DESIGN AND IMPLEMENTATION OF IPV6 ADDRESS AUTOCONFIGURATION FOR AODV IN MOBILE AD HOC NETWORKS[*]

Youngmin Kim[1], Sanghyun Ahn[1], Youngju Lee[1], Jaehoon Jeong[2], and Jaehwoon Lee[3]

Summary

An advantage of the mobile ad hoc network (MANET) is that mobile nodes can self-organize the network topology without the help of network infrastructure. However, for the perfect self-organization of the MANET, each mobile node needs to self-configure its address. Even though a mobile node configures a unique address during the booting time, its address may conflict with nodes in other MANETs since MANETs containing the same address can be merged. The address autoconfiguration protocol implemented in this work consists of the strong DAD (Duplicate Address Detection) and the weak DAD. A unique address of a node is assigned by the strong DAD during the booting time and the weak DAD is used to detect address conflict and resolve address conflict during the ad hoc routing. In this work, we have implemented address autoconfiguration in the IPv6-based MANET using AODV as the routing protocol. We describe how the IPv6 address autoconfiguration is implemented and verify our implementation by showing the test scenarios on our testbed.

Introduction

Mobile nodes in MANET can self-organize multi-hop wireless paths without the help of network infrastructure. Thus, public interest of ad hoc networking is being increased not only in battle fields or emergencies where the infrastructured network is difficult to deploy, but also at home and office.

Most researches on MANET focus on the area of routing, and the IETF MANET working group has published AODV (Ad Hoc On Demand Distance Vector) [1][2], OLSR (Optimized Link State Routing), TBRPF (Topology Dissemination Based on Reverse-Path Forwarding) and so on as RFCs. However, in order to accomplish the self-organization of the network topology, mobile nodes should be able to self-configure their addresses as well as routes. Even though a mobile node self-configures its address during the booting time, address conflict may happen when two or more MANETs are merged. Therefore, address autoconfiguration must have a scheme to detect and resolve this address conflict. The node that has to resolve address conflict also needs a scheme to keep the on-going connections associated with the conflicted address, such as TCP sessions.

Due to the advance in networking technology and the desire of human being for convenience, a number of electronic equipments can be handled via networks. The IPv6 address, which has the address space of 128 bits, will be adopted as the standard in order to be able to assign unique

[1] University of Seoul, Seoul, 130-743, Korea
[2] University of Minnesota, Minneapolis, MN 55455, USA
[3] Dongguk University, Seoul, 100-715, Korea
* This work was sponsored by ETRI (Electronics and Telecommunications Research Institute).
* This work was supported by grant No. R01-2004-10372-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

addresses to more devices. Therefore, in our implementation, we have adopted the IPv6 addressing.

In this work, we have designed and implemented the IPv6 address autoconfiguration protocol for AODV on the basis of the work of the ad hoc IP address autoconfiguration for AODV [3] and show how the IPv6 address autoconfiguration protocol works in our testbed. The ad hoc IP address autoconfiguration for AODV defines message formats for address autoconfiguration, and explains how the address autoconfiguration protocol is integrated with AODV.

Related Work

Address autoconfiguration protocols can be classified into stateful and stateless approaches. DHCP (Dynamic Host Configuration Protocol) [4] is the representative protocol of the stateful approach, which is usually used in wired networks. Address conflict never occurs in the network using DHCP because the DHCP server keeps state information on the addresses that have already been assigned.

In MANET where each mobile node can move around, it is not feasible that a mobile node maintains a connection with a DHCP server and a DHCP server manages address information on nodes. Thus, it is a more feasible approach that mobile nodes make use of the stateless approach where each node selects its address and asks other nodes whether it conflicts with others. The IPv6 stateless address autoconfiguration [5][6] is designed for wired networks, and can assign a unique address by exchanging information with only directly connected nodes.

However, the IPv6 stateless address autoconfiguration designed for a subnet link [6] does not work in the multihop ad hoc network, thus [7] presents a new scheme of address autoconfiguration for MANET. A mobile node places a randomly selected address in an address request (AREQ) message and broadcasts it to MANET to which it belongs. If the address of a node receiving the AREQ message is the same as that of the AREQ message, the node replies with an address reply (AREP) message via the reverse route established by the AREQ message. On the other hand, the originator of the AREQ message can know that the candidate address does not conflict with others if it receives no AREP message until the timer for the AREP message expires, and it configures the address in its network interface as a permanent address. A serious drawback of this scheme is that address duplication due to network merging cannot be resolved. The strong DAD (Duplicate Address Detection) is the time-based DAD based on the hop-count and the timer of the DAD control message for the purpose of checking if there is address duplication in a connected MANET partition within a finitely bounded time [8]. However, the strong DAD may not detect a duplicate address of a node out of the range of the hop-count (or TTL) of the DAD control message, so as a result the weak DAD has been proposed [8].

The Weak DAD [8] and the passive DAD [9] are able to detect and resolve even address conflict that can arise after two or more MANETs merge. A mobile node adds a key to the routing control message in the weak DAD. The key can be chosen once by each node either randomly or based on a universally unique ID. Each node stores both the address and the interface key for address autoconfiguration. Then, a node can figure out address conflict when the address of the node is the same as that of the sender of a routing control message but their interface keys are different. The passive DAD can detect address duplication with using the sequence number of the routing control message and without using the additional information such as the interface key.

In this work, we have implemented the ad hoc IPv6 address autoconfiguration for AODV on the basis of the Internet drafts of Jeong [3][10][11] that contain the strong DAD, the weak DAD and the maintenance of upper-layer sessions. In [11], the requirements for IP address autoconfiguration in ad hoc networks are specified. [10] shows steps to autoconfigure IPv4 or IPv6 addresses, defines message formats, and specifies how to resolve the address duplication in order to guarantee the maintenance of upper-layer sessions, such as TCP sessions.

Ad hoc IPv6 address autoconfiguration for AODV

The ad hoc IPv6 address autoconfiguration for AODV consists of the strong DAD and the weak DAD [3][10]. The address of a node is assigned by the strong DAD during the booting time. And the weak DAD is used to detect address conflict and resolve the conflict by routing control messages after booting because the strong DAD may not detect duplicate addresses caused by network merging. Three kinds of addresses are used in the strong DAD: temporary, tentative and permanent address. A temporary address is temporarily used as the source address of a node during the strong DAD, and a tentative address is a candidate address included in the AREQ message. After the timer for address duplication checking expires, the tentative address becomes a permanent address and the node configures the address in its wireless network interface.

When a mobile node moves, its address may conflict with those of other nodes either out of range of the hop-count of a control message or in other MANETs. The weak DAD can detect and resolve this address conflict through routing control messages including additional fields, such as the network interface key.

Each node makes use of both the address and the interface key in order to perform the weak DAD. If a mobile node detects address conflict by the weak DAD and has sessions that were generated while the conflicted address was used, the node needs to maintain the upper-layer sessions through some address handover even after the duplicate address is replaced with a new address. Our address handover scheme is based on the IP-over-IP tunneling where the duplicate address is placed in the source address of the inner IP header and the new address in the source address of the outer IP header [3][10]. We will describe how the ad hoc IPv6 address autoconfiguration for AODV is implemented in detail.

Implementation of the ad hoc IPv6 address autoconfiguration for AODV

We have implemented the ad hoc IPv6 address autoconfiguration for AODV, one of the most popular MANET routing protocols, on the linux kernel 2.4.21. We modified the AODV of NIST [12] to make AODV protocol for IPv6.

Fig. 1 shows the message formats for the ad hoc IPv6 address autoconfiguration for AODV. The detailed message formats are specified in [3][10]. The strong DAD is carried out by RREQ-AREQ and RREP-AREP messages that include a tentative address. Each mobile node can detect address conflict with the RREQ or RREP message including the interface key, and resolve address conflict with the RREP-AERR message by means of the weak DAD.

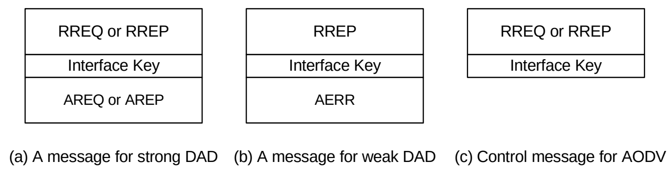

Figure 1 Message formats for the ad hoc IPv6 address autoconfiguration for AODV

Fig. 2 shows how a RREQ-AREQ message is transmitted as a part of the strong DAD. Temporary and tentative addresses are randomly chosen to build a RREQ-AREQ message and broadcasted MANET-widely. If a node receives a RREP-AREP message from another node which implies address duplication, it transmits another RREQ-AREQ message with a newly generated tentative address. The node fails to have a permanent address if it continues to send RREQ-AREQ messages until fail_cnt becomes zero, indicating that every tentative address, a candidate address for a permanent address, has conflicted with the addresses occupied by other nodes. The tentative address is chosen as a permanent address of the node only when the node has never received any RREP-AREP messages until the number of retransmitted RREQ-AREQ messages becomes AREQ-RETRIES. The reason for performing address duplicate detection multiple times is to ensure the detection of address conflict in the dynamic and error-prone MANET environment.

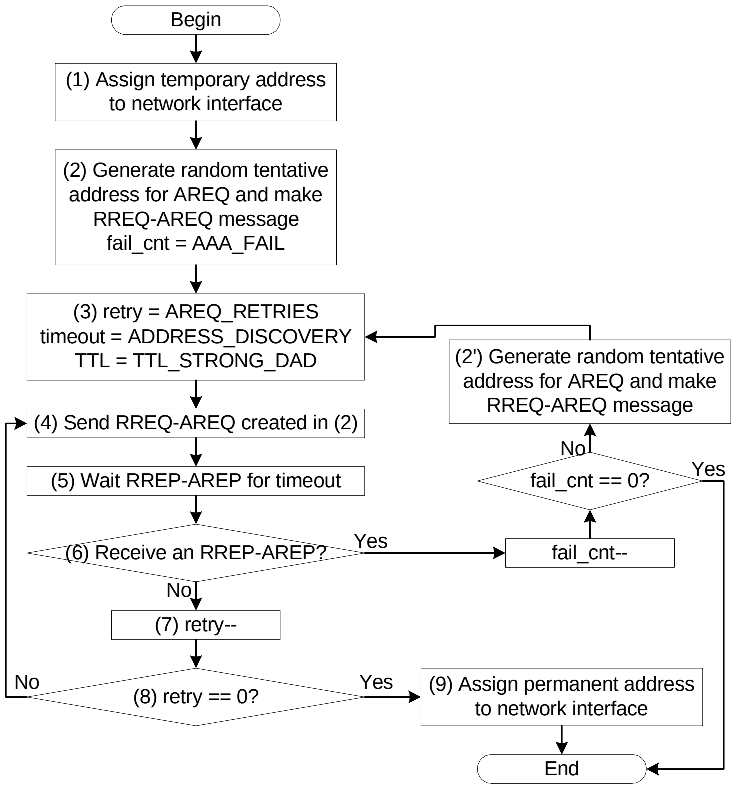
Begin
(1) Assign temporary address to network interface
(2) Generate random tentative address for AREQ and make RREQ-AREQ message fail_cnt = AAA_FAIL
(3) retry = AREQ_RETRIES timeout = ADDRESS_DISCOVERY TTL = TTL_STRONG_DAD
(2') Generate random tentative address for AREQ and make RREQ-AREQ message
(4) Send RREQ-AREQ created in (2)
No
fail_cnt == 0?
Yes
(5) Wait RREP-AREP for timeout
(6) Receive an RREP-AREP?
Yes
fail_cnt--
No
(7) retry--
No
(8) retry == 0?
Yes
(9) Assign permanent address to network interface
End

Figure 2 Strong DAD procedure at a newly joined node

The netfilter [13] of linux kernel is used to handle messages for the ad hoc IPv6 address autoconfiguration for AODV. When a node receives a RREQ-AREQ message, if the address of the node is different from the tentative address of the RREQ-AREQ message, it stores the information in order to prevent it from receiving duplicate packets and sends the packet to event_queue so that it can rebroadcast the RREQ-AREQ message. If the addresses are the same, the node generates a RREP-AREP message and sends it to event_queue so that the kernel thread aaa_thread can transmit the packet to the originator of the RREQ-AREQ message in order to inform it of address conflict.

In order to detect address conflict in the middle of route setup of AODV, the node checks whether both the originator address and the interface key of the received packet are the same as those of the node itself or in routing table. If the addresses do not conflict, the packet is processed as normal. Otherwise, it generates a RREP-AERR message with code 0 indicating the occurrence of address conflict and sends to event_queue. This procedure is specified in Fig. 3.

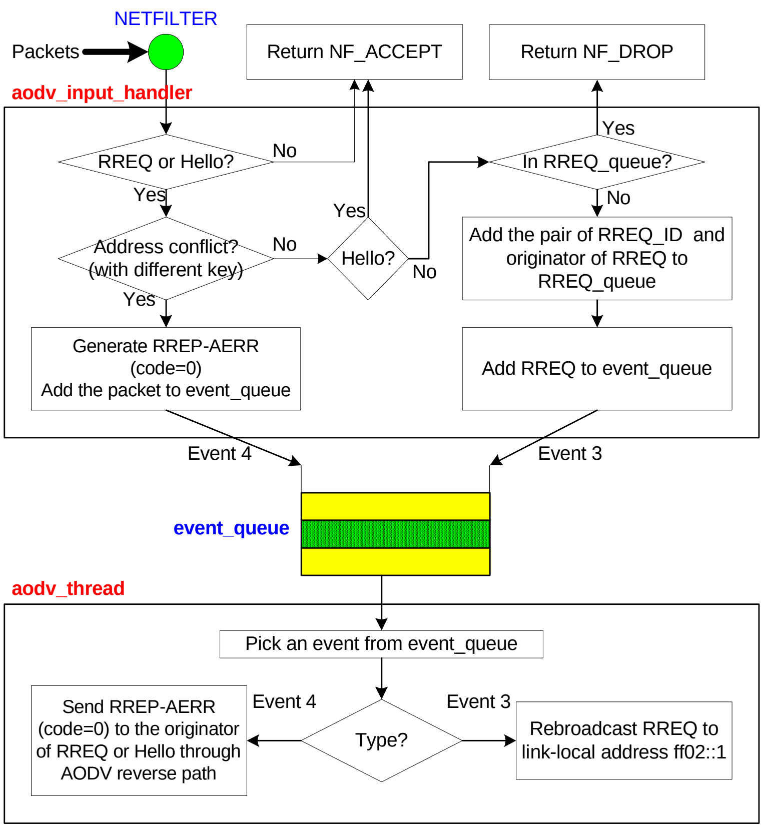
NETFILTER
Packets
Return NF_ACCEPT
Return NF_DROP
aodv_input_handler
RREQ or Hello?
No
Yes
Yes
In RREQ_queue?
No
Address conflict? (with different key)
No
Yes
Hello?
No
Add the pair of RREQ_ID and originator of RREQ to RREQ_queue
Yes
Generate RREP-AERR (code=0) Add the packet to event_queue
Add RREQ to event_queue
Event 4
Event 3
event_queue
aodv_thread
Pick an event from event_queue
Send RREP-AERR (code=0) to the originator of RREQ or Hello through AODV reverse path
Event 4
Type?
Event 3
Rebroadcast RREQ to link-local address ff02::1

Figure 3 Weak DAD procedure during route setup

The node that receives a RREP-AERR message can resolve the conflict through the procedure in Fig. 4. If the node is not the destination of the RREP-AERR message, it forwards the message to the next hop. Otherwise, it takes a proper action according to the code in the RREP-AERR. Since code 0 means that another node with the same address exists in the MANET, the node changes its address through the strong DAD. In addition, it should inform its peer nodes of its new address by sending RREP-AERR messages with code 1 so that the communicating sessions can be preserved. When a node receives a RREP-AERR message with code 1, it can become aware that the originator address of the message is changed. And, the node adds information about old and new addresses to its address mapping cache address_map_cache maintaining the association of a duplicate address and a new address so that it can preserve the old sessions through the IP-over-IP tunneling. The node should also send a RREP-AERR message with code 2 to the originator of the RREP-AERR message with code 1 as a reply in order to confirm that it recognizes the address change and is ready to receive packets using the new source address.

Tests in a MANET testbed

Four laptops, installing redhat 9.0 with kernel 2.4.21, were prepared in order to test the implemented address autoconfiguration for AODV. A tentative address should be generated randomly, but we added an ability to assign a predefined address to the tentative address so that address conflict can occur intentionally.

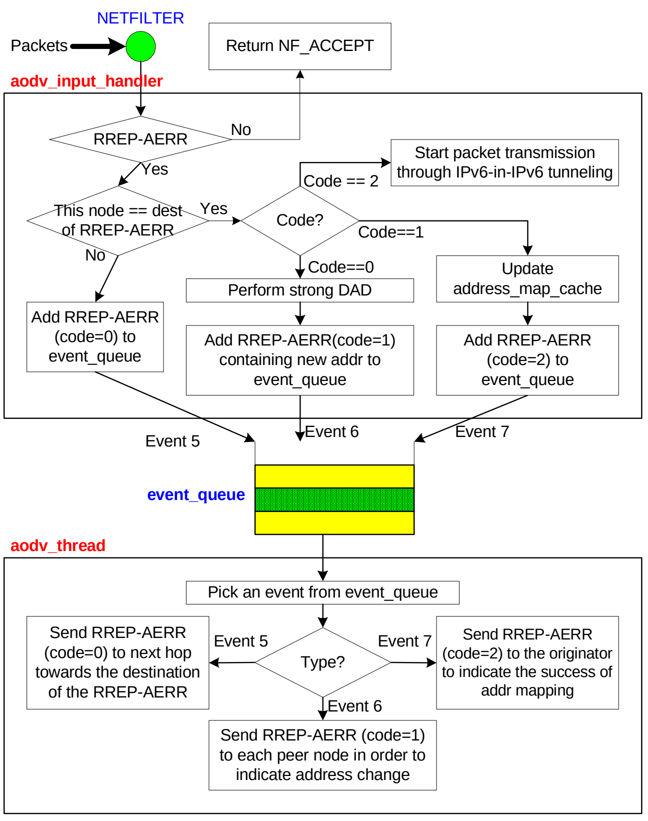
NETFILTER
Packets
Return NF_ACCEPT
aodv_input_handler
RREP-AERR
No
Yes
Start packet transmission through IPv6-in-IPv6 tunneling
Code == 2
This node == dest of RREP-AERR
Yes
Code?
Code==1
No
Code==0
Perform strong DAD
Update address_map_cache
Add RREP-AERR (code=0) to event_queue
Add RREP-AERR(code=1) containing new addr to event_queue
Add RREP-AERR (code=2) to event_queue
Event 5
Event 6
Event 7
event_queue
aodv_thread
Pick an event from event_queue
Send RREP-AERR (code=0) to next hop towards the destination of the RREP-AERR
Event 5
Type?
Event 7
Send RREP-AERR (code=2) to the originator to indicate the success of addr mapping
Event 6
Send RREP-AERR (code=1) to each peer node in order to indicate address change

Figure 4 Procedure at a node receiving a RREP-AERR message

Node B and C have their own addresses, but node A does not have an address yet in Fig. 5. Therefore, node A has to autoconfigure an address via the strong DAD. Node B and C already know each other by exchanging Hello messages. Node A chooses ADDR3, the same address as node C, as its tentative address and then broadcasts a RREQ-AREQ message so that node B having a route entry for node C is able to detect address conflict.

Since a route entry having the same address as the tentative address of the received RREQ-AREQ message with the different interface key already exists in the routing table of node B (i.e., ADDR3, ADDR3, KEY3), node B transmits a RREP-AREP message to node A. Both node A and C receive the message, but node C discards it because the interface keys are different. Node A transmits a RREQ-AREQ message with a new tentative address after receiving the RREP-AREP message. If node A does not receive any RREP-AREP messages until the timer expires, it retransmits the RREQ-AREQ message with the same tentative address. If the node does not receive any RREP-AREP messages while it retransmits the same RREQ-AREQ message three times, it configures the tentative address in its wireless interface as a permanent address.

In Fig. 6, MANET 1 consists of node A, B and C, and MANET 2 has only node D, and the address of node A is the same as that of node D. Node C and D update their routing tables by exchanging Hello messages after merging. Node C cannot detect address conflict up to then. If node D broadcasts a RREQ message with its interface key in order to communicate with node B, the node B can detect address conflict and transmits to node D the RREP-AERR message with code 0 because it knows the information on node A. Then, node D performs the strong DAD to autoconfigure its new address.

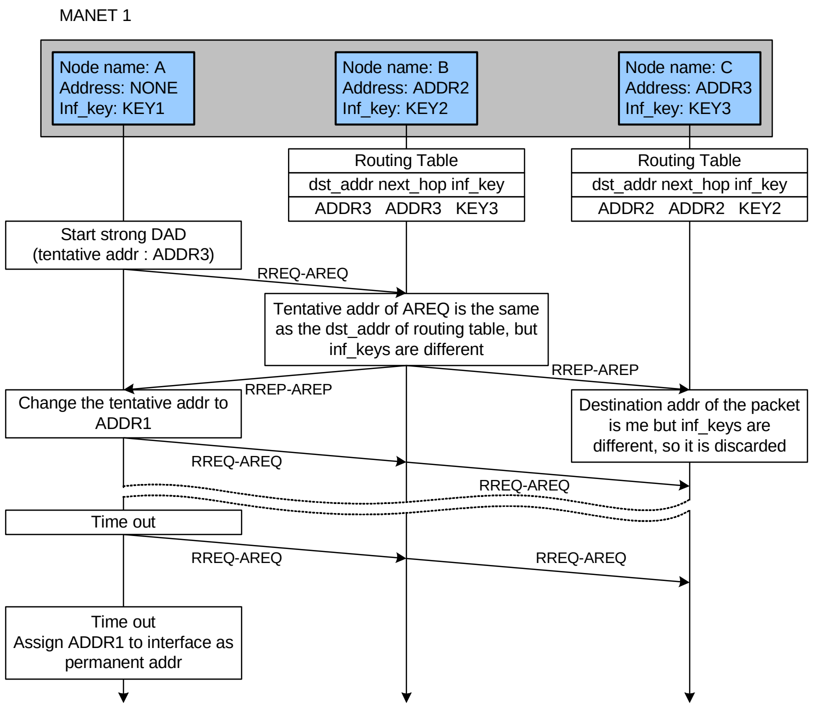
MANET 1
Node name: A
Address: NONE
Inf_key: KEY1
Node name: B
Address: ADDR2
Inf_key: KEY2
Node name: C
Address: ADDR3
Inf_key: KEY3
Routing Table
dst_addr next_hop inf_key
ADDR3 ADDR3 KEY3
Routing Table
dst_addr next_hop inf_key
ADDR2 ADDR2 KEY2
Start strong DAD
(tentative addr : ADDR3)
RREQ-AREQ
Tentative addr of AREQ is the same as the dst_addr of routing table, but inf_keys are different
RREP-AREP
RREP-AREP
Change the tentative addr to ADDR1
Destination addr of the packet is me but inf_keys are different, so it is discarded
RREQ-AREQ
RREQ-AREQ
Time out
RREQ-AREQ
RREQ-AREQ
Time out
Assign ADDR1 to interface as permanent addr

Figure 5 Test scenario for the verification of the strong DAD

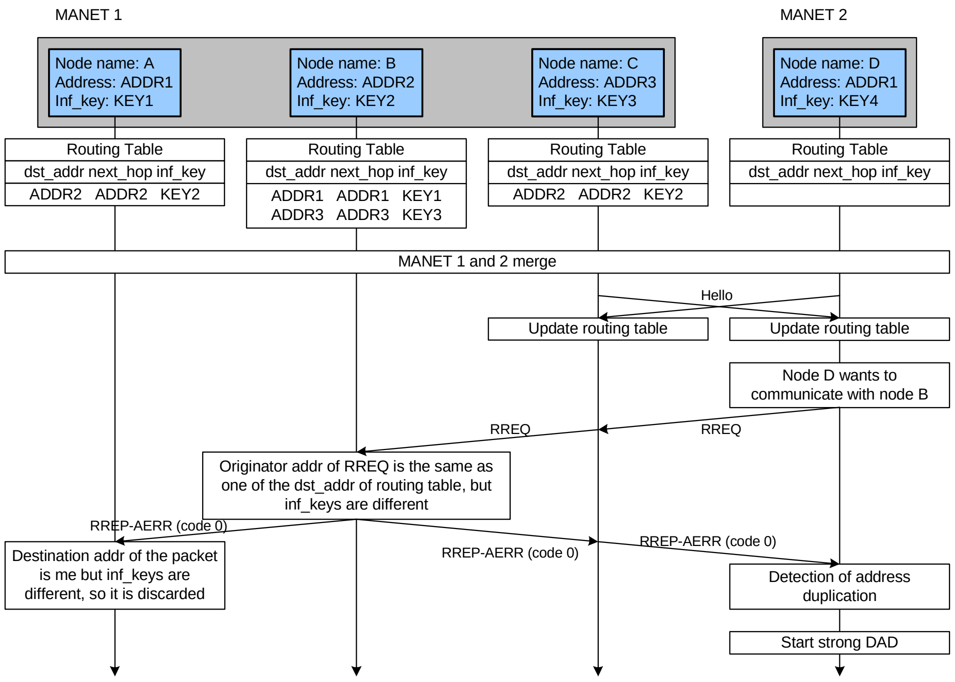

Figure 6 Test scenario for the verification of the weak DAD when two networks merge

We have verified the correctness of our implemented address autoconfiguration for AODV. However, since linux kernel 2.4.21 does not provide the tunneling mechanism, we could not implement the modules to maintain upper-layer sessions.

Conclusion and future work

We have presented the design and the implementation of the ad hoc IPv6 address autoconfiguration to facilitate the address configuration of mobile nodes placed in the MANET based on AODV. A mobile node not only can autoconfigure its address during the booting time via the strong DAD, but also can resolve address conflict during the ad hoc routing via the weak DAD and address handover based on the IP-over-IP tunneling. Besides, we have verified that the strong and the weak DAD work well in our MANET testbed.

As future work, this implementation will have to include the passive DAD that can detect address conflict without additional information such as the interface key. Moreover, we will need to remove the redundant information between routing control messages (e.g., RREQ or RREP) and the messages for address autoconfiguration (e.g., AREQ or AREP) so that the size of the entire message can be reduced.

References

1. C. Perkins, E. Belding-Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.
2. C. Perkins, E. Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing for IP version 6", Internet draft, draft-ietf-manet-aodv6-01.txt, Nov. 2000.
3. J. Jeong et al., "Ad Hoc IP Address Autoconfiguration for AODV", Internet draft, draft-jeong-manet-aodvaddr-autoconf-01.txt, July 2004.
4. R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, Mar. 1997.
5. T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP version 6", RFC 2461, Dec. 1998.
6. S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, Dec. 1998.
7. C. Perkins et al., "IP Address Autoconfiguration for Ad Hoc Networks", Internet draft, draft-ietf-manetautoconf-01.txt, Nov. 2001.
8. N. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks", In *Proc. ACM MobiHoc 2002*, June 2002.
9. K. Weniger, "Passive Duplicate Address Detection in Mobile Ad Hoc Networks", In *Proc. IEEE WCNC 2003*, Mar. 2003.
10. J. Jeong et al., "Ad Hoc IP Address Autoconfiguration", Internet draft, draft-jeong-adhoc-ip-addrautoconf-04.txt, Feb. 2005.
11. J. Jeong et al., "Requirements for Ad Hoc IP Address Autoconfiguration", Internet draft, draft-jeong-manetaddr-autoconf-reqts-04.txt, Feb. 2005.
12. Kernel AODV. http://w3.antd.nist.gov/wctg/aodv kernel/
13. Netfilter. http://www.netfilter.org/