A Multiple Care-of Addresses model

Bruno Sousa[1],Marco Silva[1],
Kostas Pentikousis[2], Marilia Curado[1]
[1] Laboratory of Communications and Telematics,
Center for Informatics and Systems
University of Coimbra, Portugal
[2] Huawei Technologies,
European Research Center
Carnotstrasse 4, 10587 Berlin, Germany
bmsousa@dei.uc.pt

29 June 2011

Outline

1. Introduction
2. Motivation
3. mCoA++ Specification
4. Results
5. Conclusion

Introduction

Multiple Care-of Address (MCoA):

- is an extension of MIPv6.
- enables MIPv6 nodes the registration of diverse addresses (labelled as Care-of Addresses).
- specifies different modes of registration: bulk-registration (multiple addresses are conveyed in a single message), normal as MIPv6.

MCoA does not:

- Specify how addresses can be used (e.g., all at the same time)
- Does not include policies for flows.

Motivation

We are working in multihoming area, the possibility of using multiple addresses is required. MCoA fits in this case, if we consider IP mobility.

Why develop an implementation?

- Existent implementations are old (based on Internet Drafts);
- No free implementation in networks simulators (ns-2, ns-3, OMNeT++);
- **Need a base framework to test multihoming (my PhD) proposals**

Introduction

mCoA++ Implementation requirements:

- Keep compatibility with existent MIPv6 implementation.
- Public available and for free.
- Allow extensions.

mCoA++ first choices:

- *xMIPv6* model - an accurate implementation of MIPv6 in *OMNeT++*. It is also free and public available.
- *OMNeT++* is a network simulator with reputation in the academic community.
- Implemented in *C++*.

General details

MCoA++, is available at: `http://mcoa.dei.uc.pt`.

1. Version 1 - first implementation of MCoA.
   - Extends *xMIPv6*.
   - Based on *INET* framework (implementation of IP protocols in *OMNeT++*).
2. **Version 2** - second implementation of MCoA.
   - Based on *INETMANET* framework (adds support for multiple radios).
   - Bug fixes (as usual).

Implementation details - Addresses

Address usage in MCoA:

- *ALL* - all the addresses are used simultaneously.

```
//  Code from MCoAUDPBase::sendToUDPMCOA method
if (useMode == MCOA_TUN_ALL_ADR_ALL){
 std::vector<AdrInfo>::iterator it;
 for (it=adrsAvailable.begin(); it < adrsAvailable.end(); it++){
   cPacket *msg1 = msg->dup();
   EV << "MCoAUDPBase duplicating message for adr " << it->mSrc << endl;
   //Please note dup objects need to add controlinfo.
   sendToUDP(msg1, it->mSrc, srcPort, destAddr, destPort , true);
 }
}
```

- *SINGLE* selects a care of address following two approaches:

  1. In a round robin approach
     MCOA_TUN_ALL_ADR_SINGLE_RR.
  2. The first registered address
     MCOA_TUN_ALL_ADR_SINGLE_FIRST.

Implementation details - Notifications Network layer

- Node returned home

```
nb->fireChangeNotification(NF_MIPv6_MN_RETURNED_HOME, NULL);
```

- Routing Optimization finished

```
nb->fireChangeNotification(NF_MIPv6_RO_COMPLETED, NULL);
```

- New routes

```
nb->fireChangeNotification(NF_IPv6_ROUTE_ADDED, route);
```

- New tunnel

```
nb->fireChangeNotification(NF_IPv6_TUNNEL_ADDED, route);
```

- Handover occurred

```
nb->fireChangeNotification(NF_IPv6_HANDOVER_OCCURRED, NULL);
```

Implementation details - Notifications in Upper layers

Upper layers subscribe to events:

```
//code in MCoAUDPBase::startMCoAUDPBase method
nb = NotificationBoardAccess().get();
nb->subscribe(this, NF_IPv6_TUNNEL_ADDED);
nb->subscribe(this, NF_IPv6_TUNNEL_DELETED);
```

Process subscribed events:

```
//code in MCoAUDPBase::receiveChangeNotification method
if (category==NF_IPv6_TUNNEL_ADDED){
 EV << "NEW address obtained " << adrInfo << endl;
 if (!isAdrInVec(adraux)){
   adrsAvailable.push_back(adraux);
   lenAdrs++;
   if (!adraux.mSrc.isUnspecified()){
     adraux.sockID= MCoAUDPBase::bindToPort(localPort, adraux.mSrc);
   }
   EV << ''address " <<  adraux.mSrc << " added sucessfully '';
 }else{
   EV << "Address is already in vector not adding" << endl;
 }
```

Implementation details - Classes

New Classes were added:

- *MCoA* to configure settings for MCoA protocol, .e.g. if bulk registration mode is performed.
- Classes to act as key in data structures, *KeyMCoABind* and *KeyMCoADAD*.
- Classes to allow information exchange in events notification, such as *IPv6PrefAdr*, *IPv6TunAdr*.

Some classes were modified:

- *xMIPv6* main class where all the mobility procedures are included (e.g. *processBUMessage*, *processBAMessage*).
- *BindingUpdateList*, *BindingCache* modified to include new keys in respective data structures.

Implementation details - Messages

Implemented new Binding Status, e.g. *MCOA_NOTCOMPLETE*.
Implemented MCoA options:

- *Binding Identifier Mobility - BID* to identify bindings. In MIPv6 this was done by CoA.
- *MobilityBIDOptions* array of BID options.

Messages modified to include *MobilityBIDOptions*:

- *Binding Update* add also *Authorization Data Option*.
- *Binding Acknowledgment* add also *Authorization Data Option*.
- *Home Test Init* and *Home Test*.
- *Care Test Init* and *Care Test*.
- *Binding Refresh Request*.

Implementation details - Application Support

New class *MCoAUDPBase* to support MCoA facilities for UDP applications:

- Subscribes events e.g. *NF_IPv6_TUNNEL_ADDED*.
- Performs binding per Address and port *MCoAUDPBase::bindToPort(int port, IPvXAddress &srcAddr)*.
- Perform sending according address usage *MCoAUDPBase::sendToUDPMCOA(...)*.
- Act as a base application that others applications can extend.

Included Applications (extending *MCoAUDPBase)*:

- Video Streaming
- VoIP

Evaluation - Methodology

Goals of evaluation:

- Validate *mCoA++* implementation.
- Demonstrate usability of *mCoA++*.

Methodology

- VoIP and Video Applications.
- Include different speeds (3km/h, 30km/h).
- Include different types of failures: Handover-HO and Network-Net.
- Include different types of address usage.
- Handover Time: $T_{HO} = t_{regCR} - t_{ASSOC}$.

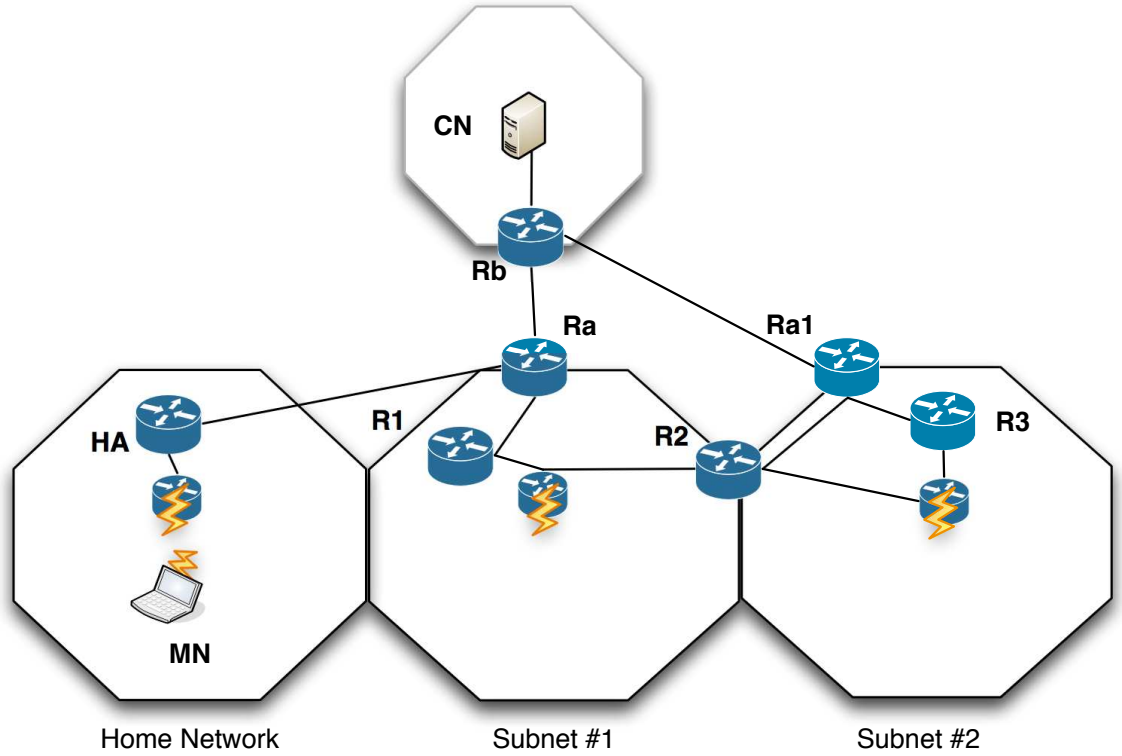
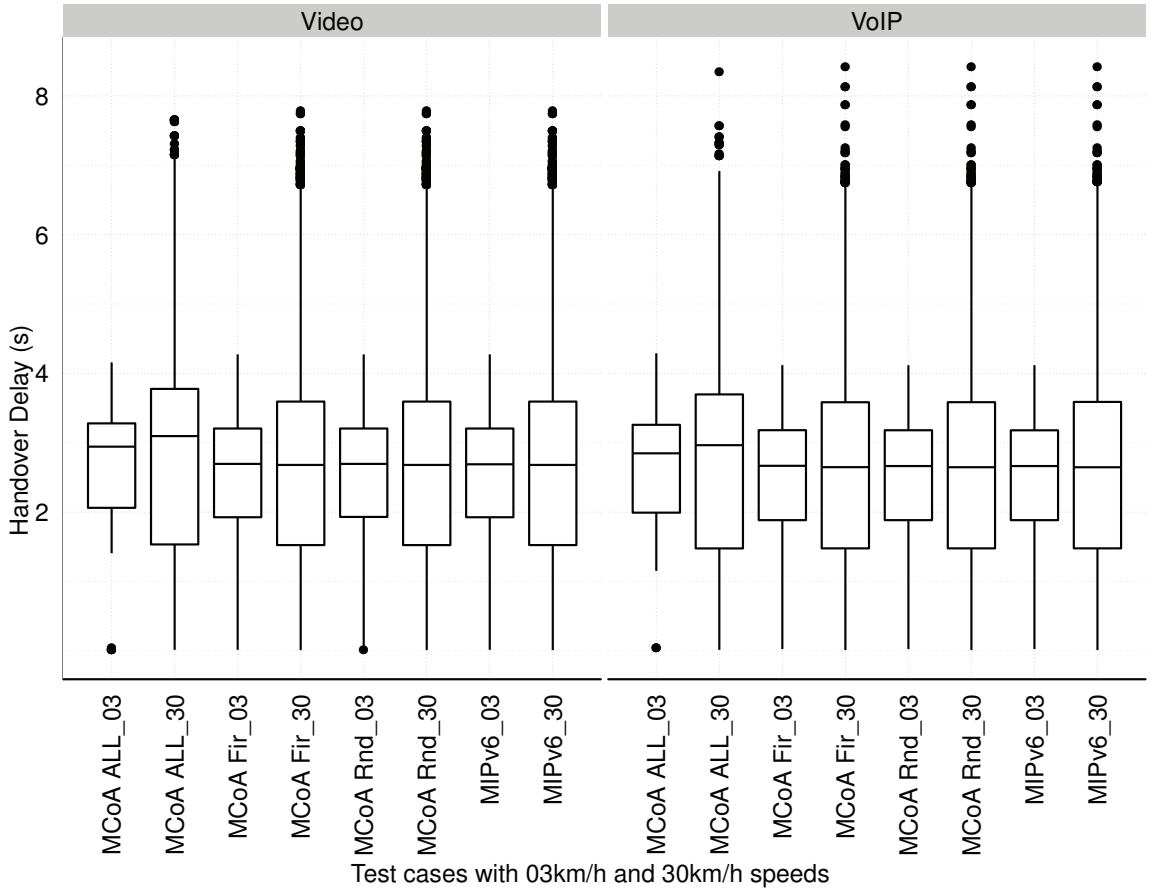


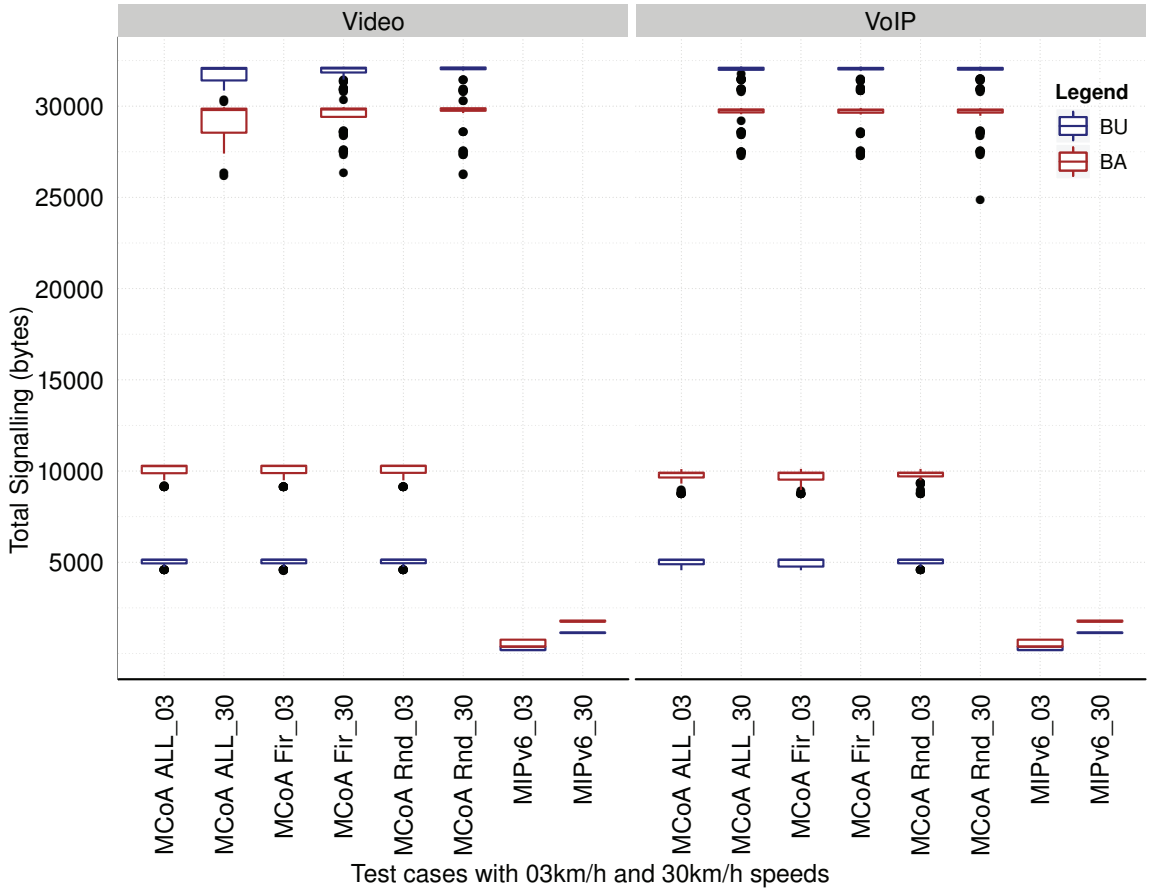
Figure 1: Simulation Scenario

Results - Handover Time



- MCoA single cases have the same values as MIPv6.

Results - Signalling Cost



- MCoA cases introduce more overhead than MIPv6.

Our Contributions

- *mCoA++* if freely available at `http://mcoa.dei.uc.pt`.
- *mCoA++* is useful for researchers interested in mobile IP networks and multiaccess.
- *mCoA++* is useful for evaluation of multimedia applications over mobile IP networks.
- *mCoA++* can be used as a base platform to integrate Mobile IP extensions.

Conclusion

- *mCoA++* includes MCoA functionalities.
- Current versions include support for UDP applications.
- MCoA is already working with *VoIPTool* (allows use of real audio data).

Next Developments:

- Integrate support for TCP applications.
- Extend MCoA with Flow Bindings support (RFC 6089).
- Introduce new address usage approaches.

Thank You