/**
****
 *
 *   File: readme.asn
 *
 *   Author: Adnan Abu-Mahfouz
 *
 *   Date: March 2012
 *
 *   Description: Localisation system in wireless sensor
networks using ns-2

****
*********/

/**
****
*****/
1. Introduction:
ns-2 contains several flexible features that encourage researchers to use
ns-2 to investigate the characteristics of wireless sensor networks (WSNs).
However, to implement and evaluate localisation algorithms, the current ns-
2 version (ns-2.34) should be extended and new modules should be added.

In this file, we will show how to add the new modules and also to modify
some of the existing files. The following article explains this extension:

A. M. Abu-Mahfouz and G. P. Hancke, GP., â€œns-2 extension to simulate
localization system in wireless sensor networks,â€ In Proceedings of the
IEEE Africon 2011 conference, 13-15 September , Livingstone, Zambia, 2011,
pp. 01-07. http://researchspace.csir.co.za/dspace/handle/10204/5559

The reader can also be referred to the following PhD thesis, for further
details about the localisation system in general (chapter two) and the
implementation (chapter three).

Abu-Mahfouz, AMI 2011, Accurate and efficient localisation in wireless
sensor networks using a best-reference selection, PhD thesis, University of
Pretoria, Pretoria, http://upetd.up.ac.za/thesis/available/etd-10122011-
124534/

Please note that, I will refer to directory /ns-allinone-2.34/ns-2.34 as
~ns

/**
****
*****/

2. Guidelines for extending ns-2:
After extracting the archive file named â€œns2_localisationâ€ you will get
the following contents:
        - location directory: which contains the new files
        - modified directory: which contains the modified files
        - tcl-examples: which contains a few tcl example files that using
the new localisation system
        - installation guidelines: shows how to install ns-2, if you did
not do that yet.
        - readme.asn file: the current file

Follow these steps to be able to simulate localisation system:

a. If you did not install ns-2 yet, follow the instructions shown in the â€œinstallation guidelinesâ€ to install it. Make sure it is working probably.

b. Copy the directory â€œlocationâ€ to ~ns.

c. If you have just installed a fresh copy of ns-2, you can simple copy the files in the â€œmodifiedâ€ directory to where they should be. You may make a backup of the original files (e.g. before replacing â€œpacket.hâ€ you can simply rename it as packet_old.h, then copy packet.h from â€œmodifiedâ€ directory to â€œ~ns/common/â€. The following table shows the modified files and where they should be copied.

Directory	Modified files
~ns/common/	packet.h
~ns/common/	mobilenode.h
~ns/common/	mobilenode.cc
~ns/common/	location.h
~ns/tcl/lib/	ns-packet.tcl
~ns/tcl/lib/	ns-default.tcl
~ns/tcl/lib/	ns-lib.tcl
~ns/tcl/lib/	ns-node.tcl
~ns/tcl/lib/	ns-namsupp.tcl

If you have already made some modifications into the ns-2 files, then you should make the new modifications by yourself as shown below (Section 5 â€œModified filesâ€).

d. Open "~ns/Makefile" and add the following:

Under INCLUDES= add the new directory

```
-I./location
```

Under OBJ_STL add the corresponding object files names of the new modules

```
location/locationrequest.o \
location/locationresponse.o \
location/locationdiscovery.o \
location/mmse.o \
location/position.o \
location/nearestposition.o \
location/refineposition.o
```

e. OPTIONAL (this step can be done later after you successfully test the new localisation system):

i. In order to stop the continuous printing of the receiving power at the terminal, which increasing the running time especially if you would like to run the simulation for long time, you can simply do the following:

- Open the file ~ns/mobile/propogation.cc
- Within the FreeSpace::Pr( ) function, comment the following line

```
// printf("%lf: d: %lf, Pr: %e\n", Scheduler::instance().clock(), d, Pr);
```

ii. In order to stop logging the energy of the node into the trace file, do the following:

- Open the file ~ns/common/mobilenode.cc

- Comment the entire body of log_energy() function

f. Open a terminal in Accessories/Terminal. Then cd ns-allinone-2.34/ns-2.34, then run make. (Note: if you get some errors just run make clean then make)

/**
****
*****/

3. Simulating localisation system
Using the ns-2 extension does not require new knowledge or writing a specific code to run the simulator. Normal users who have the basic knowledge to run a simple wireless network using ns-2 are able to write a simple Tcl script to simulate the proposed localization system. However, if you find any difficulty to understand how localisation system has been configured and used in these files you can refer to the two references mentioned early. If you are new to ns-2 you can refer to the following reference for some tutorials about configuring wireless networks.

"The network simulator - ns-2," 12 February 2010, http://nsnam.isi.edu/nsnam/index.php/User_Information.

Six tcl examples can be found under the directory â€œtcl-examplesâ€. At the beginning of each files you can find how to run it.
To execute any of the first three files loc1.tcl, loc2.tcl or loc3.tcl you can use the following command syntax:
ns filename METHOD
METHOD: is representing the localisation method that will be used in the estimation, an integer value of 1,2 or 3 can be used to consider one of the following localisation methods: general(1), nearest3(2), refine(3)

For example, to execute the first file using the general localisation algorithm, you can run the following command:
ns loc1.tcl 1

To execute the other tcl files; loc4.tcl, loc5.tcl or loc6.tcl, one extra argument should be included in the command syntax:
ns filename SEED METHOD
SEED: is the seed value used by RNG (0, 1, 2,...). This argument is used to specify where the nodes will be distributed, which allows evaluating the localisation algorithm considering different nodes distribution. If you comparing different localisation algorithms the same value(s) of SEED should be used (except the value of 0).

For example, to execute the fifth file using the nearest localisation algorithm, you can run the following command:
ns loc5.tcl 1 2

/**
****
*****/

4. Traced location data:
The format of the traced data will be as follows:

L 13.845754 3 0.499617 5 0.725425 45.353581 44.919124
| time node energy | error X and Y coordination for estimated position
| |

L is for Location   # of used references

You can simply use the grep command to filter only the location information form the trace file. Assuming the name of the trace file is â€œlocation.trâ€, then the extracted location information can be store in the file named â€œresultâ€ by using the following command:
grep "^L" location.tr > result

This command means filter only those lines that begin with the letter L

/**
****
*****/

5. Modified files:
This section shows the modifications that have been done to the existing files. You can also refer to the files in â€œmodifiedâ€ directory to see exactly how these modifications have been done.

/*========~ns/common/packet.h==
==*/

Add:
```
        #define HDR_LOCREQ(p)  (hdr_locreq::access(p))
        //location request
        #define HDR_LOCRES(p)  (hdr_locres::access(p))
        //location response
```

Before the PT_NTYPE add the following two packet types.
```
        // Location request and response packets
        static const packet_t PT_LOCREQ = 62; //where 62 is the current
value of PT_NTYPE
        static const packet_t PT_LOCRES = 63;
```

Modify the value of PT_NTYPE
```
        static packet_tPT_NTYPE = 64; // This MUST be the LAST one
```

Inside the function initName() of class p_info add:
```
        // location request and response
        name_[PT_LOCREQ] = "locreq";
        name_[PT_LOCRES] = "locres";
```
/*==
==*/

/*========~ns/common/mobilenode.h==
==*/

Add the following:
```
        inline Topography* get_topography() { return T_;}

        // log the location information
        void log_loc(double, int, double, double);
```

/*==
==*/

/*========~ns/common/mobilenode.cc===
==*/

Add the following function to log the location information

```
void
MobileNode::log_loc(double error_, int no_ref_, double x, double y)
{
        if (!log_target_)
                return;
        Scheduler &s = Scheduler::instance();

        sprintf(log_target_->pt_->buffer(),"L %f %d %f %d %f %f %f",
                                s.clock(),
                                address_,
                                energy_model_->energy(),
                                no_ref_,
                                error_,
                                x,
                                y);

        log_target_->pt_->dump();
}

/*==========================================================================
==============================================*/

/*========~ns/common/location.h=============================================
==============================================*/

Add a null constructor
        Location(): X(0), Y(0), Z(0) {}

Add the getters and setters of individual parameters
        virtual double getx() {return X;}
        virtual double gety() {return Y;}
        virtual double getz() {return Z;}

        virtual void setx(double x) {X = x;}
        virtual void sety(double y) {Y = y;}
        virtual void setz(double z) {Z = z;}

Add the following function to check if two locations are the same
        virtual int is_equal(Location *loc)
        {
                if((X == loc->X) && (Y == loc->Y) && (Z == loc->Z))
                        return 1;
                else
                        return 0;
        }

Add the following function to estimate the distance between two locations
        virtual double distance(Location *loc)
        {
                double dx = X - loc->X;
                double dy = Y - loc->Y;
                double dz = Z - loc->Z;

                return sqrt( dx * dx + dy * dy + dz * dz);
        }
/*==========================================================================
==============================================*/

/*========~ns/tcl/lib/ns-
packet.tcl================================================================
=======================*/
```

```
Before the end of foreach prot {} add
        LocReq  #Location Request
        LocRes  #Location Response

/*==========================================================================
==============================================*/

/*========~ns/tcl/lib/ns-
default.tcl=============================================================
=======================*/

Add the following to the end of the file:
        #Location discovery
        Agent/LocReq set packetSize_ 100
        Agent/LocRes set packetSize_ 100
        Application/LocDiscovery set distanceError_ 0 ;# considering
distance measurement   error, enable(1) or disable (0)
        Application/LocDiscovery set random_ 0
        Application/LocDiscovery set reqFreq_ 5.0            ;# every 5.0
sec
        Application/LocDiscovery set maxRequests_ 268435456   ;# 0x10000000
        Application/LocDiscovery set showColor_ 1     ;# Node colouring,
enable(1) or disable(0)
        Application/LocDiscovery set subset_ 1        ;# Localisation
algorithm, general method (1), nearest3(2), , refinement(3)
        Simulator set attribute_ ""
/*==========================================================================
==============================================*/

/*========~ns/tcl/lib/ns-
lib.tcl=================================================================
=======================*/

Add to # New node structure
        # added here to specify node attribute (beacon, reference or
unknown)
        #                       -attribute BEACON/REFERECE/UNKNOWN

Add the following instproc
        #used for location discovery process
        Simulator instproc attribute {val} {$self set attribute_ $val}

Within Simulator instproc node-config args, add the following instvar
        #Location discovery
        attribute_

Within Simulator instproc create-wireless-node args, add the following
instvar
        #Location discovery
        attribute_

Also add:
        #location discovery
        if [info exists attribute_] {
                $node set nodeAttribute_ $attribute_
        }
/*==========================================================================
==============================================*/
```

```
/*========~ns/tcl/lib/ns-
node.tcl=============================================================================
=======================*/

Within Node instproc init args, add the following instvar
        #Location discovery
        nodeAttribute_

Add the following instproc
        #Location discovery
        Node instproc attribute {} {
                return [$self set nodeAttribute_]
        }
/*=============================================================================
================================================*/

/*========~ns/tcl/lib/ns-
namsupp.tcl=============================================================================
=======================*/

Modify Node instproc color as following
        #enable changing the node's color after running the simulator by
Ian Downard
        Node instproc color { color } {
                $self instvar attr_ id_

                set ns [Simulator instance]

                set attr_(COLOR) $color
                set attr_(LCOLOR) $color
                if [$ns is-started] {
                        # color must be initialized
                        $ns puts-nam-config \
                        [eval list "n -t [$ns now] -s $id_ -S COLOR -c
$color -o $attr_(COLOR) -i $color -I $attr_(LCOLOR)"]
                }
        }
/*=============================================================================
================================================*/
```