Overlay service architecture for Mobile ad-hoc networks

Anandha Gopalan and Taieb Znati
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, U.S.A
Email: {axgopala, znati}@cs.pitt.edu

***Abstract*—**
Ad-hoc networks are an emerging technology with enormous potential. However, the lack of support for large-scale deployment of applications has hindered the adoption of these networks for commercial use. This paper addresses this shortcoming and details an architecture that supports large-scale deployment of services in ad-hoc networks.

I. Introduction

A new class of wireless networks called wireless ad-hoc networks (or ad-hoc networks [1]) are emerging as a viable alternative to cellular wireless networks. Ad-hoc networks are a collection of collaborative nodes that can communicate among themselves without the help of any existing infrastructure. Due to the lack of an existing infrastructure, ad-hoc networks are easily deployed and can be configured to suit a specific purpose.

However, despite the advantages provided by ad-hoc networks, these networks do not find much use outside of military and crisis management. Deployment of these networks on a large-scale has been hampered by the the lack of support for providing services. Solving the problem of the lack of a service architecture will have a major impact by allowing for the deployment of large scale applications in ad-hoc networks. This will take ad-hoc networks from primarily a military type of network into a wide-spread commercially accepted network.

To deploy services and enable service discovery in an ad-hoc network, an infrastructure is needed. We need to develop a framework along with a set of protocols and applications that can help in service deployment in ad-hoc networks. The following research challenges must be addressed while designing this framework:

Question 1: How do we discover a service provider ?

The primary question to be addressed while designing any service discovery protocol is the location of services that are available in the network. Service discovery in traditional infrastructured network occurs by querying a central server that contains information about the services available in the network or by querying a set of nodes in the network that act as a distributed server containing the required information [2], [10].

This problem is especially challenging in ad-hoc networks due to the fact that the mobility of the nodes in an ad-hoc network can cause a change in the topology of the network. Its imperative for any service location protocol designed for ad-hoc networks to take into account the effect of mobility on the network. Due to the lack of an infrastructure, there is a necessity for the protocol to be a server-portfolio driven protocol, whereby a server advertises not only the services that it offers, but also the interface for the services that it provides along with an interface for its mobility profile the way it sees it (this is the portfolio of the server). This mobility profile will be very useful to the client while it tries to contact the server to request a service from the server.

Question 2: Once a service is located, how do we route traffic to a mobile server ?

[1] In this document, Wireless ad-hoc networks and ad-hoc networks will be used interchangeably

Routing in an ad-hoc network is quite different in nature to routing in infrastructured networks, as there are no base stations and routers in ad-hoc networks [4], [7], [8], [3], [1]. The nodes in the network collude to help each other route traffic through the network. As mentioned earlier, mobility of the nodes changes the topology of the network necessitating a route change each time a node moves to a different location in the network. Once a service provider is located, the client must be able to contact the service provider to avail of the service.

The routing protocol most suitable for such a framework would be a geographically-based routing protocol [5], since the knowledge of the location of the server is available to the client through the server-portfolio that is available to the client when the client queries for information about the server. Due to the mobility of the server, the client must be able to use the server-portfolio to *predict* the current location of the server and use this information to route to the server. This is the reason why the protocol is called a *predictive geographically-based* routing protocol.

Question 3: How do we make the framework robust and scalable ?

Scalability and robustness are two very important characteristics of any protocol designed to provide a service structure for service location and discovery. Scalability is needed to make sure that the network grows gracefully as nodes join the network and the network size increases. The network must be robust to make sure that a consistent view of the network is maintained even after a failure by making sure that the after effects of the failure are mitigated as soon as possible.

Achieving scalability in ad-hoc networks is a challenging problem, as the nodes in the network generally do not have the whole view of the network, due to the changes that can occur in the network by the mobility of the nodes. The protocol must make sure that the new information generated by a node joining the network is percolated to the other nodes in the network in an expeditious manner. Robustness can be achieved in the network by using redundancy, so that if some parts of the network fail, the information is not lost and can be recovered by using other nodes in the network.

The objective of this research is to provide a secure, robust and efficient framework along with the protocols and algorithms to allow for large scale service deployment in ad-hoc networks. The research takes a peer-to-peer [6] based unique approach to enable large scale deployment of network services over ad-hoc networks.

This *vision* of having a service architecture for ad-hoc networks is further enhanced by the emerging trends in wireless networks: better power handling capabilities in the mobile nodes, newer version of 802.11 that allows nodes to communicate over longer distances and advances in embedded systems with MEMS (Micro-Electro-Mechanical Systems), ASIC (Application Specific Integrated Circuit) and nanotechnology.

The rest of the document is organized as follows: Section II concludes the document by providing an overview of the different components of the protocol and the algorithms used.

II. General overview of OSAM operation

A. Virtual Home

The entire network is partitioned into different virtual zones (cells) as shown in figure 1 (similar to a cellular phone network). Each one of these cells is associated with a virtual home. A virtual home is a physical location that consists of nodes in the network (a virtual home may sometimes contain no nodes depending on the node density in the network). Every node in the network is associated with a *virtual home* (VH), which is the *most likely home* where it resides. The virtual home can be considered to be an anchor for the node. A node leaves behind its *travel signature* (which consists of the expected direction of travel and the expected speed of the node) upon leaving its virtual home. The node recruits other members in its virtual home to hold this information on its behalf. The size of the virtual cells is very important to the correct functioning of the protocols. The important research questions that need to be addressed are:

Question 4: What are the tradeoffs in deciding the size of the virtual cell

The size of the virtual cell is a very important issue while dividing the network into virtual cells. A large virtual cell results in lesser number of virtual cells existing in the network and hence management of these virtual cells is easier. The disadvantage to having large virtual cells is that tracking down a node inside a cell is harder. A small virtual cell helps in tracking down a node inside a cell, but the disadvantage of this scheme is that this results in a large number of virtual cells and maintaining information about all these virtual cells would result in a large overhead.

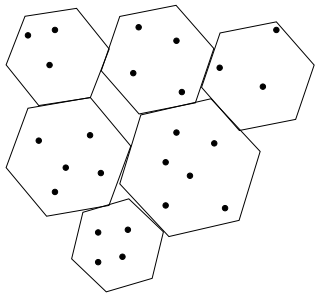

Fig. 1. Partitioning the network into zones

Question 5: How does the size of a virtual cell reflect on the management of the cell

Virtual cell management is responsible for maintaining information about the virtual cell and also about the neighboring virtual cells. A large virtual cell would result in a larger number of nodes being part of the virtual cell and would thus result in a very high overhead in maintaining the information about the nodes in the virtual cell. However, with large virtual cells, the number of virtual cells in the network would be small and this helps in reducing the overhead in maintaining the information about the virtual cell's neighbors. A small virtual cell results in lesser number of nodes being part of that virtual cell and hence maintaining information about these nodes does not require a very high overhead. The flip-side to having small virtual cells is that this results in a lot of virtual cells in the network, and this leads to a large overhead while maintaining information about the neighboring virtual cells.

The above discussion provides us with a broad overview of the importance of the size of a virtual cell. Several methods have been suggested to determine the size: determine the size based on the terrain, use information about the number of nodes in that area. A good determination of the size of a virtual cell could be using the zip code information.

B. Service Location

When a node requires a service, it contacts a name service, the name service returns the list of nodes that offer this service along with the virtual home where these nodes reside. The node can then decide on the best node that is offering the service (this decision can be made based on past history, distance to the node and the stability of the node). Algorithm 1 and figure 2 detail the procedure by which a client node *C* obtains a service from a server node *S* that is mobile.

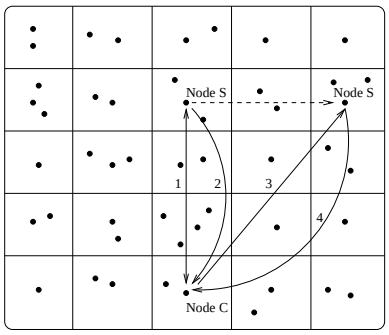

Fig. 2. A mobile server

Algorithm 1: Handling server mobility
Input: Void
Output: Result
SERVER-MOBILITY()
(1) C calculates VH(S) based on the node id of S
(2) Using directional routing, C send a message to VH(S) intended for S (msg 1 in the figure)
(3) The nodes in the VH(S) reply with the $travel\ signature$ of S, which is a metric $[t_0, V(t_0), D(t_0), P_V(t), P_D(t)]$ (msg 2 in the figure)
(4) C uses the metric to calculate the new position of S and sends a message to S (msg 3 in the figure)
(5) S upon receiving the message by C acknowledges it (msg 4 in the figure)
(6) **break**

The components of the metric are:

- t_0: starting time
- $V(t_0)$: expected average starting speed.
- $D(t_0)$: expected average initial direction.
- $P_v(t)$: Predictor for speed after t units of time since departure.
- $P_d(t)$: Predictor for direction after t units of time since departure.

In order to be able to support the operations detailed in the figure and the algorithm, the following capabilities must be available: Virtual home registration and discovery, virtual home and current location management, and tunneling and traffic routing. Sections II-C - II-E talk about these capabilities in greater detail.

C. Virtual home registration and discovery

Every node in the network is associated with a virtual home, as shown in figure 2. The virtual home is a location in space where the node resides. The nodes in the network need to be registered with a service similar to a name service in order for them to be discovered.

The name servers are nodes in the network with the extra functionality of acting as databases holding information about the nodes in the network. The information that is stored at the name servers consist of the node id, services that this node offers and the virtual home of the node (for routing purposes). The information is stored in a distributed manner where each name server is responsible for some part of the name space.

When a node needs to register a service, it calculates a hash from the value of its node id (based on a hash function that is known to all nodes in the network); based upon the hash value, the node registers with the appropriate name server. The novelty of this scheme is that the name server could also be mobile, hence the node that needs to register finds out the virtual home of the name server (again, by hashing the id of the name server) and uses this information to register itself with the name server.

When a node A requires to discover the services available in the network, it queries one of the name servers to retrieve this information. Again, the nodes finds out the virtual home of the name server (since the name server could be mobile) and routes the information to this name server. The name server upon receipt of this query replies with the list of nodes that offer the service that node A was requesting. Node A can now choose from this list and can avail of the service directly from the node.

D. Virtual home and current location management

Location management is a very important part of this framework. Since, the nodes in the network have the ability to be mobile, this framework must be able to handle mobile nodes. As mentioned earlier, when nodes register themselves with the name servers in the network, they also register their virtual homes (the most likely home of the node). The virtual home of the node is very important for routing purposes, as messages are intended for a node are routed to the node's virtual home.

When a node A become mobile or leaves its virtual home, there has to be some way for other nodes to know the new location of node A. The earliest methods to handle mobility was using mobile IP, which used the concept of a home zone and a foreign zone. Each time a node changes its location, it has to inform its home zone. This leads to a lot of overhead and the number of messages in the network also grows. This framework does not require a node to keep updating its position to its virtual home, instead the node can leave behind its *travel signature*, which consists of some information with regards to the expected direction and speed of travel of node A. Using this information, nodes which wish to contact node A, can re-calculate the position based on the time elapsed since node A left its virtual home.

When a node A moves outside its virtual home, it needs to recruit other nodes in its virtual home as *care-off* nodes which are responsible for maintaining the *travel signature* of A. In the event that there exists no node in its virtual home, node A queries its neighboring virtual homes in an effort to recruit nodes to maintain its *travel signature*.

When a node A moves to a new location L_2 outside its virtual home $L_1$and decides to move again, it again leaves behind its new *travel signature* with some nodes in L_2. This is in case another node B was trying to locate A and sends a message to L_2 and cannot locate A, it can use the new travel signature of A to locate node A.

E. Tunneling and traffic routing

The nodes in an ad-hoc network can be mobile and this mobility needs to be incorporated into the protocol. When a node D leaves its virtual home, it leaves behind its *travel signature*, which consists of the metric $[t_0, V(t_0), D(t_0), P_V(t), P_D(t)]$ (as shown in algorithm 1). This *travel signature* is stored with some of the nodes in the virtual home that are recruited by node D. If there are no nodes in the virtual home, then node D, tries to recruit some nodes from its neighboring virtual homes to hold its *travel signature*.

When another node S tries to contact node D, and node D is not present in its virtual home, this *travel signature* is returned to node S. At the same time, this request is tunneled directly to node D, since the nodes in the virtual home already know the *travel signature* of node D, since they are responsible for maintaining this information.

Once node S receives the *travel signature* of node D, it tries to route the traffic to node D. To reduce flooding in the network, the traffic is sent in a cone-shaped fashion as shown in figure 3, (similar to [9], but here all the nodes need not know about the position of every other node in the network), any node in region 1 has the highest priority to forward the traffic, while a node in zone 2 has lower priority. If a node in zone 1 fails to forward the traffic or there is no node in zone 1, nodes in zone 2 forward the traffic towards D. Each node that receives this routing information re-calculates the position of D using its *travel signature* and thus re-calculates the cone based on this information before sending the messages.

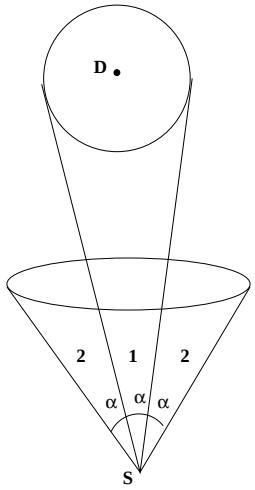

Fig. 3. Directional Routing

REFERENCES

[1] A. Gopalan, S. Dwivedi, T. Znati and A. B. McDonald. On the implementation of the (α, t)-Cluster Protocol on Linux. In *Proc. 37th Annual Simulation Symposium*, Apr. 2004.

[2] E. Guttmann and C. Perkins and J. Veizades and M. Day. Service Location Protocol, 1999.

[3] Z. J. Haas and M. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. *Internet Draft*, Aug. 1998.

[4] David B Johnson and David A Maltz. Dynamic Source Routing in Ad hoc Wireless Networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[5] Y. B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad-Hoc Networks. In *Proc. ACM/IEEE MOBICOM*, Oct. 1998.

[6] Dejan S. Milojicic, Vana Kalogeraki, Kiran Nataraja Rajan Lukose, Jim Pruyne, Bruno Richard, and Zhichen Xu Sami Rollins. Peer-to-Peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[7] C. Perkins and E. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proceedings 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, 1999.

[8] C. R. Perkins and P. Bhagwat. Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM*, pages 234–244, Oct. 1994.

[9] S. Basagni, I. Chlamtac, V. R. Syrotiuk and B. A. Woodward. A Distance Routing Effect Algirithm for Mobility (DREAM). In *Proc. ACM/IEEE Mobicom*, pages 76–84, October 1998.

[10] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu. Simple Service Discovery Protocol/1.0. *Internet Draft*, Oct. 1999.