success rate of 83.25% in scenario 100-10, and falls to 75.25% in the highest dynamic ratio scenario (i.e. scenario 30-30). The higher the dynamic ratio of the network is, the lower the success rate of this strategy. If l and l' are relatively low, the routing demands of the underlying routing protocols are relatively low. As was discussed in section 5.2.3., when the routing demands of the network are low, the DSR protocol performs well even in a high dynamic

ratio scenario. As for the DSDV protocol, the routing overhead has no significant impact with respect to the dynamic ratio described by Broch et. al [10]. However, it has a lower packet delivery ratio than the DSR protocol when the dynamic ratio of the network increases. The strategy can obtain a 7.42% higher success rate when it is combined with the DSR protocol than when combined with the DSDV protocol.

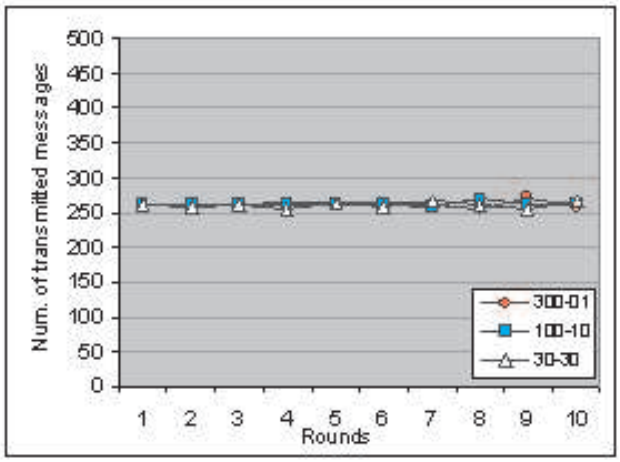

Figure 5.18: Post-to-l, query-l' DSR num. of transmitted messages

From Figures 5.18 and 5.19, we can note the following. First, this strategy consumes a relatively low amount of network bandwidth for each round in terms of the number of transmitted messages, when combined with the DSR protocol or DSDV protocol. It is designed to uniformly post and query services for each round, determining the same number of postings and queryings for each round. When l and l' are low, the number of ServiceReply messages are relatively low.

Second, we find that the dynamic ratio of the network scenarios has a minimal

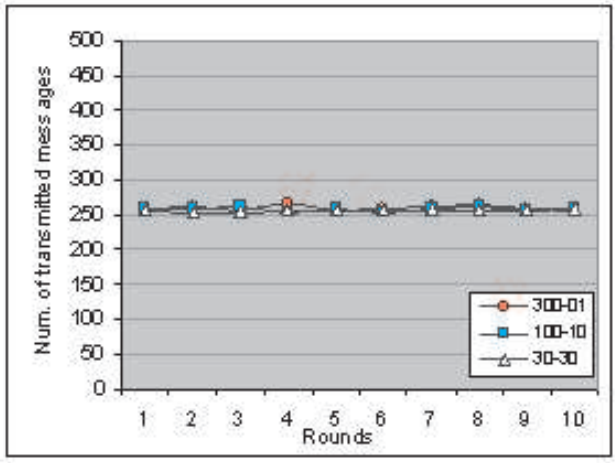

Figure 5.19: Post-to-l, query-l' DSDV num. of transmitted messages

effect on the number of transmitted messages when this strategy is combined with the DSR protocol. In each of the three network scenarios this strategy has almost the same number of transmitted messages in each round. When this strategy is combined with the DSDV protocol, the dynamic ratio has a minimal effect on the number of transmitted messages. During these ten rounds, the strategy has 20 more total transmitted messages in the lowest dynamic ratio scenario 300-01 than in scenario 100-10, and it has 22 more total transmitted messages in scenario 100-10 than in the highest dynamic ratio scenario 30-30. The number of transmitted messages is 1.11% more when combined with the DSR protocol than with the DSDV protocol. As we explained for Figures 5.16 and 5.17, the DSR protocol has a very high packet delivery ratio in all of the three dynamic ratio network scenarios, while the DSDV protocol has a lower packet delivery ratio in this strategy when the dynamic ratio of network scenarios increases.

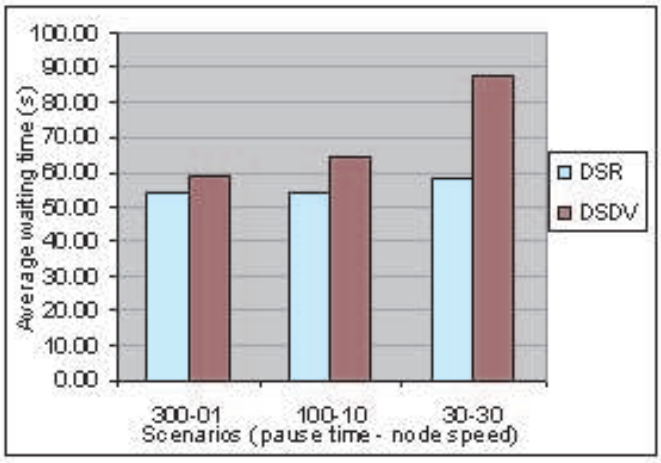

Figure 5.20: Post-to-l, query-l' DSR and DSDV average waiting time

The average waiting times for the DSR protocol and the DSDV protocol are shown in Figure 5.20. First, this strategy has a long average waiting time. A node may be randomly chosen to be posted to or queried from multiple hops away, so the average waiting time to receive the ServiceReply messages may be very long. Second, when this strategy is combined with the DSR protocol, the average waiting times of the three dynamic ratios are similar, which indicates a minimal performance difference with respect to the dynamic ratio of the network. However, when this strategy is combined with the DSDV protocol, as the dynamic ratio increases, the average waiting time becomes longer. Finally, when this strategy is combined with the DSR protocol, it has a 21.24% shorter average waiting time than with the DSDV protocol. The explanation for Figures 5.16 and 5.17 also can illuminate this feature. When the underlying routing protocol is the DSR protocol, this strategy has nearly identical success rates in all three dynamic

ratio scenarios, because the DSR protocol has a very high packet delivery ratio in these three dynamic ratio network scenarios. The DSDV protocol, however, has a lower packet delivery ratio than the DSR protocol in this strategy when the dynamic ratio of network scenarios increases. Under these circumstances, the strategy combined with the DSDV protocol has more unsuccessful clients, which lengthens the average waiting time.

5.2.5 With memory: Post-to-l, query-l' with memory strategy

This strategy is a modified Post-to-l, query-l' strategy which aims to improve the success rate by storing the identifiers of the posted or queried nodes. In this strategy, using network unicast communication, all the servers in the network post their services to l nodes, and all the clients query l' nodes, where l and l' are less than or equal to N and are positive integers. All the nodes memorize the identifiers of the nodes they posted to or queried from in previous rounds. Each round only involves nodes which have not been visited previously. When l and l' are chosen such that l times r is greater than or equal to N, and l' times r is greater than or equal to N, meaning that after r rounds, all the servers have posted their services to all the nodes and all the clients have queried each node in the network at least once, this strategy can achieve a very high success rate of up to 100% when is combined with the DSR protocol, and at least 97.75%

when combined with the DSDV protocol. Each node is required to have a cache which stores all the identifiers of posted or queried nodes. It has a long average waiting time and a low number of transmitted messages. Compared with the uniform memoryless Post-to-l, query-l' strategy, the success rate of this strategy is improved by 10.34% with the DSR protocol and by 16.5% with the DSDV protocol. The number of transmitted messages is 9% more when it is combined with the DSR protocol and 6.6% more with the DSDV protocol. The average waiting time of this strategy is 11.6% shorter with the DSR protocol and 12.4% shorter with the DSDV protocol.

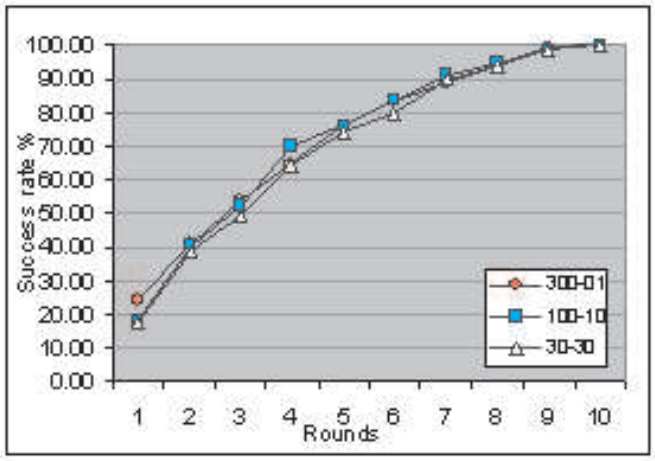

Figure 5.21: Post-to-l, query-l' with visited nodes DSR success rate

Figures 5.21 and 5.22 highlight the following characteristics of this strategy: First, we observe that this strategy can achieve a very high success rate of 100% when it is combined with the DSR protocol, and a high success rate of 97.75% with the DSDV protocol. In this strategy, we choose the same value for l, l' as

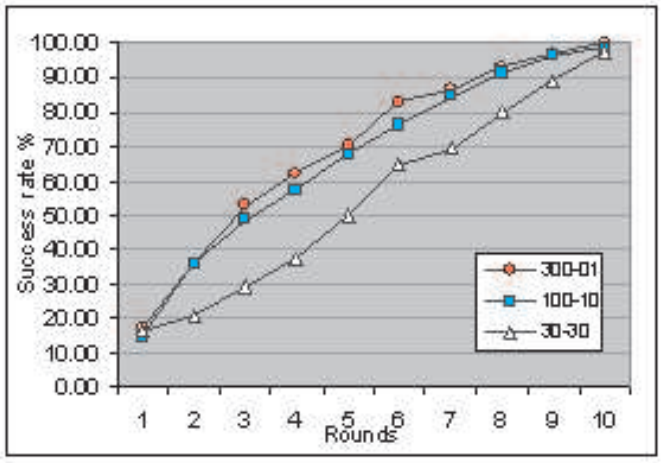

Figure 5.22: Post-to-l, query-l' with visited nodes DSDV success rate

those chosen in Section 5.2.4 for fair comparison purposes. Let l and l' be equal to five, such that l times r is equal to N and l' times r is equal to N, where r is equal to ten. A node in this strategy memorizes the identifiers of the nodes it has posted to or queried from in previous rounds. Only nodes which have not yet been visited are involved in each round. This arrangement allows all nodes in the network to have been posted and queried just once after ten rounds, which forces a node to locate the service it wants by traversing all the nodes of the network to achieve a high success rate.

Second, we find that when combined with the DSR protocol, the dynamic ratio of the network has little effect on the success rate. This strategy has a very similar success rate over each round, and it can achieve a very high success rate of 100% in all of the three scenarios. However, the dynamic ratio has a minor effect on this when combined with the DSDV protocol. The strategy can achieve

a very high success rate of 100% in the lowest dynamic ratio scenario 300-01, a success rate of 98.5% in scenario 100-10 and a success rate of 97.75% in the highest dynamic ratio scenario 30-30. When this strategy is combined with the DSR protocol, it has a 1.25% higher success rate than the one with the DSDV protocol. As in the explanations of Figures 5.16 and 5.17, the DSDV protocol has a lower packet ratio than the DSR protocol when the dynamic ratio of the network scenario increases, therefore, the success rate decreases as the dynamic ratio of the network increases. Compared with the uniform memoryless Post-to-l, query-l' strategy, the success rate of this strategy is improved by 10.34% when it is combined with the DSR protocol and by 16.5% with the DSDV protocol.

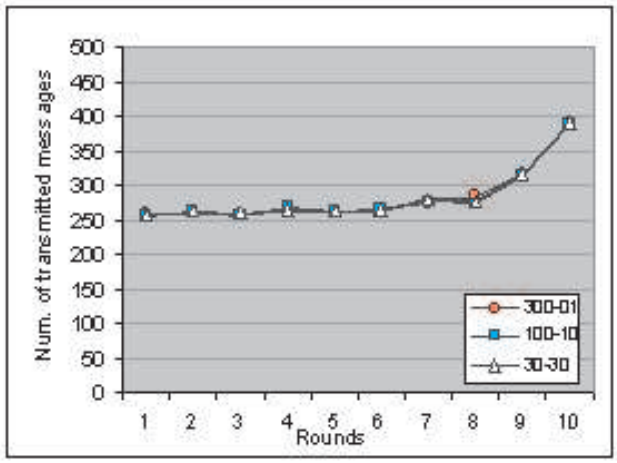

Figure 5.23: Post-to-l, query-l' with visited nodes DSR num. of transmitted messages

Figures 5.23 and 5.24 show several features of this strategy. First, this strategy has a low number of transmitted messages when combined with the DSR

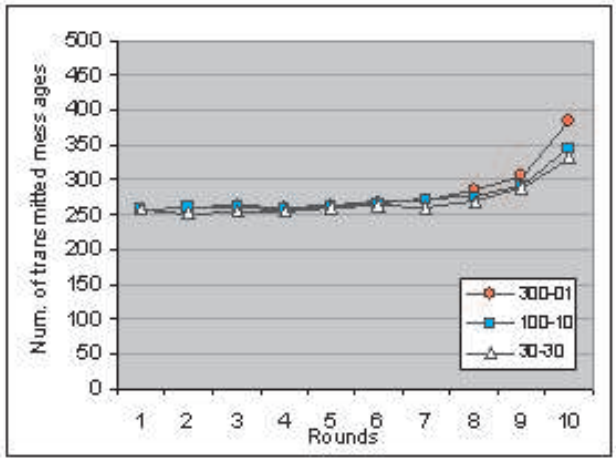

Figure 5.24: Post-to-l, query-l' with visited nodes DSDV num. of transmitted messages

protocol or DSDV protocol. It is designed to uniformly post to and query from a constant number of nodes in each round, which determines the identical number of postings and queryings for each round. As the number of rounds increases, more and more nodes have been posted to or queried from without repetition, resulting in more ServiceReply messages in the network. Second, we find that the dynamic ratio of the network scenario has little effect on the number of transmitted messages when this strategy is combined with the DSR protocol. This strategy has almost the same number of transmitted messages in each round in all three dynamic ratio network scenarios. When it is combined with the DSDV protocol, the dynamic ratio has a minimal effect on the number of transmitted messages. Within ten rounds, the strategy has 75 more total transmitted messages in the lowest dynamic ratio scenario 300-01 than in scenario 100-10, and

it has 56 more total transmitted messages in scenario 100-10 than in the highest dynamic ratio scenario 30-30. The number of transmitted messages is 3.24% more when the strategy is combined with the DSR protocol than with the DSDV protocol. From the discussion of Figures 5.18 and 5.19, we note that the DSR protocol has a very high packet delivery ratio in all three dynamic ratio network scenarios, and the DSDV protocol has a lower packet delivery ratio than with the DSR protocol in this strategy when the dynamic ratio of network scenarios increases. Compared with the uniform memoryless Post-to-l, query-l' strategy, the total number of transmitted messages in this strategy is 9% more when it is combined with the DSR protocol and 6.6% more with the DSDV protocol.

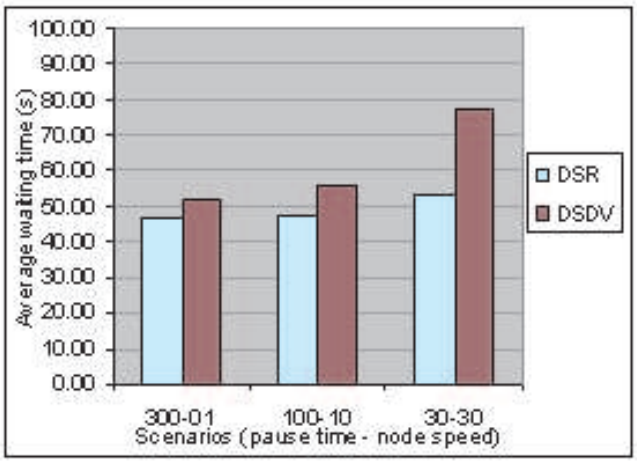

Figure 5.25: Post-to-l, query-l' with visited nodes DSR and DSDV average waiting time

The average waiting time for this strategy when it is combined with the DSR protocol and the DSDV protocol is illustrated in Figure 5.25. We can summarize

several characteristics from this figure. First, this strategy has a long average waiting time. A node may be randomly chosen to be posted to or queried from multiple hops away, so the waiting time to receive the ServiceReply messages may be very long. Second, when this strategy is combined with the DSR protocol, the average waiting times of these three dynamic ratio scenarios are similar. With the DSDV protocol, the higher the dynamic ratio of the network, the longer the average waiting time for this strategy. Third, we also notice that this strategy has a 19.04% shorter average waiting time when combined with the DSR protocol than with the DSDV protocol. We can refer to the discussion of Figure 5.20 to give the reasons for the first two features of this strategy. Compared with the uniform memoryless Post-to-l, query-l' strategy, the average waiting time for this strategy is 11.6% shorter with the DSR protocol and 12.4% shorter with the DSDV protocol. As was mentioned for Figures 5.21 and 5.22, this strategy is able to achieve a high success rate. This also implies that during these ten rounds, almost all the clients find their services. In the uniform memoryless Post-to-l, query-l' strategy, some clients are not posted to or queried from, so these clients may wait a longer time.

5.3 Summary

In Section 5.2 we reviewed in detail the three performance metrics of the greedy, conservative, incremental, uniform memoryless and with memory strategies when

they are combined with the DSR protocol and DSDV protocol. Table 5.1 gives a complete view of the performance of these strategies when combined with these protocols. For each strategy, we list the maximum success rate, average waiting time and total number of transmitted messages.

Each of the metrics in Table 5.1 were calculated with a 90% confidence interval. Table 5.1 can be summarized as follows based on the parameters we chose. The greedy strategy consumes significant network resources, in terms of the number of transmitted messages, to achieve a high success rate of 100%. It has the highest number of transmitted messages and the lowest average waiting time of all the strategies.

The conservative strategy largely reduces the number of transmitted messages. This strategy can also achieve a high success rate of 100% in a high dynamic ratio ad hoc network. For ad hoc networks with a lower dynamic ratio, the success rate is low (e.g. 81.5%), because numerous nodes may not be posted to or queried from at all. This strategy has a low number of transmitted messages and a short average waiting time.

The incremental, uniform memoryless and with memory strategies use a network unicast mechanism. The size of the posting (querying) set affects the three performance metrics to a considerable degree. According to our chosen parameters, we can summarize as follow. The incremental strategy has a low number of transmitted messages. It can also achieve a high success rate of 96%, while the

Strategy	**Routing protocol**	**Dynamic ratio**	**Max. success rate**	**Total num. of transmitted msgs.**	**Average waiting time(s)**
Greedy	DSR	8.06	100%	16215	7.50
		61.48	100%	15601	7.85
		190.72	98.75%	13100	10.42
	DSDV	8.06	100%	16557	4.93
		61.48	100%	15777	5.03
		190.72	100%	14887	5.46
Conservative	DSR	8.06	82.25%	2909	36.55
		61.48	99.75%	5455	20.12
		190.72	100%	3620	8.42
	DSDV	8.06	81.5%	2122	37.62
		61.48	99.5%	4519	27.81
		190.72	100%	3408	10.09
Incremental	DSR	8.06	96%	2900	79.72
		61.48	94.75%	2894	81.32
		190.72	96%	2899	83.76
	DSDV	8.06	97%	2898	83.23
		61.48	87.75%	2856	90.80
		190.72	81.75%	2828	107.15
Uniform memoryless	DSR	8.06	89%	2602	54.55
		61.48	92.5%	2610	54.30
		190.72	87.5%	2602	57.72
	DSDV	8.06	88.25%	2596	59.27
		61.48	83.25%	2577	64.57
		190.72	75.25%	2555	87.64
With memory	DSR	8.06	100%	2845	46.96
		61.48	100%	2833	47.29
		190.72	100%	2832	53.05
	DSDV	8.06	100%	2816	51.89
		61.48	98.5%	2741	56.12
		190.72	97.75%	2686	77.27

Table 5.1: Performance comparison of the five Post-query strategies when combined with the DSR protocol and DSDV protocol

tradeoff is that it has the longest average waiting time of all the strategies.

The uniform memoryless strategy consists of rounds of uniform and memoryless repetitions of the $(l,\ l')$ post-query protocol. The number of transmitted messages is low and remains constant in each round. Hence, this strategy uses a relatively low amount of network bandwidth in each round, if l and l' are relatively low. In our design, both of them are set to five. It has a long average waiting time.

The *with memory* strategy can achieve a very high success rate of up to 100%. Each node is required to have a cache which stores all the identifiers of posted or queried nodes. It has a long average waiting time and a low number of transmitted messages. This strategy has the same l and l' as the uniform memoryless strategy. Compared with the uniform memoryless Post-to-l, query-l' strategy, the success rate of this strategy is improved by 10.34% with the DSR protocol and by 16.5% with the DSDV protocol. The number of transmitted messages is 9% more when it is combined with the DSR protocol and 6.6% more with the DSDV protocol. The average waiting time of this strategy is 11.6% shorter with the DSR protocol and 12.4% shorter with the DSDV protocol.

Among these five strategies, only the greedy strategy has a higher success rate, a lower average waiting time and a higher number of transmitted messages when combined with the DSDV protocol than with the DSR protocol. The other four strategies have a lower success rate, a longer average waiting time and a

lower number of transmitted messages when combined with the DSDV protocol than with the DSR protocol. Among these five strategies, only the conservative strategy has a higher success rate, a shorter average waiting time and a higher number of transmitted messages in a high dynamic ratio network scenario. The other four strategies have a lower success rate, a higher average waiting time and a lower number of transmitted messages than in a high dynamic ratio network scenario.

In Table 5.1, we have five Post-query strategies, each of which can be combined with two types of ad hoc routing protocols. Each strategy, combined with a routing protocol, has three different dynamic ratio network scenarios. Thus, we have 30 combinations, so when we want to adopt a Post-query strategy to an ad hoc network, this table can serve as a reference. Before we select a strategy, some analysis should be conducted on the size of the ad hoc network, the available ad hoc routing protocols, the dynamic ratio, the available support for broadcast and unicast communication, the amount of available bandwidth and the specific requirements for the three performance metrics. For an ad hoc network that supports broadcast communication which requires a high success rate and a low average waiting time, regardless of the number of transmitted messages, the greedy strategy combined with the DSDV protocol are a good choice. It can be adopted in the different dynamic ratio network scenarios. For an ad hoc network that supports broadcast communication which requires a high success rate and a

low number of transmitted messages, regardless of the average waiting time, the *with memory* combined with the DSR protocol are the most suitable one. When we have an ad hoc network with a high dynamic ratio, the conservative strategy combined with the DSR protocol or with the DSDV protocol are the best two choices to achieve a high success rate with a low number of transmitted messages and a low average waiting time. For an ad hoc network which can only spare a dedicated bandwidth for a service discovery strategy, the uniform memoryless strategy can meet this requirement when both l and l' are relatively low, though the corresponding success rate is not high.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

With the increasing use of ad hoc networks, mobile nodes need to discover the available services in a network quickly and correctly. Therefore, research on service discovery over ad hoc networks is attracting attention. An efficient service discovery mechanism should balance its performance against its cost in the context of ad hoc networks. In this thesis, five types of service discovery strategies, namely the greedy, conservative, incremental, uniform memoryless and with memory Post-query strategies combined with the DSR protocol and DSDV protocol, were investigated for their performance in three different dynamic ratio ad hoc networks. In addition, efforts have been made on multiple fronts related to the design, implementation, and simulation of these five strategies. The work of

this thesis can be summarized as follows:

1. Proposal of the conservative Post-query strategy;

2. Design and implementation within the NS-2 environment of five Post-query strategies: the greedy, conservative, incremental, uniform memoryless and with memory Post-query strategies combined with the DSR protocol and DSDV protocol;

3. Design of a simulation model which covers the following aspects: the selection and construction of network topologies, the choice of the parameters used in the simulation, and the development of simulation scenarios for different dynamic ratios;

4. Performance evaluation of these five strategies combined with the DSR protocol and DSDV protocol.

The evaluation revealed the performance of these five strategies combined with the DSR protocol and DSDV protocol. A summary of the performance evaluation and design suggestions for different ad hoc networks is given in Section 5.3.

6.2 Future work

In this thesis we gave the design and implementation of the five Post-query strategies in Chapter 3. We carried out simulations of all these five strategies combined

with the DSR protocol and DSDV protocol in Chapter 4, and we evaluated the performance of these five strategies in Chapter 5. In Section 5.3, we concluded our evaluation results together with design suggestions for different ad hoc networks. Several interesting problems remain for further investigation.

1. For the incremental, uniform memoryless and with memory Post-query strategies which use unicast network communication, the sizes of the posting and querying sets greatly influence the performance. For these three strategies, we could dig deeper to evaluate how the performance is influenced if we vary the sizes of posting and querying sets.

2. In order to maintain equivalent comparisons, our performance evaluation of these five strategies adopted the same round interval and the same execution interval for posting and querying. However, some Post-query strategies such as the greedy and conservative strategies, may perform differently if the round interval and the execution interval are variable.

3. We assumed that all the services of the network follow a uniform distribution, such that each kind of service is requested with equal probability. It would be very interesting to study networks in which the services have different probabilities. For example, in some ad hoc networks more clients may request a web browser service than request a scanner service.

4. We chose the DSR protocol and DSDV protocol as our typical ad hoc

routing protocols. In the future, we may combine our Post-query strategies with other kinds of ad hoc routing protocols such as AODV [Per99], TORA [Par97] [Par98], OLSR [Jac01] and ZRP [Haa97].

5. We could extend our research according to the heuristic model proposed by Koodli and Perkins [Koo02]. Their approach is to add extensions to suitable ad hoc network routing protocols, in order to find the services and routes to these services at the same time.

Appendix A

Message Sequence Chart 96

Standard

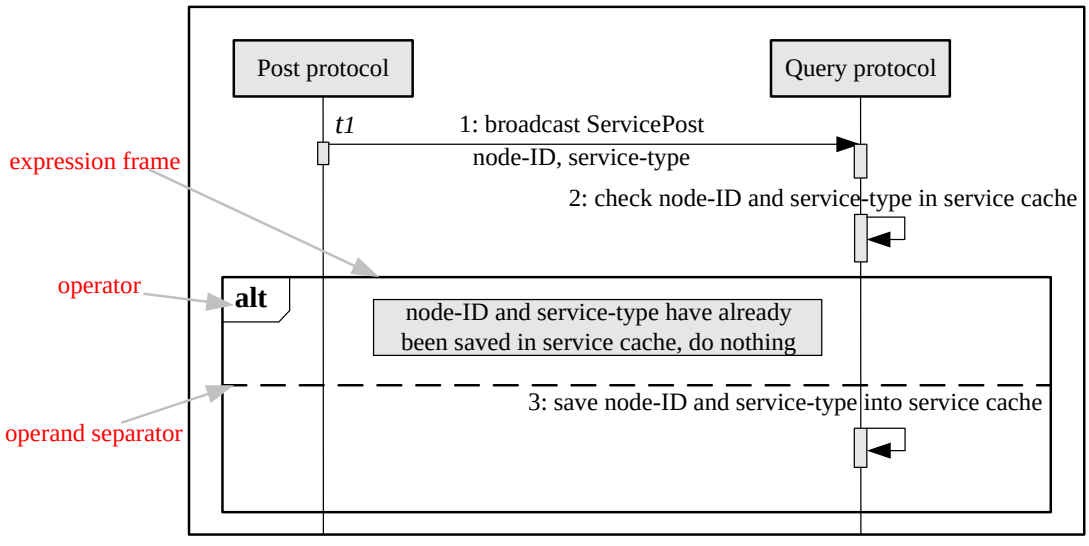

Figure A.1: Message sequence chart 96 example

Figure A.1 illustrates some standard notations of MSC96. The expressions

are enclosed by an expression frame. The operands are separated by a dashed separation line, and the operator is depicted in the left upper corner of the expression frame. The operator *alt* describes a point of decision where one of several alternative courses of action can be followed. However, the point of decision is deferred to the point where the alternatives differ.

Bibliography

[Bar03] M. Barbeau, E. Kranakis, Modeling and Performance Analysis of Service Discovery Strategies in Ad Hoc Networks, Proceedings of International Conference on Wireless Networks (ICWN), Las Vegas, Nevada, 2003.

[Blu01] Bluetooth, Specification of the Bluetooth System, Specification Volume 1, 2001, Specification Volume 2, 2001. http://www.bluetooth.com

[Bro98] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98), pp. 85-97, Dallas, TX, October 1998.

[Che02] L. Cheng, Service Advertisement and Discovery in Mobile Ad Hoc Networks, Workshop on Ad Hoc Communications and Collaboration

in Ubiquitous Computing Environments, New Orleans, Louisiana, USA, November, 2002.

[Fal97] K. Fall, K. Varadhan, editors. *ns* notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. http://www.isi.edu/nsnam/ns/.

[Gut99] E. Guttman, C. Perkins, J. Veizades, M. Day, Service Location Protocol, Version 2, IETF RFC 2608, June 1999.

[Haa97] Z. Haas, M. Pearlman, The Zone Routing Protocol (ZRP) for Ad hoc Networks, IETF Draft - Mobile Ad hoc NETworking (MANET) Working Group of the IETF, November 1997.

[Hel03] S. Helal, N. Desai, V. Verma, C. Lee, Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks, Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.

[Her01] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, A. Schade, DEAPspace - Transient Ad-Hoc Networking of Pervasive Devices, Computer Networks, vol. 35, pp. 411-428, 2001.

[Jac01] P. Jacquet, P. Muhlethaler, A. Qayyum, Optimized Link State Routing Protocol, IETF Draft, August 2001.

[Joh96] D. Johnson, D. Maltz, Dynamic Source Routing in Ad-Hoc Wireless Networks, in Mobile Computing, 1996

[Jub87] J. Jubin, J. Tornow, The DARPA Packet Radio Network Protocols. Proceedings of IEEE, 75(1), pp. 21-32, January 1987.

[Koo02] R. Koodli, C. Perkins, Service Discovery in On-Demand Ad Hoc Networks, Internet-Draft, October 2002.

[Kra92] E. Kranakis, P. Vitányi, A Note on Weighted Distributed Match-Making, in Mathematical Systems Theory, Vol. 25, 123-140, 1992.

[Lee99] S. Lee, M. Gerla, C. Toh, On-demand multicast routing protocol (ODMRP) for ad hoc networks, Internet Draft, work in progress, June 1999.

[Mal99] D. Maltz, The Effects of On-Demand Behavior in Routing Protocols for Ad Hoc Networks, IEEE Journal on Selected Areas in Communications Special Issue on Mobile and Wireless Networks, pp. 1439-1453, August 1999.

[Msc99] Tutorial on MSC-96, TIMe Electronic Textbook v 4.0, http://www.item.ntnu.no/fag/SIE5020/msc/msc96.pdf, 1999.

[Mul88] S. Mullender, P.Vitányi, Distributed Match-Making, Algorithmica, Vol. 3, pp. 367-391, 1988.

[Nid01] M. Nidd, Service Discovery in DEAPspace, IEEE Personal Communications, August 2001.

[Par97] V. Park, M. Corson, A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. Proceedings of the IEEE Conference on Computer Communications, pp. 1405-1413, Kobe, Japan, April 1997.

[Par98] V. Park, M. Corson, A Performance Comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-State Routing. Proceedings of IEEE Symposium on Computers and Communication'98, pp. 592-598, Athens, Greece, June 1998.

[Per94] C. Perkins, P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, Proceedings of the ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, London, UK, pp. 234-244, August 1994.

[Per99] C. Perkins, E. Royer, Ad-hoc On-Demand Distance Vector Routing, Proceedings of Second IEEE Workshop. Mobile Computing System and Applications, February 1999.

[Sun99] Sun Microsystems, Technical White Paper: Jini Architectural Overview. http://www.sun.com/jini/, December 1999.

[Tuc93] B. Tuch, Development of WaveLAN, an ISM Band Wireless LAN. AT&T Technical Journal, 72(4), pp. 27-33, July/August 1993.