Quantitative Analysis of the Handover Process on Homogeneous and Heterogeneous Networks[1]

Victor Guimaraes da Silva
Dept. Systems and Computer Engineering
Carleton University, Ottawa, ON, Canada
victorguimaraesdasil@cmail.carleton.ca

***Abstract*—Due to the increasingly demand for mobile data, operators have been using different techniques to deal with this challenge. One of the factors that influence the overall network performance is the efficient use of the radio spectrum. One approach suggested by the 3rd Generation Partnership Project (3GPP) is the use of Heterogeneous Networks (HetNets) which assist on the reuse of the radio resources. Even though HetNets are able to improve the cell performance, it is still necessary to improve the control algorithms regarding the handover (HO) process because increasing the number of eNBs on a determined area can increase the amount of handovers and control messages which can bring significant amount of overhead which worsen the cell performance. The goal of this paper is to show a quantitative analysis of the handover process on HomNets and HetNets, specifically the amount of handovers.**

Keywords— LTE-Advanced, Handover, Heterogeneous networks, Time-to-trigger, DEVS, CD++.

I. Introduction

THE number of mobile broadband subscriptions is continuously growing and is predicted to reach 7.7 billion by 2021 [1]. Due to the demand for mobile data, it is important to use the radio spectrum efficiently since the amount of bandwidth available is limited. The use of heterogeneous networks (HetNet) is considered one of the effective methods to improve the capacity of cellular networks. HetNets are composed of various kinds of wireless access nodes with varied capacities. In the LTE-Advanced and 5G cellular networks, HetNets are a combination of macro-cells (MeNBs) and low-power nodes, for instance, pico-cells (PeNB), femto-cells, Remote Radio Heads (RRHs), and relay nodes. The CD++ toolkit was used for modeling and simulation (M&S). It provides an environment to execute DEVS and Cell-DEVS models. Different scenarios for homogeneous networks (HomNet) and HetNets under urban area setting using DEVS formalism were modeled in order to study the handover (HO) process on those networks, more specifically, the amount of HO.

The results showed that the number of HO on the HetNet scenario increased when using the current algorithm. Even though HetNets are able to improve the cell performance, it is still necessary to improve the control algorithms regarding the handover process because increasing the number of eNBs on a determined area can increase the amount of handovers and control messages which can bring significant amount of overhead which worsen the cell performance.

II. Background

In short, the conventional handover process is made to avoid any service interruption when a user is going out of a determined cell coverage. If the receiving power is not enough, the User Equipment will connect to another eNB.

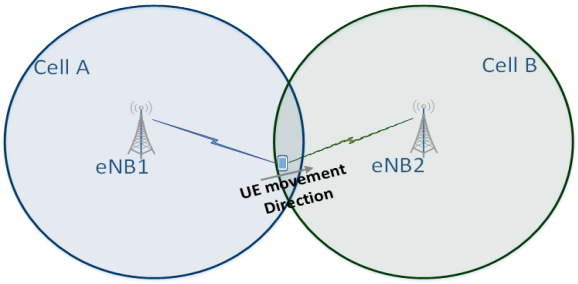

Fig. 1. High-level Handover Architecture

The User Equipment (UE) measures the Serving eNB`s receiving power (named RSRP in the 3GPP standard) every 40 ms. The UE also measures the second highest RSRP (Target eNB, or TeNB), periodically.

If the Serving eNB (SeNB) RSRP is lower than a threshold (in comparison to the TeNB receiving power, named Handover criteria) the UE takes five RSRP measures (one every 40 ms) and averages out the measurements. If the averaged RSRP meets the Handover criteria the UE sets a time-to-trigger (TTT) of 160 ms, which makes the UE wait that amount of time until measuring the RSRP again.

After waiting and measuring again, the UE sends (trigger) a Measurement Report (MR) to SeNB. After receiving the MR, SeNB checks the MR and takes HO decision. The SeNB will send HO request to TeNB and TeNB sends HO request acknowledgement (ACK) to SeNB. The SeNB sends RRC Connection Reconfiguration message (HO Command) to UE. Finally, UE sends RRC Connection Reconfiguration Complete message to TeNB (Connect to the TeNB).

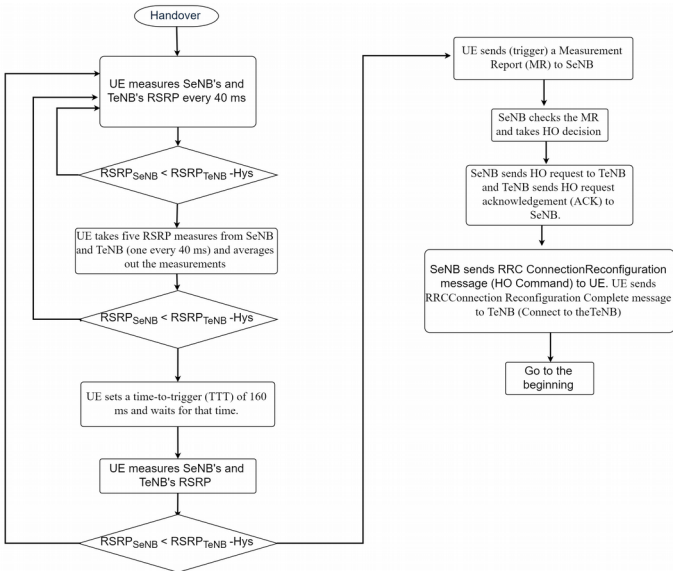

Fig. 2. Handover Algorithm

It is important to notice that the process previously described happens whether the UE is moving or not. Therefore, the UE will have a specified velocity and position. The UE position is calculated before the RSRP. In order to simplify the simulator, the UE has a uniform linear motion with constant velocity. The UE position is calculated every 40 ms.

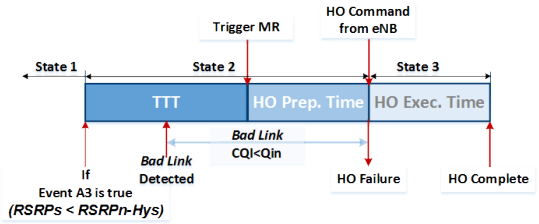

Fig. 3. Block Diagram of Handover States

III. Models

Two similar DEVS models were created in order to simulate the Handover process on HetNets and HomNets. The first simulator is called HetNet simulator. The basic components are three main coupled models: MeNB, PeNB, and UE. MeNB and PeNB stand for macro and pico cell, respectively, and UE stands for User Equipment. The top level coupled model is the HetNet geographic area which is composed of a number of cells. Each cell contains one MeNB, a determined number of PeNBs and many UEs.

Each coupled model, MeNB, PeNB and UE, is composed of two atomic models, a processor and a queue. The UE Queue is responsible for storing the messages received from the eNBs. A UE Queue has a ID. A UEProc has a ID, an initial position (currentX and currentY), a start time (sTime), an end time (etime), a final position (endX and endY), and a speed.

The UEProc is responsible for calculating the RSRP, the UE position, it is also responsible for requesting new messages stored on its Queue, processing the control messages received from the eNBs, and sending messages to the eNBs.

MeNB and PeNB have the same behaviour, the main difference is the Transmitted power, where MeNB has more power than PeNB. Both Queues store messages from UEs and other eNBs. A Queue has a ID. Both MeNBProc and PeNBProc are responsible for requesting new messages from the Queue, processing the messages and output other control messages to UEs and/or eNBs. A MeNBProc has a ID, a position (currentX and currentY), a frequency, and MeNBPower. A PeNBProc has a ID, a position (currentX and currentY), a frequency, and PeNBPower. The eNBs are linked to each other through a X2 Link and they are linked to the UEs through a Radio Link. Each MeNB and PeNB has two outputs (Out and X2out) and two inputs (In and X2in).

The MeNB and PeNB X2out ports are connected to other MeNBs and PeNBs X2In ports. The MeNB and PeNB Out ports are connected to UEs` In ports. The Req port is used to request new messages from the Queue.

The second simulator is called HomNet simulator. It is basically the same as a HetNet simulator. The only difference is that it does not have PeNBs since it is a homogeneous network.

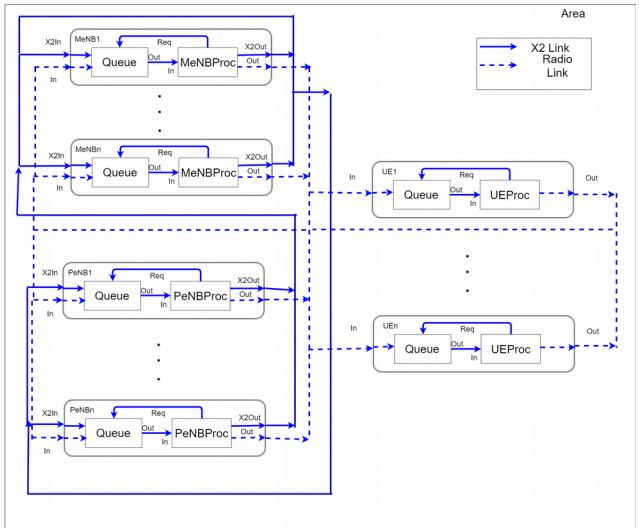

Fig. 4. Structure of HetNet simulator

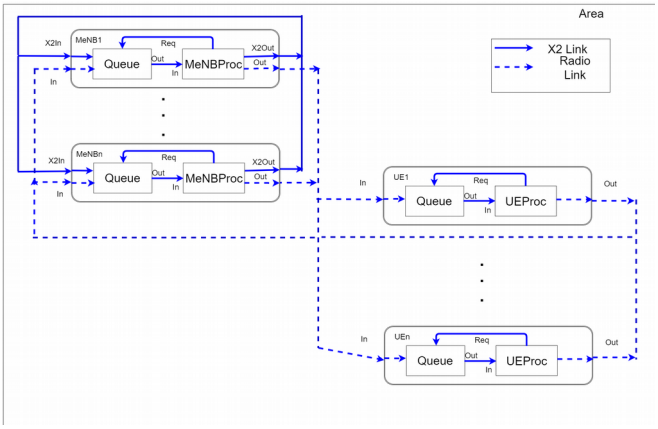

Fig. 5. Structure of HomNet simulator

DEVS formal specification for each of the coupled and atomic models:

$M_{HetNet} = \langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, select \rangle$
where,

$X = \emptyset;$
$Y = \emptyset;$
$D = \{(UE_i, MeNB_j, PeNB_k) / i, j, \text{and } k \in \mathbb{N}\}$
$M_d = \{M_{UE}, M_{meNB}, M_{PeNB}\}$
(where M_{UE}, M_{peNB}, and M_{MeNB} are coupled models);
$EIC = \emptyset$
$EOC = \emptyset$
$[(MeNB_i, X2Out), (MeNB_j, X2In) / i, j \in \mathbb{N}, i \neq j, \text{i.e. each MeNB does not connect to itself}];$

each PeNB connects to only one MeNB.
IC = { $[(MeNB_i, Out), (UE_k, In) / i,k \in \mathbb{N}];$
$[(UE_k, Out), (MeNB_i, In) / i,k \in \mathbb{N}];$
$[(PeNB_i, Out), (UE_k, In) / i,k \in \mathbb{N}];$
$[(UE_k, Out), (PeNB_i, In) / i,k \in \mathbb{N}]$ }
$select$ = {UE, MeNB ,PeNB}.

$M_{HomNet} = \langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, select \rangle$
where,

$X = \emptyset;$
$Y = \emptyset;$
$D = \{(UE_i, MeNB_j) / i,j \in \mathbb{N}\}$
$M_d = \{M_{UE}, M_{MeNB}\}$ (where both M_{UE} and M_{MeNB} are coupled models);
$EIC = \emptyset$
$EOC = \emptyset$
$[(MeNB_i, X2Out), (MeNB_j, X2In) / i, j \in \mathbb{N}, i \neq j, \text{i.e. each MeNB does not connect to itself}];$

$IC \subseteq \{[(MeNB_i, Out), (UE_k, In) / i,k \in \mathbb{N}]; [(UE_k, Out), (MeNB_i, In) / i,k \in \mathbb{N}]\}$

$select$ = {UE, MeNB }.

$M_{UE} = \langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, select \rangle$
where,

$X = \{(In,N)\}$;
$Y = \{(Out,N)\}$;
$D = \{Queue, UEProc\}$
$M_d = \{M_{Queue}, M_{UEProc}\}$ (where both M_{Queue} and M_{UEProc} are atomic models);
$EIC = \{((Self, In), (Queue, In)),\}$;
$EOC = \{((UEProc, Out), (Self, Out))\}$;
$IC \subseteq \{[(Queue, Out), (UEProc, In)]\}$
$select = \{UEProc, Queue\}$.

$M_{MeNB} = \langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, select \rangle$
where,

$X = \{(In, N),(X2In, N)\}$;
$Y = \{(Out, N), (X2Out, N)\}$;
$D = \{Queue, MeNBProc\}$
$M_d = \{M_{Queue}, M_{MeNBProc}\}$
(where both M_{Queue} and $M_{MeNBProc}$ are atomic models);
$EIC = \{((Self, In), (Queue, In)), ((Self, X2In), (Queue, X2In))\}$;
$EOC = \{((MeNBProc, Out), (Self, Out)),$
$((MeNBProc, X2Out), (Self, X2Out))\}$;
$IC \subseteq \{[(Queue,Out),(MeNBProc,In)];[(MeNBProc,Req),(Queue,Req)]\}$
$select = \{MeNBProc, Queue\}$.

$M_{PeNB} = \langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, select \rangle$
where,

$X = \{(In, N),(X2In, N)\}$;
$Y = \{(Out, N), (X2Out, N)\}$;
$D = \{Queue, PeNBProc\}$
$M_d = \{M_{Queue}, M_{PeNBProc}\}$
(where both M_{Queue} and $M_{PeNBProc}$ are atomic models);
$EIC = \{((Self, In), (Queue, In)), ((Self, X2In), (Queue, X2In))\}$;
$EOC = \{((PeNBProc, Out), (Self, Out)),$
$((PeNBProc, X2Out), (Self, X2Out))\}$;
$IC \subseteq \{[(Queue,Out),(PeNBProc,In)];[(PeNBProc,Req),(Queue,Req)]\}$
$select = \{PeNBProc, Queue\}$.

$Queue = \langle S, X, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$
where,
$S = \{ status \in \{Passive,Active\}, state \in \{Idle,Push,Pop,Send\} \}$;
$X = \{In, Req\}$;
$Y = \{Out\}$;

$\delta_{int}(s)$ {
switch (state){
case Idle:
if (Request == 1) state = Send;
else passivate();
break;
case Push:
state = Idle;
break;
case Send:
Request = 0;
passivate();
break;

case Pop:
state = Idle;
break;
}
}

$\delta_{ext}(s,e,x)$ {
if(x.port == In || x.port == X2in) {
if((x.destinationID == this->id || x.destinationID == 9999) && (x.sourceID != this->id){
add(x.value, elementsO);
state = Push;
}
else{ //message input port is Req
// 1 means add message to the queue.
//1-queue is empty
//2-queue is not empty
if (x.value == 1) {
if(sizeOf(elementsO) <= 0){
Request = 1;
passivate();
}
else {
Request = 1;
state = Send;
status = Active;
ta(status, ProcessTime);
}
}
else if(x.value == 2 && sizeOf(elementsO)>0){
// 2 means delete received message from the queue
//delete from the queue
state = Pop;
pop_front(elementsO);
Request = 0;
status = Active;
ta(status, ProcessTime);
}
}
}
}

```
λ(s) {
   if ((state == Send) && (Request == 1)) {
      sendOutput(msg.time(), Out, NULL, front(elementsO));
   }
}
```

UEProc = <S, X, Y, δ_{int}, δ_{ext}, λ, ta>

$S = \{ status \in \{ Passive, Active \}, state \in \{ Idle, SendPack, RecPack \} \};$

```
X = {In};
Y = {Out, Req};
δint(s) {
   switch (state){
      case Idle:
         state = SendPack;
         break;
      case RecPack:
         state = Idle;
         break;
      case SendPack:
         state = RecPack;
         status = Active;
         ta(status, ProcessTime);
         break;
      }
}
```

```
δext(s,e,x) {
   if (x.port==In) {
      state = SendPack;
      if (x.message == CTRL_MeNB) {
         //calculates the Serving eNB
      }
      else if (x.message == HO_CMD) {
         //TeNB becomes SeNB i.e. concludes the HO process.
      }
   }
   status = Active;
   ta(status, ProcessTime);
}
```

```
λ(s) {
   if (state == Idle){
      sendOutput(msg.time(), Req, 1, NULL);
      // 1 means UE processor requesting to queue to send him message
   }
   else if (state == SendPack) {
      //calculates RSRP and the UE position.
      //If HO criteria is met, it sends the RSRP to SeNB.
   }
}
```

MeNBProc = <S, X, Y, δ_{int}, δ_{ext}, λ, ta>

$S = \{ status \in \{ Passive, Active \}, state \in \{ Idle, SendPack, RecPack \} \};$

```
X = {In};
Y = {X2Out,Out, Req};
δint(s) {
   switch (state){
      case Idle:
         state = RecPack;
         status = Active;
         ta(status, ProcessTime);
         break;
      case RecPack:
         state = Idle;
         break;
      case SendPack:
         state = RecPack;
         status = Active;
         ta(status, ProcessTime);
         break;
   }
}
```

```
δext(s,e,x) {
  if (x.port==In) {
    // Received a new packet from Queue and will process and sent to others.
    state = SendPack;
    //For messages received from the UE
    if (x.message == CSI_FEEDBACK) {
      //Extract CSI Feedback.
      //Set flags that define what kind of message will be sent.
    }
    //Received MR from the UE
    else if (x.message == MES_REP) {
    //Extract Measurement Report.
    //Set flags that define what kind of message will be sent.
    }
    //Received HO_REQ from the  eNB
    else if (x.message == HO_REQ) {
      //Extract HO Request.
      //Set flags that define what kind of message will be sent.
    }
     //Received HO_ACK from the  eNB
     else if (x.message == HO_ACK) {
       //Received HO Acknowledgement.
       //Set flags that define what kind of message will be sent.
    }
    else if (x.message == CTRL_MeNB) {
      //Received Control Command from other eNBs.
      //extract the information received from the Control Command.
    }
  }
}
```

```
λ(s) {
  if (ctrlCmdFlag == true){
    // All the Macro eNBs share the system information and also send to UEs
    CTRLcommand = new CTRLCmd(this->id, 9999, posx, posy, f, p);
    sendOutput(msg.time(), Out, NULL, CTRLcommand);
    sendOutput(msg.time(), X2out, NULL, CTRLcommand);
    ctrlCmdFlag = false;
  }
  if (MRFlags[MRsourceUEid] == true){
    // All the Macro eNBs share the system information and also send to UEs
    //It sends HO Request
    sendOutput(msg.time(), X2out, NULL, HOreq);
    MRFlags[MRsourceUEid] = false;
  }
  if (HOReqFlags[HOReqSourceUEid] == true){
     //It sends HO Request Acknowledgement
     sendOutput(msg.time(), X2out, NULL, HOReqAcknowledgement);
     HOReqFlags[HOReqSourceUEid] = false;
  }

  if (HOAckFlags[HOAcksourceUEid] == true){
    //It sends RRC Command.
    sendOutput(msg.time(), Out, NULL, RRCcmd);
    HOAckFlags[HOAcksourceUEid] = false;
  }
  if (state == Idle){
    sendOutput(msg.time(), Req, 1, NULL);
    // 1 means UE processor requesting to queue to send him message
  }
  else if (state == SendPack) {
    // It sends required messages to others (eNBs or UEs) and
    //after sending the message it sends request to Queue to delete the received message
    sendOutput(msg.time(), Req, 2, NULL);
  }
}
```

Queue = <S, X, Y, δ_{int}, δ_{ext}, λ, ta> (MeNB)

where,

$S = \{ status \in \{Passive, Active\}, state \in \{Idle, Push, Pop, Send\} \}$;

X = {X2In, In, Req};

Y = {Out};

```
δint(s) {
  switch (state){
  case Idle:
    if (Request == 1) state = Send;
    else passivate();
    break;
  case Push:
    state = Idle;
    break;
  case Send:
    Request = 0;
    passivate();
    break;
  case Pop:
    state = Idle;
    break;
  }
}
```

```
δext(s,e,x) {
   if(x.port == In || x.port == X2in) {
      if((x.destinationID == this->id || x.destinationID == 9999) && (x.sourceID != this->id){
         add(x.value, elementsO);
         state = Push;
      }
      else{     //message input port is Req
               // 1 means add message to the queue.
        //1-queue is empty
        //2-queue is not empty
        if (x.value == 1) {
           if(sizeOf(elementsO) <= 0 ){
              Request = 1;
              passivate();
           }
           else {
             Request = 1;
             state = Send;
             status = Active;
             ta(status, ProcessTime);
           }
        }
        else if(x.value == 2 && sizeOf(elementsO)>0){
            // 2 means delete received message from the queue
            //delete from the queue
            state = Pop;
            pop_front(elementsO);
            Request = 0;
            status = Active;
            ta(status, ProcessTime);
         }

      }
   }
}
```

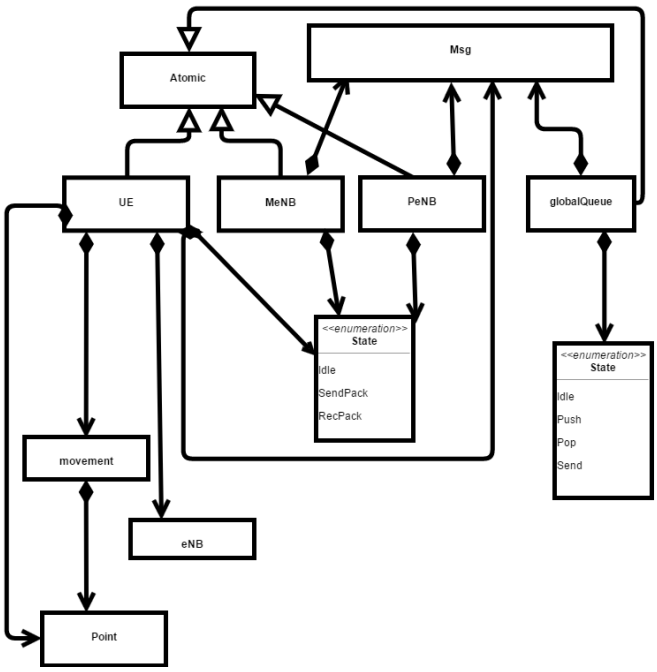

Fig. 6. UML model of HetNet simulator with the main components of the software

```
λ(s) {
   if ((state == Send) && (Request == 1)) {
      sendOutput(msg.time(), Out, NULL, front(elementsO));
   }
}
```

The UE.cpp and UE.h use the class movement.cpp in order to calculate the UE`s position according to the simulation time. From the position, the received power can be calculated. Thereafter, the HO procedure started to be tackled on the UEProc and MeNBProc sides.

Since the HO procedure is based on the exchange of messages, the Msg.cpp and Msg.h were used to assist on the process. The classes UE.cpp and MeNB.cpp made use of Msg.cpp class.

In order to create the simulation scenarios, a script made in c++ was created to generate the ma files. It is called maFileGen. It generates ma files according to what is defined by the user inside maFileGen.cpp.

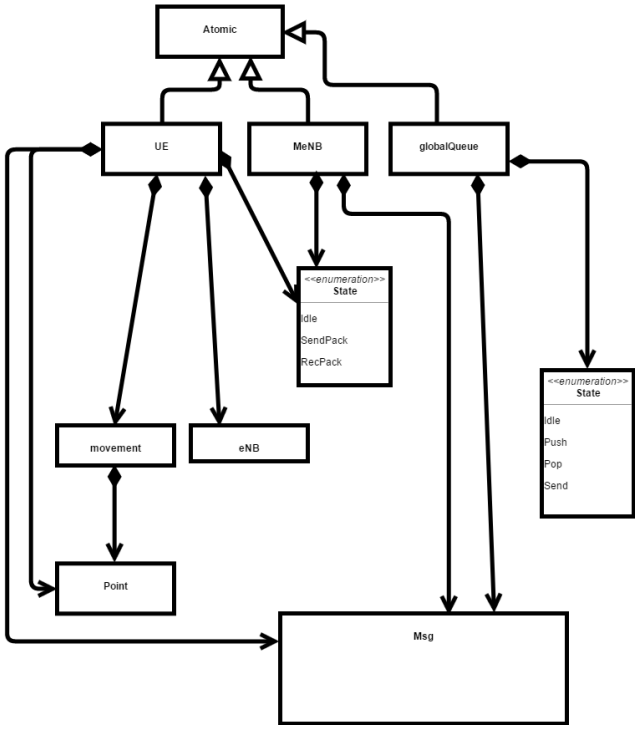

Fig. 7. UML model of HomNet simulator with the main components of the software

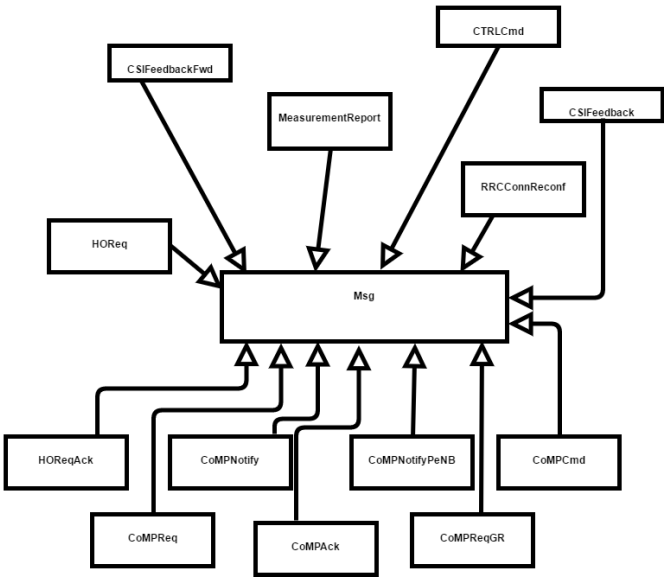

Fig. 8. Extension of the UML model for the class Msg.

IV. Simulation assumptions

Figure 9 shows the simplified network architectures of sample simulation scenarios that were used to analyze the handover performance based on different parameters suggested in 3GPP standard for LTE and LTE-Advanced cellular networks. To do that, a series of simulations on this model was run, based on the initial conditions summarized in table 1 [4]. The cell radius and antenna gain parameters were chosen to align with the specifications outlined in LTE release 12. As suggested in [3], the frequency has been set to 2000 MHz for macro eNBs, and 3500 MHz for Pico eNBs, the macro cell radius has been set to 500 m, and the antenna gain has been set to 12 dBi for the macro eNB, 5 dBi for pico eNB and 0 dBi for the UE.

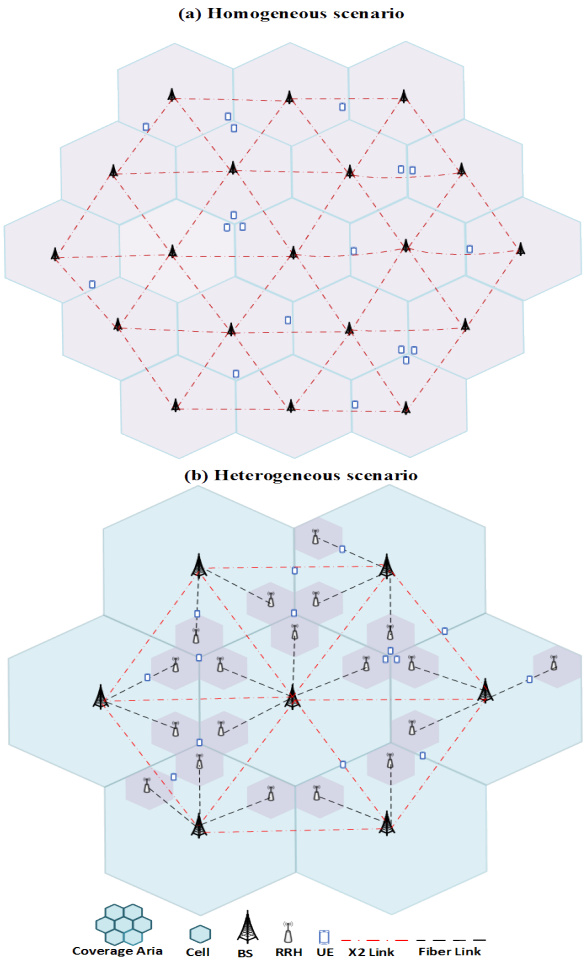

Fig. 9. Network Architectures

Parameters	Values
Number of macro eNBs	7
Number of Pico eNBs	18 in each macro cell
Number of UEs	100, 200, 500
UE Distribution	Uniform: randomly into the macro cell area
Frequency	2000 MHz, 3500 MHz
Macro eNB Transmit power	43 dBm
Small eNB Transmit Power	30 dBm
Cell Radius	500 m
Antenna gain	12 dBi (Macro eNB), 05 dBi (Pico eNB) and 0 dBi (UEs)
MCL	70 dB
LogF	10 dB
RSRP Sample	Every 40 ms
TTT (ms)	160
A3 offset	3 dB
CoMP Threshold	6 dB
UE speed (km/h)	3, 10, 20, 30
Handover preparation time	50 ms

Table 1. Simulation assumptions

In the simulation scenarios, cells are considered to be macro cells and pico cells in an urban area. A typical transmission power for a macro eNB is normally between 43 dBm to 48dBm and between 23 dBm to 30 dBm for a pico [4] eNB. Therefore, the transmit power for a MeNB was set to 43dBm and for a PeNB 30 dBm. The propagation model for macro cell is considered based on 3GPP standard in [2] and [3] as follows:

Macro Cell: $128.1 + 37.6\log_{10}(d)$ (1)

Pico Cell: $147 + 36.7\log_{10}(d)$ (2)

Where d is the distance between UE and BS.

V. RESULTS

The file resultCount.txt shows the number of HO. The files logfile.txt and receivedMsg.txt were created to ensure the correctness of the results since they show the information concerning received power, UE position, etc. The follow text shows a bit of those files from the result of the first simulation scenario.

From logfile.txt:

@UE: 3591 eNB ID: 1 RECEIVED THE CSI_FEEDBACK FROM UE ID: 3591
@UE: 3591 Verifying HO msg.time().asMsecs() 59680 TRAVELED DISTANCE: 497.333 metres at position: (1893, 1893) Total distance to be done: 707.107 metres ActualServingID: 1 ActualServingPower: -71 NewServingID: 2 NewServingPower: -68

@UE: 3591 Verifying HO msg.time().asMsecs() 59720 TRAVELED DISTANCE: 497.667 metres at position: (1893, 1893) Total distance to be done: 707.107 metres ActualServingID: 1 ActualServingPower: -71 NewServingID: 2 NewServingPower: -68

@UE: 3591 Verifying HO msg.time().asMsecs() 59760 TRAVELED DISTANCE: 498 metres at position: (1893, 1893) Total distance to be done: 707.107 metres ActualServingID: 1 ActualServingPower: -71 NewServingID: 2 NewServingPower: -68

@UE: 3591 Verifying HO msg.time().asMsecs() 59800 TRAVELED DISTANCE: 498.333 metres at position: (1893, 1893) Total distance to be done: 707.107 metres ActualServingID: 1 ActualServingPower: -71 NewServingID: 2 NewServingPower: -68

@UE: 3591 Verifying HO msg.time().asMsecs() 59840 TRAVELED DISTANCE: 498.667 metres at position: (1894, 1894) Total distance to be done: 707.107 metres ActualServingID: 1 ActualServingPower: -71 NewServingID: 2 NewServingPower: -68

@UE: 3591 TTT msg.time().asMsecs() 59840 TRAVELED DISTANCE: 498.667 metres at position: (1894, 1894) Total distance to be done: 707.107 metres ActualServingID: 1 ActualServingPower: -71 NewServingID: 2 NewServingPower: -68

@UE: 3591 eNB ID: 1 RECEIVED THE MR FROM UE ID: 3591 MR->getTargetID() 2
MRtargedID 2
@UE: 3591 eNB ID: 2 RECEIVED THE HO_REQ FROM eNB ID: 1 HOReqtargedID 2
@UE: 3591 eNB ID: 1 RECEIVED THE HO_ACK FROM eNB ID: 2 HOAcktargedID 2
UE ID: 3591 RECEIVED THE HO_CMD FROM eNB ID: 1 TeNB 2

UE: 3591 reached final Position 707 metres

From logfile.txt:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%@UE 3591 CSIFeedback time13

UE Speed	Homogeneous Network	HetNet With Same frequency for Macro and Pico	HetNet With Different frequency for Macro and Pico
3	39	128	110
10	239	552	405
20	420	929	668
30	469	1059	776

Table 2. Simulation results

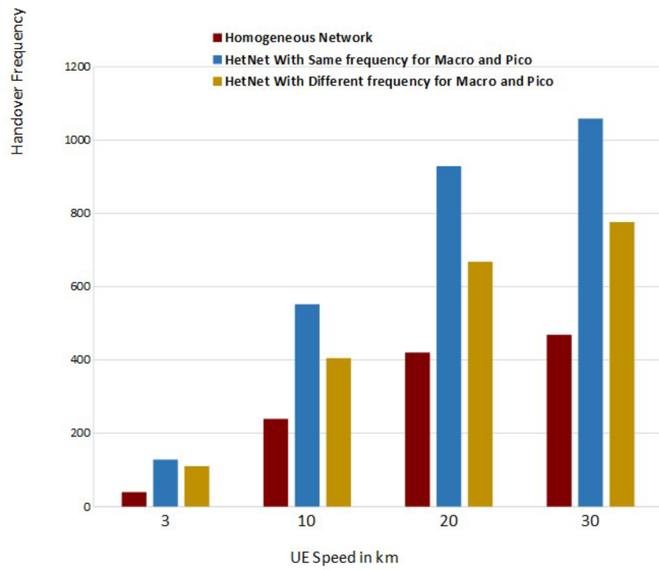

Fig. 10. Graph with the results

VI. Conclusions

The results show that the number of handovers on HetNet increased when using the current algorithm what was expected since the increased number of transmitters in a same cell area increases the chance of have HO.

Moreover, the UE speed plays a big role on the HO process because the UE will pass through more eNBs and if there is enough time for processing the RSRP, it may connect to different eNBs more often.

The amount of handover is one of the indicators of the network performance. Another parameter that indicates the efficiency of the network is the handover failure (HOF) rate. Therefore, it is also important to analyze that parameter, and to do so, a HOF model need to be implemented to better study the cell performance, which should be done in future works.

Also, other parameters such as TTT and Handover preparation time could be simulated with different values in order to find an optimum configuration.

Finally, the increased number of Handover will cause message overhead and will worsen the network performance. Therefore it is needed to develop a more efficient algorithm for the handover process.

REFERENCES

[1] Ericsson, "Ericsson Mobility Report," 2016 [Online]. Available: https://www.ericsson.com/res/docs/2016/ericssonmobility-report-2016.pdf. [Accessed 26 November 2016].

[2] "3GPP TS 36.814".

[3] "3GPP TR 36.842".

[4] ETSI, "3GPP TR 36.942 version 13.0.0 Release 13," 01 2016. [Online]. Available: http://www.etsi.org/deliver/etsi_tr/136900_136999/136942/13.00.00_60/tr_136942v130000p.pdf. [Accessed February 2016].