



Prototype Implementation of Anycast-based Service Discovery for Mobile Ad Hoc Networks

Jidong Wu, Oliver Stanze, Kilian Weniger und Martina Zitterbart

Institute of Telematics, University of Karlsruhe
Zirkel 2, Karlsruhe, D-76128, Germany
{wulstanzelwenigerlzit}@tm.uka.de

Abstract: Mobile ad hoc networks are self-organized, and devices in such networks have to locate available services dynamically. We propose to utilize anycast for efficient service discovery. With anycast, requests of service sent by clients can be delivered to the closest service provider. The Ad Hoc On-demand Distance Vector (AODV) routing protocol is extended to support anycast routing. We present a prototype implementation which demonstrates the application of anycast-based service discovery.



1 Introduction




Mobile ad hoc networks (MANETs) are composed by mobile devices. They are self-organized wireless multihop networks which can operate without the support of a fixed infrastructure. Mobile ad hoc networks may find their applications, for example, at big conferences or exhibitions, and in the communication among cars.

An important problem in mobile ad hoc networks is how mobile devices locate available services in the network. For example, services may be printing services provided by network printers, naming services provided by name servers, or interconnectivity services provided by Internet gateways. Furthermore, in order to improve service resilience, one service may be provided by two or more providers in the network.

In MANETs, services or providers of services are hardly known a priori. Services may be dynamically created and maintained due to the property of self-organization of MANETs. The availability of services may change frequently because of devices joining and leaving the network, or because of time-limited provision of services. As a result, rather than pre-configuration, devices in MANETs have to use dynamic *service discovery* mechanisms for locating available services.

Well-known service discovery mechanisms developed for fixed networks are not suited for MANETs, because they do not take into account the unique properties of a MANET like limited wireless bandwidth, frequent changes of network topology, multi-hop communication, and decentralized network control. Therefore, these existing mechanisms cannot be applied to MANETs in a straightforward way. Service discovery is, accordingly, an active research issue in the field of MANETs.

In this paper we propose to utilize anycast for efficient service discovery in MANETs. The concept of anycasting [PMM93], i.e., delivering datagrams to (the closest) one of





members of an anycast group, can be applied to service discovery in MANETs. Assuming well-known anycast addresses are assigned to common services, mobile devices could simply send service requests to those anycast addresses. The network is then responsible for delivering these requests to appropriate service providers.

Nevertheless, anycast routing is still not supported by well-known ad hoc routing protocols. In order to support anycast routing, we have made an extension to Ad Hoc On-demand Distance Vector (AODV) routing protocol. This extension has been implemented and used to build a demonstration of anycast-based service discovery.

The rest of the paper is organized as follows. We discuss anycast-based service discovery in MANETs in section 2, and describe our anycast extension to AODV in section 3. Section 4 describes the prototype implementation as well as the demonstration. Finally, we give a summary in section 5.

2 Anycast-based service discovery

Recently many mechanisms and technologies have been proposed for service discovery. These mechanisms allow users of portable or mobile devices to find services in a new network, and help new devices be easily integrated into an existing network. However, the networks in question are usually assumed to be wired networks, which are part of an existing network infrastructure, or to be small wireless networks in which all devices are located in the direct transmission range of each other.

In general, various approaches to service discovery can be classified into two categories: directory-based or non-directory-based approaches. In directory-based approaches, there are repositories which store the information of available services. These repositories provide a directory of available services. Providers of service register the attributes of their services with the repositories, while clients of service inquire services by sending requests to the repositories. For instance, *lookup services* used in Jini [Sun01a, Sun01b] and *directory agents* used in IETF's SLP [GPVD99] are examples of such repositories.

In contrast, in non-directory-based approaches there do not exist any directories of services. Therefore, to find available services in a network, either providers of services periodically advertise their services, or clients inquire on-demand about services interested. Because periodical advertisements may cause waste of communication bandwidth, in most cases on-demand inquiry is preferred. Usually, clients can send their inquiries to all service providers by multicasting (if possible) or flooding. All providers of the service then answer to the clients with the information of their services. That is, clients have to contact with all providers of a service before they can select one from them.

However, there are some situations where clients want to contact anyone of all providers of a specific service. That is, clients do not differentiate between different providers of the same service. For such cases, anycast is a efficient means to contact the closest service provider.

For instance, clients of naming service only care whether a human-readable name is resolved to an IP address. They do actually not care which name server has made this resolution.



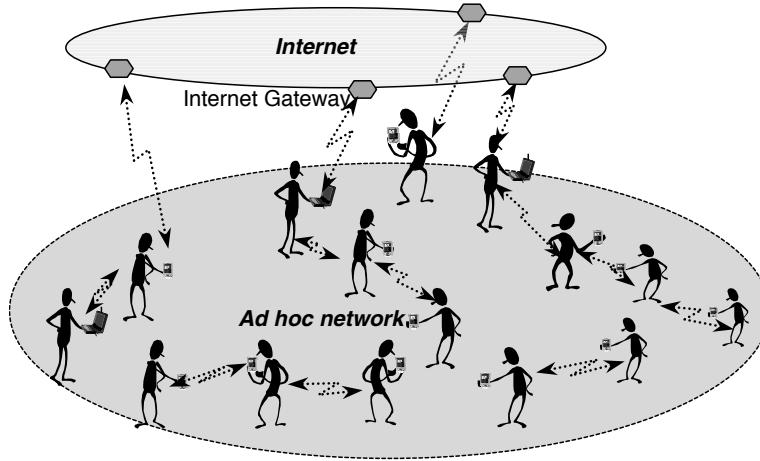


Figure 1: Internet networking for mobile ad hoc networks

As another example, anycasting can be used to find the interconnectivity service provided by Internet gateways. As Fig. 1 shows, mobile devices communicate with each other in MANETs, and there exists also some Internet gateways to which mobile devices can connect. In this case, mobile devices have to find gateways so that they can send their data to the Internet through these gateways. Besides, mobile devices can ask Internet gateways to assign them global addresses, if they need, in addition to their local addresses being unique in the ad hoc network [PWM⁺02].

Anycast can unify gateway discovery as well as the delivery of data packets to gateways. As long as Internet gateways can be identified by well-known anycast addresses, mobile devices can use anycast to reach one “near” gateway.

In summary, anycasting can be applied to service discovery in MANETs:

- Well-known anycast addresses are assigned to services (e.g., Internet access service in MANETs).
- Mobile devices send service requests to those anycast addresses when they want to locate the service interested.
- The network is then responsible for delivering requests to the service providers (e.g. the Internet gateways).

3 AODV anycast algorithm (AODVA)

We have made an extension for anycast routing to the well-established ad hoc routing protocol AODV (Ad Hoc On-Demand Distance Vector protocol) [PR99, PBRD03]. The extension is named AODVA – AODV anycast routing algorithm [WZ02].

The basic concepts of AODV, for example, the so-called *destination sequence number* (DSN), are applied to AODVA as well. The DSN and the unicast address of the anycast

group member identify the freshness of the routing information associated with this specific member with respect to the anycast address. In the following text, the term *nodes* is used to refer to mobile devices.

3.1 Anycast peers

Similarly to AODV, AODVA assumes that communication among nodes in the network is bi-directional and does not need to be reliable. Every node has a unique unicast address, and can have anycast addresses additionally. An anycast address is an address which can be assigned to two or more nodes, which we call *anycast members* of an *anycast group* identified by this anycast address.

AODV discovers routes on-demand. When a node has data packets to send and there is still not a route for the destination of data packets, the node initiates route discovery. The node floods route requests in the network, and route requests contain the destination address for which a route has to be found. Intermediate nodes can answer with route replies if they have a fresh route for that destination. Otherwise, the destination node receives the route requests finally, and answers with a route reply. The traversal of route requests and route replies establishes a bi-direction path between the node originating route requests and the destination node.

Each node maintains a destination sequence number (DSN), which is an increasing number. The DSN is included in routing messages to identify the freshness of the routing information. A node receiving a routing message stores the DSN for the corresponding destination node. Therefore, when a new routing message arrives, the node can determine, by comparison of the locally stored DSN with the newly received DSN, whether the newly received routing information is up-to-date.

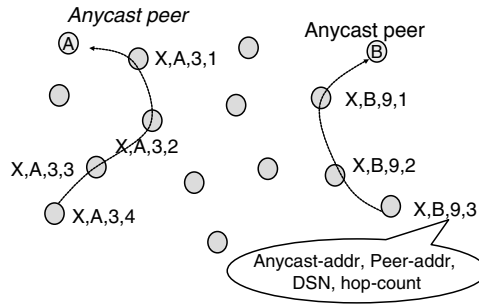


Figure 2: Anycast routes and anycast peers

A node communicates simultaneously with only one anycast member. This particular anycast member is called an *anycast peer* in AODVA. In Fig. 2, for example, different nodes select different anycast members (A and B) as their anycast peers. The labels near each node denote the routing information for the anycast address X stored locally at that node.

Each node maintains an anycast routing table, which keeps the routing information of anycast addresses of interest. It looks just like the routing table used in AODV unicast routing. An entry in the anycast routing table contains an anycast address, the unicast address of the successor node, the unicast address of the anycast peer, the anycast peer's DSN, the distance to the anycast peer (in hops), and the expiration time of the entry. The anycast address and the successor node's address are looked up for forwarding anycast data packets.

3.2 Route discovery and maintenance

An anycast routing message includes the anycast address for which a new route has to be found. It also includes the unicast address and the destination sequence number of its current anycast peer, and the distance (in hops) to its current anycast peer, if the node sending anycast routing messages has such information.

The procedure of route discovery for an anycast address is similar to that for a unicast address in AODV. A node looking for a route floods route requests. Destinations (i. e., anycast members of this anycast group) or intermediate nodes knowing a route can answer with route replies.

When a node receives a route reply for an anycast address, it updates its routing table as described below:

- *peer refresh*: The route reply reports a route to the same anycast peer, and the route reply contains a greater DSN, or an equal DSN but a shorter distance. Just as in the case of unicast routing, the node updates in its routing table the DSN as well as the distance, and selects as its new successor the node from which it has received the route reply.
- *peer revision*: The route reply is sent by its current successor for the anycast address. Since in AODV a successor node has always newer information, the node updates its routing table using the information contained in the route reply.
- *peer switch*: The route reply reports a route with a shorter distance to another anycast peer, and the route reply is not sent by its current successor. Before making route update, the node initiates an *anycast update procedure* to make sure that it uses the up-to-date routing information. It can change its anycast peer as well as the successor node in its routing table when the anycast update procedure finishes successfully.

Maintaining anycast routes in this way prevents from using the stalled routing information upon route update. Therefore, routes for anycast addresses established by AODVA are free of routing loops.

4 Prototype implementation

We have made a prototype implementation of AODVA and use it in a demonstration developed in the BMBF project IPonAir [ZW⁺03, IPo]. For the purpose of demonstration, the provision of pictures captured by webcams is treated to be a "service", which is provided



by two laptops equipped with webcams. Clients want to get the service from the closest service provider, i.e., get pictures from the closest webcam.

The prototype implementation consists of the following components:

- AODV anycast (AODVA) daemon, which enables anycast routing
- Webcam server/client application

For the purpose of demonstration, an additional component called topology controller has been implemented. A real multi-hop WLAN network spans a large geographical area. The topology controller emulates a dynamic multihop network topology, so that the demonstration can be put in a small area (e.g., on a table).

These components as well as the demonstration are described in the next sections in more details.

4.1 AODV anycast routing implementation

AODV anycast routing is implemented by extending an existing AODV implementation called AODV-UU, the AODV implementation from Uppsala University, Sweden. AODV-UU is a user space routing daemon. The daemon processes AODV routing messages, and updates its internal routing table accordingly. The changes of the internal routing table of the daemon are then written into the kernel routing table. To support anycast routing, the internal routing table of AODV-UU is enhanced to accommodate anycast routing entries. Anycast routing algorithm is used to update these anycast routing entries (Fig. 3).

All network packets are captured and delivered to the routing daemon. AODV is an on-demand routing protocol, that is, routes are only established when there are data packets to be sent. Therefore, data packets originated by local processes are captured and delivered from the kernel to the routing daemon, which then checks whether there are routes for data packets. If the routes exist, the data packets are injected into the kernel again. Otherwise, the routing daemon puts the data packets in its local packet buffer, and issues route discovery messages. After the routes are found, these data packets are then removed from the packet buffer and re-injected in the kernel.

Linux netfilter, which is a general framework for packet mangling inside the Linux kernel, provides hooks at which packets can be captured and delivered from the kernel to the user space and re-injected from the user space into the kernel. The module `ip_queue` is a queue handler which registers with the netfilter to perform the mechanics of passing packets between the kernel and the user space [Net].

Routing messages are processed by the routing daemon, which updates the internal routing table accordingly. The changes of the internal routing tables are then written into the kernel through system calls.

4.2 Application software: webcam server/client

The application *webcam server* acts as a provider of a service. A webcam server captures pictures through its attached webcam, and sends these pictures when receiving requests



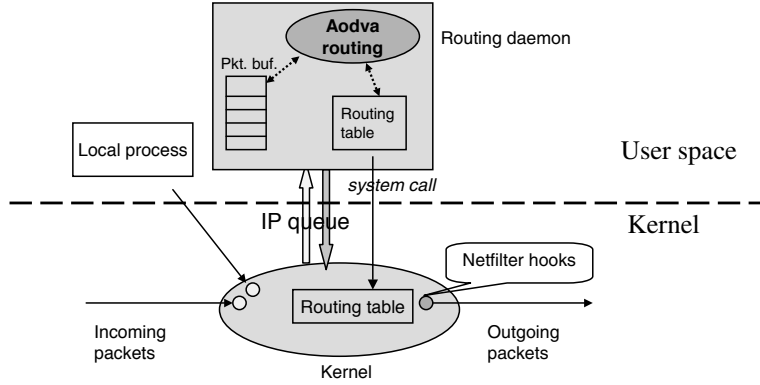


Figure 3: AODVA routing daemon



Figure 4: Graphical user interface of webcam server/client

from clients. The application *webcam client* acts as a client of a service. A webcam client sends requests to webcam servers and shows the received pictures.

The various information of the network is shown on webcam servers and clients: for example, the statistics of routing message, devices in the neighborhood, and the volume of network traffic (Fig. 4).

4.3 Topology controller

To enable the demonstration in a small area, we have implemented an application which we call *topology controller*. The topology controller configures the connectivities among devices so that a dynamic multihop network topology can be emulated.

The main task of the topology controller is to set and show the connectivity between devices. The MAC filter, which is a kind of *iptables netfilter*, allows filtering of network packets using the MAC addresses [Net]. This feature is used to emulate a multihop network. To set the connectivity of a device, the topology controller logs in via secure remote login (SSH) at that device, and sets MAC-filter rules for that device. The MAC-filter rules force packets sent from certain neighbors to be ignored (i.e., not delivered to higher layers)

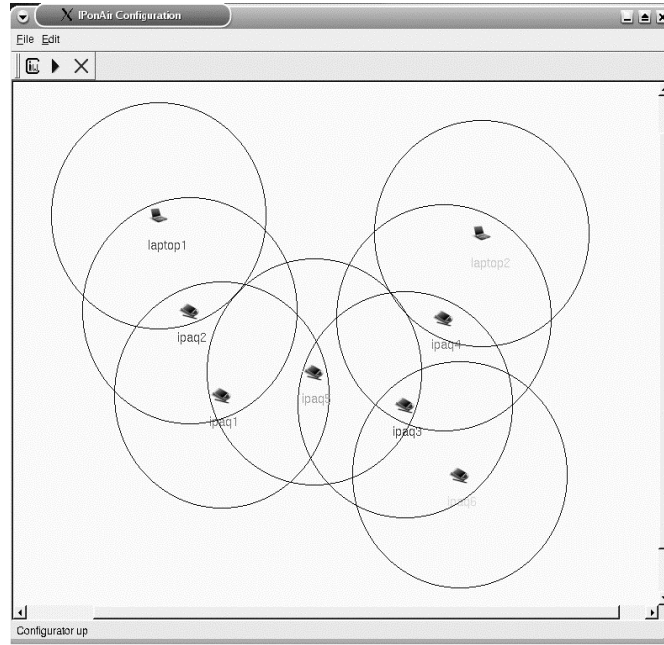


Figure 5: Graphical user interface of the topology controller

The topology controller provides a graphics interface to change and to show the network topology (Fig. 5). Each device is represented by its name and a circle. When a device is located in the circle of another device, it means that these two devices can directly communicate with each other. In order to change the network topology, one can move devices to other positions, and press the “update” button. Then the topology controller logs in at the devices and sets the new MAC filters for each device.

The code of topology controller, which is named *wireless network topology emulator* (WNTE), can be downloaded from the Sourceforge web site [wnt].

4.4 Demonstration setup

The demonstration is developed in the project IPonAir to demonstrate anycast-based services discovery in mobile ad hoc networks.

The demonstration consists of devices such as laptops and HP iPAQs (see Fig. 6). Linux is installed on all devices. Laptops are installed with the Suse Linux distribution, and iPAQs are installed with the Linux distribution developed by the Familiar Project [Fam]. All devices are equipped with IEEE 802.11 WLAN cards and act as nodes of a multihop network.

Two laptops are assumed to provide the same service, and one iPAQ acts as a client sending requests for the service. Both laptops are equipped with webcams, and can provide the pictures captured by webcams to clients. An anycast address is assigned to both laptops.

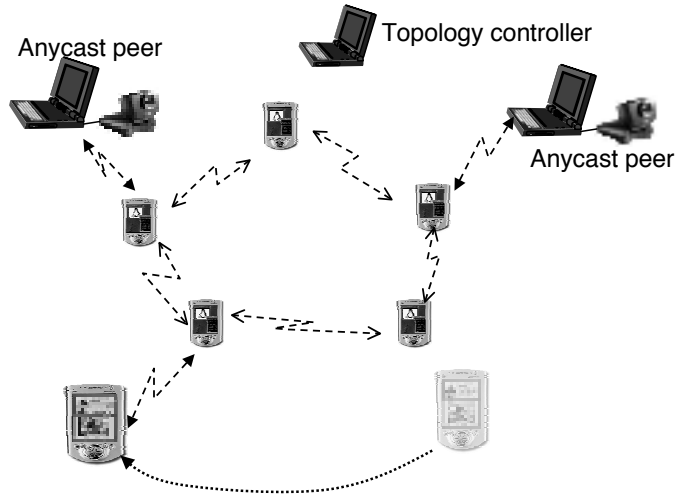


Figure 6: Demonstration setup

The webcam client on the iPAQ issues service requests periodically. These requests are addressed to the anycast address, i.e., the described anycast mechanism is used to deliver service requests to the closest service provider.

All devices are actually in direct wireless transmission range of each other. Therefore, an additional laptop is used as a topology controller so that a dynamic multihop network topology is emulated. When the connections of an iPAQ are changed, i.e., when the iPAQ “moves”, it gets pictures from the closest webcam after the movement. The pictures captured by the webcams are different so that one can distinguish which laptop is providing pictures.

5 Summary

As MANETs are self-organized, dynamical service discovery mechanisms are needed for mobile devices to locate service providers. As a result, service discovery becomes an active research issue on mobile ad hoc networking. We propose to utilize anycast as an efficient means of service discovery in MANETs. Providers of services are identified with well-known anycast addresses, and service clients can find services with anycast.

Since anycast routing is still not supported by the state-of-the-art ad hoc routing protocols, we have developed an anycast extension to AODV. The extension is used as a key component of a demonstration, which shows the application of anycast-based service discovery.

Nevertheless, further research is needed on anycast-based service discovery in MANETs. Some recent research work proposes to use Mobile IP to support Internet access for MANETs, but it usually assumes there is only one Internet gateway in a MANET. We plan to enhance the existing work by utilizing anycast so that mobile devices in a MANET can find and communicate with the closest Internet gateway when multiple Internet gateways exist.



Anycast can be also used to enhance the exist service discovery protocols. For example, directory-based approaches use repositories of service information, which act as an intermediary between providers and clients of services. Anycast can be used to find the closest repository.

6 Acknowledgments

We thank the German Federal Ministry of Education and Research (BMBF) for the support which enables our work in the IPonAir project (<http://www.iponair.de>). We also thank Ingmar Baumgart and Yusuf Iskenderoglu for their contribution to the implementation of the demonstration.

References

- [Fam] Familiar project. <http://familiar.handhelds.org>.
- [GPVD99] Erik Guttman, Charles E. Perkins, John Veizades, and Michael Day. Service location protocol, version 2. RFC 2608, June 1999.
- [IPo] IPonAir project. <http://www.iponair.de>.
- [Net] Netfilter/iptables project. <http://www.netfilter.org>.
- [PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. RFC 3561, July 2003.
- [PMM93] Craig Partridge, Trevor Mendez, and Walter Milliken. Host anycasting service. RFC 1546, November 1993.
- [PR99] Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.
- [PWM⁺02] C. Perkins, R. Wakikawa, J. Malinen, A. Nilsson, and A. Tuominen. Internet connectivity for mobile adhoc networks. *Wireless Communications and Mobile Computing*, (2):465–482, 2002.
- [Sun01a] SunSoft. Jini architecture specification ,version 1.2, December 2001.
- [Sun01b] SunSoft. Jini technology core platform specification ,version 1.2, December 2001.
- [wnt] SourceForge.net: Wireless network topology emulator. <http://sourceforge.net/projects/wnte>.
- [WZ02] J. Wu and M. Zitterbart. Extension for anycasting in ad hoc on-demand distance vector protocol. In *Paper Digest of the 12th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN) 2002*, Stockholm, Sweden, August 2002.
- [ZW⁺03] M. Zitterbart, K. Weniger, et al. IPonAir - drahtloses internet der naechsten generation. *PIK Themenheft: Mobile Ad-hoc-Netzwerke*, December 2003.

