Project EEE08

Seamless IPv6 Handover in V2X Environment

Lee Jia Ying
Meng Fanbo

Mentored by: Associate Professor Guan Yong Liang
Dr Yang Ming
Introduction

Usage of IoT (Internet of Things) rapidly increases in our time

Demand for delay-sensitive real-time applications will increase

Requirement of seamless handoff to ensure no loss packets during IP address configuration
Figure A1
Demonstration of IPv6 Handover Process
Current solutions

Various protocols improve the efficiency of IPv6 Handover

Therefore improving the speed of IPv6 Handover

Proposed Solution

Change of the environment of the program

Increase in efficiency - administrative processes can be bypassed
Research Objectives

01 Work on IPv6 Handover process in a Linux Environment

02 Determine suitable parameters to ensure possible IPv6 Handover
Methodology

- bypass the System Command step
- functions in the API to proceed directly to the Kernel
1. Deciding on Final Kernel Libraries
Includes the necessary headers:

```c
#include <asm/types.h>
#include <linux/netlink.h>
#include <linux/rtnetlink.h>
#include <sys/socket.h>
```

```c
rtnetlink_socket = socket(AF_NETLINK, int socket_type, NETLINK_ROUTE);
```

1. `rtnetlink`
   - Allows the Kernel’s routing tables to be read or altered

2. `netlink`
   - Used to transfer information between the Kernel and user-space processes
2. Addition and deletion of Multiple IPv6 addresses

Using kernel library functions:

- RTM_NEWADDR
- RTM_GETADDR
- RTM_DELADDR

Attributes of the address:

- RTA_LENGTH
- Prefix length
- ADD/DEL IP address
Adding of Attribute: RTA_LENGTH

```
int rtln_talk(struct rtln_handle *rtln, struct nlmsg hdr *n, pid_t peer,
             unsigned groups, struct nlmsg hdr *answer)
{
    int status;
    struct nlmsg hdr *h;
    struct sockaddr nl nladdr;

    struct iovec iov = { (void*)n, n->nlmsg_len };
    char buf[1024];
    // Forming the message to be sent.
    struct msghdr msg = { (void*)&nladdr, sizeof(nladdr), &iov, 1, NULL, 0, 0 };  
```

Payload of header (to delete address)
// This is the function for adding the attributes to the message.
int addattr_l(struct nlmsghdr *n, int maxlen, int type, void *data, int alen)
{
    int len = RTA_LENGTH(alen);
    struct rtattr *rta;

    if (NLMSG_ALIGN(n->nlmsg_len) + len > maxlen)
        return -1;
    rta = (struct rtattr*)((char*)n + NLMSG_ALIGN(n->nlmsg_len));
    rta->rtattr_type = type;
    rta->rtattr_len = len;
    memcpy(RTA_DATA(rta), data, alen);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + len;
    return 0;
}
3. Addition of address labels

Two types of addresses

1. Primary (only 1)
   - where packets are sent and downloaded

2. Deprecated
   - where packets are downloaded
Should an IPv6 Address be **preferred**, it will have a preferred lifetime.

Should an IPv6 Address be **deprecated**, it will have an expired lifetime.
Adding of attribute containing flags

```c
struct ifa_cacheinfo ci;
// first test: added as preferred IP address
  ci.ifa_valid = 0xFFFFFFFFFUL;
  ci.ifa_preferred = 0xFFFFFFFFFUL;
// second test: added as deprecated IP address
  ci.ifa_valid = 0xFFFFFFFFFUL;
  ci.ifa_preferred = 0;
  addattr_l (&req.n, sizeof(req), IFA_CACHEINFO,
             &ci, sizeof(struct ifa_cacheinfo));
```
4. Addition of Routing

- Routing establishes the route taken by packets to their destination address

  Function type RTM_ROUTE is used, with payload containing routing table related information.
Modification of Routing Table:
• Different set of attributes

1. Destination Address
2. Interface
3. Route type, etc.
payload of RTM_ROUTE message

```c
memset(&req, 0, sizeof(req));

req.n.nlmsg_len = 128;
req.n.nlmsg_flags = NLM_F_REQUEST | NLM_F_CREATE;
req.n.nlmsg_type = RTM_NEWROUTE;

// setup the service header (struct rtmsg)
req.rth.rtm_family = AF_INET6;
req.rth.rtm_table = RT_TABLE_MAIN;
req.rth.rtm_protocol = RTPROT_STATIC;
req.rth.rtm_scope = RT_SCOPE_UNIVERSE;
req.rth.rtm_type = RTN_LOCAL;

req.rth.rtm_dst_len = 128;
req.ifra.ifra_prefixlen = 64;
addattr_l (&req.n, sizeof(req), IFA_LOCAL, sizeof(IFA_LOCAL))
```
Results and Discussion
1. Successful addition and deletion of IPv6 Addresses

• Different functions, causing there to be modifications to payload
Successful addition of IPv6 addresses

```
root@ubuntu:~# ./aaa
root@ubuntu:~# ifconfig
ens33: Flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 192.168.47.132  netmask 255.255.255.0  broadcast 192.168.47.255
       inet6 fe80::40de:bcde:98a5:797f/64  prefixlen 64  scopeid 0x20<link>
       inet6 aaaa:bbbb:cccc::234  prefixlen 32  scopeid 0x0<global>
       inet6 aaaa:bbbb:cccc::123  prefixlen 32  scopeid 0x0<global>
       ether 00:0c:29:bb:f3:75  txqueuelen 1000  (Ethernet)
       RX packets 1291  bytes 1046646 (1.0 MB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 437  bytes 47133 (47.1 KB)
       TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo:  Flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
     inet 127.0.0.1  netmask 255.0.0.0  broadcast 127.255.255.255
     inet6 ::1  prefixlen 128  scopeid 0x10<host>
     loop txqueuelen 1000  (Local Loopback)
     RX packets 182  bytes 12869 (12.8 KB)
     RX errors 0  dropped 0  overruns 0  frame 0
     TX packets 182  bytes 12869 (12.8 KB)
     TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```
Successful deletion of IPv6 addresses

```
^[[Aroot@ubuntu:~# ./aaa
root@ubuntu: # ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
   inet 192.168.47.132 netmask 255.255.255.0 broadcast 192.168.47.255
   inet6 fe80::40de:bcde:98a5:797f prefixlen 64 scopeid 0x20<link>
   ether 00:0c:29:bb:f3:75  txqueuelen 1000  (Ethernet)
   RX packets 1395  bytes 1099702 (1.0 MB)
   RX errors 0  dropped 0  overruns 0  frame 0
   TX packets 473  bytes 52231 (52.2 KB)
   TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
   inet 127.0.0.1 netmask 255.0.0.0
   inet6 ::1 prefixlen 128 scopeid 0x10<host>
   loop txqueuelen 1000  (Local Loopback)
   RX packets 199  bytes 14062 (14.0 KB)
   RX errors 0  dropped 0  overruns 0  frame 0
   TX packets 199  bytes 14062 (14.0 KB)
   TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```
2. Successful labelling of primary or deprecated addresses
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
   inet6 aaaa:bbbb:cccc::345/32 scope global deprecated dynamic
       valid_lft 268435443sec preferred_lft 0sec
   inet6 aaaa:bbbb:cccc::234/32 scope global dynamic
       valid_lft 268435417sec preferred_lft 268435417sec
   inet6 aaaa:bbbb:cccc::123/32 scope global dynamic
       valid_lft 268435297sec preferred_lft 268435297sec
   inet6 fe80::40de:bcde:98a5:797f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
### Attributes for this request

- **Valid lifetime** of each address label is set to be **forever**
- **Preferred lifetime** of a deprecated address is set to be **0**.

```c
struct ifa_cacheinfo ci;
    // first test: added as preferred IP address
    ci. ifa_valid = OxFFFFFFFFUL;
    ci. ifa_preferred = OxFFFFFFFFUL;
    //second test: added as deprecated IP address
    ci. ifa_valid = OxFFFFFFFFUL;
    ci. ifa_preferred = 0;
    addattr_l (&req.n, sizeof(req), IFA_CACHEINFO, &ci, sizeof(struct ifa_cacheinfo));
```
3. Successful identification of routing attributes
• Attributes needed in payload for routing modifications include:

1. Destination Address
2. Interface
3. Route type
4. Prefix length
5. Appropriate flags, etc.
Conclusion
Successful choosing of Kernel Libraries

Successful adding and deleting of IPv6 addresses on a single interface

Successful labelling of addresses

Successful identification of routing attributes
These findings work towards enabling Linux to be a viable platform for IPv6 Handover processes in the future.
Thank you for your attention!
References


