MCoA++ Manual

Bruno Sousa

2011-03-26

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Document provides some insights in the Multiple Care of Address Implementation (MCoA++) done for OMNET++ simulator.

DISCLAIMER:
This manual is not complete, and no warranty is provided, no revision has been made, it has been elaborated based on personnal notes, use as is.

Chapter 2

MCoA++ Detail

The MCoA++ is based on xMIPv6 Implementation done by Zarrar et al. MIPv6 has been extended to support the registration of multiple addresses.

This chapter includes details on the MCoA++ implementation.

2.1 History

There are two versions of MCoA implemented. The first on implemented in INET and was an extension to xMIPv6 model. As we have proceed with implementation, the second version was migrated to INETMANET.

Version	**Author**	**Enhancements**
1.0	Marco and Bruno	Multiple Care of address implementation in xMIPv6 model.
2.0	Bruno Sousa	Implementation in INETMANET. Bug fixes.

Table 2.1: Table with implementations history

2.2 Data Structures and Classes

2.2.1 KeyMCoADAD

C++ class: `KeyMCoADAD`

Package: `networklayer.mcoa`

This class is used to distinguish the different addresses on a interface basis for the DAD mechanism

2.2.2 KeyMCoABind

C++ class: `MCoABind`

Package: `networklayer.mcoa`

As the identification of the binding is no longer done based solely on the address, a new structure was created to enable the distinction of bindings in different nodes. This class includes different parametes, as presented in 2.2 and 2.3.

2.2.3 MCoA

C++ class: `MCoA`

Package: `networklayer.mcoa`

This class is used by the xMIPv6 class to get/set the configuration of the Multiple Care of Address Registration.

Despite, not specified in the RFS, this class holds the type of use to employ with the multiple care of addresses, as well as if in the deresgistraion process all the addresses are deregistered simultaneously.

2.2.4 Binding Update List (BUL)

C++ class: `BindingUpdateList`

Package: `networklayer.xmipv6`

BUL is employed by MN to manage the different bingings performed in the HA and the CNs.

The key is this structure includes now different fields and values as depicted in Table 2.2.

Field	**Description**	**Value**
BID	Binding Identification Number	$= -1$ or > 1, if using MCoA.
CoA	Care of Address	UNSPECIFIED_ADDRESS or a unicast and global IPv6 address
destBID	Address of the destination registration	IPv6 address of HA or IPv6 address of CNs.

Table 2.2: Key fields in BUL

2.2.5 Binding Cache (BC)

C++ class: `BindingCache`

Package: `networklayer.xmipv6`

BC is employed by HA and CNs to manage the different bingings performed by the MN.

The key is this structure includes now different fields and values as depicted in Table 2.3.

2.2.6 XMIPv6SM

C++ class: `XMIPv6SM`

Package: `networklayer.xmipv6`

This class implements a kind of state machine to control the xMIPv6 behaviour regarding the registration and deresgistraion of addresses.

It includes different parameters which include the preferred address, if it is returning home, or if mipv6 has been triggered.

Field	**Description**	**Value**
BID	Binding Identification Number	$= -1$ or > 1, if using MCoA.
CoA	Care of Address	UNSPECIFIED_ADDRESS or Home address of MN
destBID	Address of the destination registration	IPv6 address of HA or IPv6 address of CNs.

Table 2.3: Key fields in BC

Chapter 3

Configuration Parameters

This section presents the different configuration parameters and the respective possible values.

3.1 MCoA.ned

The class `MCoA` includes different parameters, these can be configured acoording to Table 3.1

Parameter	**Description**	**Value**
`m_prohibited`	Administratively forbid the registration of multiple CoA	true or false
`m_bulk_reg_prohibited`	If bulk registration is permitted	true or false
`mc_sim_home_and_foreign_prohibited`	If simultaneous home and foreign registration is allowed	true or false
`TypeUseMCoA`	Type of Use to give to the multiple care of addresses, if uses ALL at the same time, if uses the first one (SINGLEFIRST) or a single address chosen randomly (SINGLEROUNDROBIN)	ALL or SINGLEFIRST or SINGLEROUNDROBIN
`deregisterALL`	When deresgistering, if all the addresses are deresgistered at the same time or not	0 or 1 (default one-by-one)

Table 3.1: Configuration parameters of MCoA

3.2 Scenario Example

To be included in a next revision.

Chapter 4

MCoA++ Known Issues

As humans, software is not perfect, neither fully enjoyable by everyone. In this chapter knows issues are documented, on a release basis.

4.1 Release 2.0

4.1.1 Multiple Wireless Physical interfaces

Release 2.0 has been migrated to INETMANET to include support for multiple interfaces

4.2 Release 1.0

4.2.1 Multiple Wireless Physical interfaces

A Mobile Node does not support more then one *wlan* interface. The limitation comes from the physical model of *ieee802* implementation. On a next release we shall consider the solution in the INETMANET framework or INETHIP.

Status: Solved with version 2.0

4.2.2 Deregistration one by one

When using the type of use Register ALL, the deregistration one by one does not work.

The configuration parameters to generate this error are:

```
**.mCoA.TypeUseMCoA = "ALL"
**.mCoA.deregisterALL = 1
```

Some errors might happen, as depicted bellow:

```
 <!> Error in module (xMIPv6) MCoANetwork.MN[0].networkLayer.xMobileIPv6 (id=127):
User error: ASSERT: condition entry!=NULL false in function handleBULExpiry,
networklayer/xmipv6/xMIPv6.cc line 5158.
```

4.2.3 Employing the use of MIPv6

When MCoA is not enabled, the standard MIPV6 might not work.

The configuration parameters

```
**.mCoA.m_prohibited = true
```

Some errors might appear, as follows:

```
<!> Error in module (ICMPv6) MCoANetwork.Home_Agent.networkLayer.icmpv6 (id=91):
Gate ‘pingOut’ of compound module (cCompoundModule)
MCoANetwork.Home_Agent.networkLayer is not connected on the outside,
upon arrival of message (PingPayload)ping187
```

This error occurs when sending pings from the CN towards the MN. This issues has already been reported in the omnetgroups in the standard xMIPv6 package PingInxMIPv6 .

Chapter 5

INET MANET Modifications

This chapter presents some modifications that were performed to allow mCoA to work.

5.1 Values Modification

5.1.1 AUTOCONFIG_PERIOD

File: `linklayer/ethernet/EtherMAC.h`

The standard value 0.01 was modified to 1 to allow a greater delay in autocong ethernet messages. For instance without this modification it is not possible to have channels with the following characteristics:

```
channel internetline extends ned.DatarateChannel
{
    parameters:
        //delay = 0.1us;
        delay = 20ms;
}

Rb.pppg++ <--> internetline <--> Ra.pppg++;
```

5.1.2 UDP transport

File: `transport/UDP.cc`

```
 error("(%s)%s arrived from lower layer without control info"
```

To avoid the error presented above, the controlinfo kept in the `ctrl` variable is used in the comparion, as illustrated bellow:

```
  //else if (dynamic_cast<IPv6ControlInfo *>(udpPacket->getControlInfo())!=NULL)
  else if (dynamic_cast<IPv6ControlInfo *>(ctrl)!=NULL)
```

5.1.3 MIPv6

Recently Sent CoTI and HoTI

By reading RFC3775 (sec. 5.2.7), I think the methods `recentlySentCOTI` and `recentlySentHOTI` need to divide by 8 and not by 3. In order to tell if the CoTI or HoTI token has been sent recently.

5.2 Notifications

5.2.1 Returning Home

When the node returns home it sends a notification `NF_MIPv6_MN_RETURNED_HOME`, with the text “MIPv6 MN RETURNED HOME”