

STGDH - NS2 Reference Manual

Generated by Doxygen 1.4.6

Fri May 11 10:58:00 2007

Contents

1 Reputation mechanism protocol documentation - NS-2 protocol	1
1.1 Introduction	1
1.2 Installation	1
1.3 Running the program	2
1.4 Copyright and License	6
2 STGDH - NS2 Namespace Documentation	7
2.1 std Namespace Reference	7
3 STGDH - NS2 Class Documentation	9
3.1 DGrR_Timer Class Reference	9
3.2 DGrRAgent Class Reference	12
3.3 DGrRClass Class Reference	30
3.4 DGrRHeaderClass Class Reference	32
3.5 hdr_DGrR Struct Reference	33
4 STGDH - NS2 File Documentation	37
4.1 bin/ns-2/ns-2.31/DGrR/DGrR.cc File Reference	37
4.2 bin/ns-2/ns-2.31/DGrR/DGrR.h File Reference	38
4.3 bin/ns-2/ns-2.31/DGrR/modes.h File Reference	42

Chapter 1

Reputation mechanism protocol documentation - NS-2 protocol

1.1 Introduction

The DGrR (Distributed GRoup Reputation) project is a reputation extension of NS-2. Though the actual version as been tested on the OLSR protocol, simple modifications on the OTCL file ([OTCL example file](#)) are required to use our solution on another routing protocol, such as AODV.

Please refer to our full documentation about the recommendation evaluation.

1.1.1 Tools required:

This code is a NS-2 plugin. NS2 have to be installed first, with the OLSR package as this version of the project is made for OLSR (otherwise, you only need to update the OTCL file describes in [OTCL example file](#)).

NS-2 is available at <http://www.isi.edu/nsnam/ns/>

1.2 Installation

In order to install this application layer protocol, the package has to be installed in the NS-2 protocols folder : `#/ns-2/ns-2.31`.

Decompress the whole files in a the `#/ns-2/ns-2.31/DGrR/` subdirectory

Then, modifications have to be performed on basic ns2 files in order to integrate this package (an automatic patch is not yet available):

- `#/ns-2/ns-2.31/Makefile` : add `DGrR/DGrR.o \` like

```
DGrR/DGrR.o $$  
common/ns-process.o $$  
satellite/satgeometry.o satellite/sathandoff.o $$
```

- `#/ns-2/ns-2.31/tcl/lib/ns-default.tcl` : add `# DGrR, Agent/DGrR set packetSize_ 64` at the end of the file, like

```

Agent/OLSR set mid_ival_ 5

# DGrR
Agent/DGrR set packetSize_ 64

• #/ns-2/ns-2.31/tcl/lib/ns-packet.tcl : add DGrR # DGrR

  # Application-Layer Protocols:
  Message # a protocol to carry text messages
  Ping # Ping
  DGrR # DGrR

• #/ns-2/ns-2.31/common/packet.h : //DGrR, PT_DGrR

DGrR
PT_DGrR,

insert new packet types here
PT_NTTYPE // This MUST be the LAST one

• #/ns-2/ns-2.31/common/packet.h : //DGrR, name_[PT_DGrR] = "DGrR";

DGrR
name_[PT_DGrR] = "DGrR";

name_[PT_NTTYPE]= "undefined";
}

```

Full explanation are available at <http://www.isi.edu/nsnam/ns/tutorial/nsnew.html#third>

1.3 Running the program

1.3.1 NS2 recompilation

Before launching ns with an adapted OTCL file, the simulator must be recompiled :

```

cd $nsfolder
make

```

where \$nsfolder should look like #/ns-2/ns-2.31

1.3.2 OTCL example file

```

# =====
# Define options
# =====
set opt(chan)          Channel/WirelessChannel ;# channel type
set opt(prop)          Propagation/TwoRayGround ;# radio-propagation model
set opt(netif)         Phy/WirelessPhy ;# network interface type
set opt(mac)           Mac/802_11 ;# MAC type
set opt(ifq)           Queue/DropTail/PriQueue ;# interface queue type

```

```

set opt(ll)           LL                               ;# link layer type
set opt(ant)          Antenna/OmniAntenna             ;# antenna model
set opt(ifqlen)       50                               ;# max packet in ifq
set opt(nn)           4                               ;# number of mobilenodes
set opt(adhocRouting) AODV                           ;# routing protocol
set opt(cp)           "" ; #"/bin/ns-2/ns-2.31/tcl/mobility/scene/cbr-3-test" ; # connection pattern file
set opt(sc)           "" ; #"/bin/ns-2/ns-2.31/tcl/mobility/scene/scen-3-test" ; # node movement file.

set opt(x)           400                               ;# x coordinate of topology
set opt(y)           600                               ;# y coordinate of topology
set opt(seed)        0.0                             ;# seed for random number gen.
set opt(stop)        8000                            ;# time to stop simulation

# =====

#
# check for random seed
#
if {$opt(seed) > 0} {
    puts "Seeding Random number generator with $opt(seed)\n"
    ns-random $opt(seed)
}

#
# create simulator instance
#
set ns_ [new Simulator]

#$ns_ set-address-format expanded

#
# control OLSR behaviour from this script -
# commented lines are not needed because
# those are default values
#
Agent/OLSR set use_mac_ true
Agent/OLSR set debug_ false
Agent/OLSR set willingness 3
Agent/OLSR set hello_ival_ 2
Agent/OLSR set tc_ival_ 5

#
# control reputation mechanism
#

# synchronisation interval
set DGrR(reptimer_ival_) 50
set DGrR(groupSize) 4

# % of malicious nodes at most !
set DGrR(tau_) 10
set DGrR(NRmax_) 10

#
bindings

```

```

#
Agent/DGrR set reptimer_ival_ $DGrR(reptimer_ival_)
Agent/DGrR set tau_ $DGrR(tau_)
Agent/DGrR set NRmax_ $DGrR(NRmax_)

#
# open traces
#
set tracefd [open olsr_example.tr w]
set namtrace [open olsr_example.nam w]

set DGrRtrace [open olsr_example.rep w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

#
# create topography object
#
set topo [new Topography]

#
# define topology
#
$topo load_flatgrid $opt(x) $opt(y)

#
# create God
#
create-god $opt(nn)

set chan_1_ [new $opt(chan)]

#
# configure mobile nodes
#
$ns_ node-config -adhocRouting $opt(adhocRouting) $\$
                -llType $opt(ll) $\$
                -macType $opt(mac) $\$
                -ifqType $opt(ifq) $\$
                -ifqLen $opt(ifqlen) $\$
                -antType $opt(ant) $\$
                -propType $opt(prop) $\$
                -phyType $opt(netif) $\$
                -topoInstance $topo $\$
                -wiredRouting OFF $\$
                -agentTrace ON $\$
                -routerTrace ON $\$
                -macTrace OFF $\$
                -channel $chan_1_
#-channelType $opt(chan) $\$

create nodes & start
for {set i 0} {$i < $opt(nn)} {incr i} {
    set node_($i) [$ns_ node]
    $ns_ at 0.1 "$node_($i) start"
}

```

```

}

#
define group node (attach-agent)
#
for {set i 0} {$i < $DGrR(groupSize)} {incr i} {
    set olsrRAgent($i) [new Agent/DGrR]
    $node_($i) random-motion 0 ;# disable random motion
    $ns_ attach-agent $node_($i) $olsrRAgent($i)
}

#
# init group nodes
#

#
# Note : these operations are done only because the group mechanism
# is not implemented in NS-2
# Thus, we emulate two cases :
# - node inside the group : "initGroupEntry"
# - node outside the group and who wish enter the group (here : Universe $\ $ group) : "initEntry"
#
for {set i 0} {$i < $DGrR(groupSize)} {incr i} {

    # send first messages (tables creations)
    # wait for network stability
    $ns_ at 100.0 "$olsrRAgent($i) start"
    $ns_ at 100.0 "$olsrRAgent($i) initbroadcast"
    for {set j 0} {$j < $DGrR(groupSize)} {incr j} {
set id [$node_($j) node-addr]
$ns_ at 1.0 "$olsrRAgent($i) initGroupEntry $id"

    }

    for {set j $DGrR(groupSize)} {$j < $opt(nm) } {incr j} {
set id [$node_($j) node-addr]
$ns_ at 1.0 "$olsrRAgent($i) initentry $id"

    }

}

for {set i 0} {$i < $DGrR(groupSize)} {incr i} {
    $ns_ at $opt(stop).0001 "$olsrRAgent($i) finaldisplay"
}

#
# source connection-pattern and node-movement scripts
#
if { $opt(cp) == "" } {
    puts "*** NOTE: no connection pattern specified."
    set opt(cp) "none"
} else {
    puts "Loading connection pattern..."
    source $opt(cp)
}

```

```

}
if { $opt(sc) == "" } {
    puts "*** NOTE: no scenario file specified."
    set opt(sc) "none"
} else {
    puts "Loading scenario file..."
    source $opt(sc)
    puts "Load complete..."
}

#
# define initial node position in nam
#
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 20
}

#
# tell all nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn)} {incr i} {

    $ns_ at $opt(stop).0 "$node_($i) reset";
}

$ns_ at $opt(stop).0002 "puts \"NS EXITING...$\" ; $ns_ halt" $ns_ at $opt(stop).0001 "stop"

proc stop {} {
    global ns_ tracefd namtrace
    $ns_ flush-trace
    close $tracefd
    close $namtrace
    # exec nam out.nam &
    exit 0
}

#
# begin simulation
#
puts "Starting Simulation..."

$ns_ run

```

full examples and explanation are available at [TODO](#)

1.4 Copyright and License

[TODO](#)

Chapter 2

STGDH - NS2 Namespace Documentation

2.1 std Namespace Reference

include the std namespace for standard classes

2.1.1 Detailed Description

include the std namespace for standard classes

The std namespace is required to exploited several classes such as
map, required to model the several tables and matrix of the reputation system
list, required to register the different classes of nodes in the network simulation

Chapter 3

STGDH - NS2 Class Documentation

3.1 DGrR_Timer Class Reference

DGrR timer class for reputation interval updates.

```
#include <DGrR.h>
```

Public Member Functions

- [DGrR_Timer](#) (DGrRAgent *agent)
DGrR_Timer constructor.

Public Attributes

- bool [started](#)
Determine whether the algorithm is launched, after a synchronisation time.

Protected Member Functions

- virtual void [expire](#) (Event *e)
TimerHandler event method used to handle elapsed time events.

Protected Attributes

- [DGrRAgent](#) * [agent_](#)
handle on the associated OLSR Reputation agent

3.1.1 Detailed Description

DGrR timer class for reputation interval updates.

The [DGrR_Timer](#) class is used to send reputation messages at regular interval (which is parametered by this timer class).

The [DGrR_Timer](#) class is an extension of the TimerHandler class (include <timer-handler.h>)

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition at line 131 of file DGrR.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 DGrR_Timer::DGrR_Timer ([DGrRAgent](#) * *agent*) [inline]

[DGrR_Timer](#) constructor.

Parameters:

agent the OLSR Reputation agent handler

Definition at line 145 of file DGrR.h.

```
145                                     : TimerHandler(), started(false) {
146                                     agent_ = agent;}
```

3.1.3 Member Function Documentation

3.1.3.1 void DGrR_Timer::expire ([Event](#) * *e*) [protected, virtual]

TimerHandler event method used to handle elapsed time events.

Definition at line 819 of file DGrR.cc.

```
819                                     {
820     if(!started) return;
821     agent_>timer();
822     agent_>set_timer();
823 }
```

3.1.4 Member Data Documentation

3.1.4.1 [DGrRAgent](#)* [DGrR_Timer::agent_](#) [protected]

handle on the associated OLSR Reputation agent

Definition at line 153 of file DGrR.h.

3.1.4.2 bool `DGrR_Timer::started`

Determine whether the algorithm is launched, after a synchronisation time.

Note:

In the mechanism, the timer is activated only when each member knows the other members

Warning:

This parameter must be ignored if the protocol management is developed, as it is required to simulate this protocol

Definition at line 139 of file `DGrR.h`.

The documentation for this class was generated from the following files:

- [bin/ns-2/ns-2.31/DGrR/DGrR.h](#)
- [bin/ns-2/ns-2.31/DGrR/DGrR.cc](#)

3.2 DGrRAgent Class Reference

DGrR agent implementation.

```
#include <DGrR.h>
```

Public Member Functions

- [DGrRAgent](#) ()
Class constructor.
- int [buildLocalReputation](#) (nsaddr_t node)
Local Reputation evaluation.
- int [buildGroupReputation](#) (nsaddr_t node)
Build the group reputation value for the node in parameter.
- void [display](#) ()
Print information about the current node.
- virtual int [command](#) (int argc, const char *const *argv)
NS-2 standard function: communiaction interface with the TCL script.
- virtual void [recv](#) (Packet *packet, Handler *handler)
Receive a reputation packet: update the reputation parameters.
- void [badEvent](#) (nsaddr_t nodeAddr)
Bad events has to be taken into account only during updates.
- void [printRepMatrix](#) ()
Display the reputation values.
- void [printRecTable](#) ()
Display the recommandation values.
- void [decisionsValues](#) ()
Display informations about decision which would be taken.

Static Public Attributes

- static const int [DEFAULT_REPUTATION](#) = 50
Define the different default values : reputation.
- static const int [DEFAULT_RECOMMANDATION](#) = 50
Define the different default values : recommandation.
- static const int [TAU_UPDATE_EVENT](#) = 4
Define the different paramaters.

- static const int `LOCALPRCT` = 100
Define the different paramaters.
- static const int `REPUTATION_ADDING` = 80
Define the different thresholds Adding threshold.
- static const int `REPUTATION_REMOVING` = 10
Define the different thresholds.
- static const int `RECOMMANDATION_THRESHOLD` = 90
Define the different thresholds.

Private Member Functions

- void `sendDGrR` (reputationTable tab)
Send a reputations update message.
- int `diff` (int viewa, int viewb)
Recommendation evaluation, reputation differences function.
- void `buildGroupRecommandation` ()
Evaluate the new recommendation of each nodes.
- void `updateReputation` ()
Update the reputation of all nodes it watches.
- void `set_timer` ()
- void `timer` ()
- int & `timer_ival` ()
- void `broadCastReputations` ()
- void `updateEvent` ()
Update the reputation of each nodes, with TAU_UPDATE_EVENT.
- void `freeMemory` ()
Clean the memory when a node is released.
- int `getMajorityRep` (nsaddr_t node)
Return the majoritar reputation value of the node in parameter.

Private Attributes

- int `TAU`
The supported percentage of malicious nodes.
- int `timer_ival_`
The update interval value.

- int [NSrec](#)
The history conservation value.
- int [NRmax_](#)
The maximal bad behavior supported.
- `map< nsaddr_t, int >` [recTable](#)
Recommandation table.
- `reputationMatrix` [repMatrix](#)
Reputation table.
- `betaTable` [bTable](#)
Reputation decrease rates Table.
- `DGrR_Timer` [rep_timer](#)
Timer for sending reputation messages.
- `nodeList` [grpList](#)
List of the nodes who belong to the group.
- `nodeList` [outsidersList](#)
List of the external nodes.
- long [intervall](#)
The current interval ID.

Friends

- class [DGrR_Timer](#)

3.2.1 Detailed Description

DGrR agent implementation.

Class [DGrRAgent](#) : reputation agent integrated in NS-2 This class is the main component of this plugin

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition at line 173 of file DGrR.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 DGrRAgent::DGrRAgent ()

Class constructor.

Definition at line 335 of file DGrR.cc.

```

335             : Agent(PT_DGrR), rep_timer(this), interval(0) { // trace(NULL){
336
337         timer_ival_ = -1;
338         size_ = -1;
339         TAU = -1;
340         NRmax_ = -1;
341
342         bind("reptimer_ival_", &timer_ival_);
343         if(timer_ival_ == -1){
344             fprintf(stderr,"timer_ival_ not defined\n");
345         }
346         bind("packetSize_", &size_);
347         if(size_ == -1){
348             fprintf(stderr,"packetSize_ not defined\n");
349         }
350         bind("tau_", &TAU);
351         if(TAU == -1){
352             fprintf(stderr,"tau_ not defined\n");
353         }
354
355         bind("NRmax_", &NRmax_);
356         if(NRmax_ == -1){
357             fprintf(stderr,"NRmax_ not defined\n");
358         }
359
360         if((TAU == -1) || (size_ == -1) || (timer_ival_ == -1)
361            || (NRmax_ == -1)){
362             throw 1;
363         }
364
365         bind("NSrec_", &NSrec);
366         if(TAU > 25){
367             TAU = 25;
368         }
369         NSrec = -1;
370
371         if(NSrec == -1){
372             NSrec = 30;
373             fprintf(stderr,"no NSrec defined, use default value %i\n", NSrec);
374         }
375
376
377
378 }

```

3.2.3 Member Function Documentation

3.2.3.1 void DGrRAgent::badEvent (nsaddr_t nodeAddr)

Bad events has to be taken into account only during updates.

Parameters:

nodeAddr the node whose bad behavior is detected

Definition at line 562 of file DGrR.cc.

```

562                                     {
563     bTable[nodeAddr] ++;
564     if(      bTable[nodeAddr] > NRmax_) bTable[nodeAddr] = NRmax_;
565
566     (*repMatrix[nodeAddr])[here_.addr_] -=
567     (TAU_UPDATE_EVENT +
568     pow(2,bTable[nodeAddr]) * 100 / pow(2,NRmax_));
569
570     if((*repMatrix[nodeAddr])[here_.addr_] < 0)
571         (*repMatrix[nodeAddr])[here_.addr_] = 0;
572
573 }

```

3.2.3.2 void DGrRAgent::broadcastReputations () [private]

Definition at line 779 of file DGrR.cc.

```

779                                     {
780     reputationMatrix::iterator matrixIterator = repMatrix.begin();
781     map<nsaddr_t, int> _map = map<nsaddr_t, int>();
782     while(matrixIterator != repMatrix.end()){
783         _map[(*matrixIterator).first] = (*(matrixIterator).second)[here_.addr_];
784         matrixIterator ++;
785     }
786
787     sendDGrR(_map);
788 }

```

3.2.3.3 void DGrRAgent::buildGroupRecommandation () [private]

Evaluate the new recommendation of each nodes.

Warning:

As a node watches over each other members, we have here a diff on each nodes. must be changed if the watch is changed.

Definition at line 475 of file DGrR.cc.

```

475                                     {
476     //updateRecommandationSS
477
478     reputationMatrix::iterator matrixIterator = repMatrix.begin();
479
480     map<nsaddr_t, int> global_rep = map<nsaddr_t, int>();
481     map<nsaddr_t, int> majority_rep = map<nsaddr_t, int>();
482
483     while(matrixIterator != repMatrix.end()){
484         global_rep[(*matrixIterator).first] = buildGroupReputation((*matrixIterator).first);
485
486         majority_rep[(*matrixIterator).first] = getMajorityRep((*matrixIterator).first);
487         matrixIterator++;
488     }
489
490
491
492
493     matrixIterator = repMatrix.begin(); //for all nodeSrc
494     reputationMatrix::iterator subMatrixIterator;
495     int tmpvalue=0;

```

```

496     int localvalue;
497
498     //for all member of the group
499     for(nodeList::iterator iter = grpList.begin(); iter != grpList.end(); iter++)
500     {
501
502         //     tmpvalue : 100 if node is majority
503         //     0 otherwise
504         //check reputation
505         tmpvalue=100;
506         subMatrixIterator = repMatrix.begin();
507
508         #ifdef PIMODE
509             while(subMatrixIterator != repMatrix.end()){
510         #else
511             #ifdef SUMMODE
512                 int size=0;
513                 tmpvalue=0;
514                 while(subMatrixIterator != repMatrix.end()){
515                     size++;
516             #else
517                 while(subMatrixIterator != repMatrix.end() && (tmpvalue==100)){
518             #endif
519         #endif
520
521             localvalue = (*(subMatrixIterator).second)[*iter];
522
523
524             #ifdef PIMODE
525                 //PIMODE
526                 tmpvalue = tmpvalue * diff(localvalue, majority_rep[*(subMatrixIterator).first]); //
527                 tmpvalue = tmpvalue /100;
528             #else
529                 #ifdef SUMMODE
530                     tmpvalue += diff(localvalue, global_rep[*(subMatrixIterator).first]); //glob
531                 #else
532                     tmpvalue = diff(localvalue, majority_rep[*(subMatrixIterator).first]);
533                     if(tmpvalue != 100) tmpvalue= 0;
534                 #endif
535
536             #endif
537
538             subMatrixIterator++;
539         }
540         #ifdef SUMMODE
541             tmpvalue = tmpvalue / size;
542         #endif
543         // node dependant value of NSrec
544         recTable[*iter] = NSrec * recTable[*iter] + (100-NSrec) * tmpvalue;
545         recTable[*iter] = recTable[*iter] /100;
546     }
547
548 }

```

3.2.3.4 int DGrRAgent::buildGroupReputation (nsaddr_t node)

Build the group reputation value for the node in parameter.

Note:

for now $R = \text{group}$. It would be better if $R = \text{fau}(\tau)$

Warning:

as a node watch over each other members, we have here a diff on each nodes. must be changed if the watch is changed.

Definition at line 385 of file DGrR.cc.

```

385                                     {
386         int somme = 0;
387         int fraction = 0;
388         int cFraction;
389
390     for(nodeList::iterator iter = grpList.begin(); iter != grpList.end(); iter++)
391     {
392         cFraction = recTable[*iter];
393         fraction += cFraction;
394         somme += cFraction * (*repMatrix[node])*[*iter];
395     }
396
397     if(fraction ==0) return DEFAULT_REPUTATION;
398
399     return somme/fraction;
400
401 }
```

3.2.3.5 int DGrRAgent::buildLocalReputation (nsaddr_t node)

Local Reputation evaluation.

New reputation = my_reputation + PI (reputation_i * recommandation_i) new reputation = normalize(new reputation); for now: R = group or even better : R = R = fau(tau)

Warning:

as a node watch over each other members, we have here a diff on each nodes. must be changed if the watch is changed.

Definition at line 403 of file DGrR.cc.

```

403                                     {
404         #ifdef WorstCASE1
405             if((node == 1) && (here_.addr_ == 0)) return -4;
406         #endif
407         #ifdef WorstCASE2
408             if((node == 1) && (here_.addr_ == 0)) return 100;
409         #endif
410         #ifdef WorstCASE1b
411             if((node == 1) && (here_.addr_ == 0) && (intervall %2 == 0)) return -4;
412         #endif
413         int somme = 0;
414         int fraction = 0;
415         int cFraction;
416
417     for(nodeList::iterator iter = grpList.begin(); iter != grpList.end(); iter++)
418     {
419         if((*iter) != here_.addr_){
420             cFraction = recTable[*iter];
421             fraction += cFraction;
422             somme += cFraction * (*repMatrix[node])*[*iter];
423         }
424     }
425
426     if(fraction ==0) return DEFAULT_REPUTATION;
427
428     return (LOCALPRCT * (*repMatrix[node])[here_.addr_] + somme) / (fraction + LOCALPRCT);
429 }
```

3.2.3.6 int DGrRAgent::command (int argc, const char *const * argv) [virtual]

NS-2 standard function: communication interface with the TCL script.

Note:

This function is a standard redefinition of the NS2 `command(int, const char*const*)` function

Parameters:

argc number of elements in the second parameter

argv message received

Definition at line 624 of file DGrR.cc.

```

624                                     {
625
626
627     //start the node execution : activate timers
628     if (argc == 2 && strncasecmp(argv[1], "start", strlen("start")) == 0) {
629         rep_timer.started = true;
630         rep_timer.resched((double)timer_ival());
631         return TCL_OK;
632     }
633
634     //first broadcast : defined default values
635     if (argc == 2 && strncasecmp(argv[1], "initBroadcast", strlen("initBroadcast")) == 0) {
636
637         //init
638         reputationMatrix::iterator matrixIterator = repMatrix.begin();
639         reputationMatrix::iterator tableIterator;
640         while(matrixIterator != repMatrix.end()){
641             tableIterator = repMatrix.begin();
642             while(tableIterator != repMatrix.end()){
643                 ((*matrixIterator).second)[(*tableIterator).first]
644                 = DEFAULT_REPUTATION;
645                 tableIterator++;
646             }
647             matrixIterator++;
648         }
649         broadCastReputations();
650         return TCL_OK;
651     }
652
653     //add a new node to the memory, not group member
654     if (argc == 3 && strncasecmp(argv[1], "initEntry", strlen("initEntry")) == 0) {
655         nsaddr_t node = (nsaddr_t)Address::instance().str2addr(argv[2]);
656
657         repMatrix[node] = new reputationTable();
658         outsidersList.push_back(node);
659         //fill default values
660         recTable[node] = DEFAULT_RECOMMANDATION;
661
662         return (TCL_OK);
663     }
664
665
666     //add a new node to the memory, group member
667     if (argc == 3 && strncasecmp(argv[1], "initGroupEntry", strlen("initGroupEntry")) == 0) {
668         nsaddr_t node = (nsaddr_t)Address::instance().str2addr(argv[2]);
669
670         repMatrix[node] = new reputationTable();
671         grpList.push_back(node);
672         //fill default values
673         recTable[node] = DEFAULT_RECOMMANDATION;
674
675

```

```

676         return (TCL_OK);
677     }
678
679     //finish procedure
680     if (argc == 2 && strcasecmp(argv[1], "finalDisplay", strlen("finalDisplay")) == 0) {
681         rep_timer.cancel();
682         printf("reset node %i\n", here_.addr_);
683         //stop timer
684         display();
685
686         //free memory
687         freeMemory();
688         return TCL_OK;
689
690     return (TCL_OK);
691 }
692
693
694 // If the command hasn't been processed by the agent
695 // call the command() function for the base class
696 return (Agent::command(argc, argv));
697 }

```

3.2.3.7 void DGrRAgent::decisionsValues ()

Display informations about decision which would be taken.

Note:

decisions are not taken in reality, as the group management protocol does not exist.

Definition at line 830 of file DGrR.cc.

```

830         {
831         // !!
832         //note : if we need globalReputation as group paramter,
833         //we have to first create a groupReputation state (at each
834         //timer event), otherwise we'll have partial views
835         // !!
836         int localReputation;
837         int globalRecommandation;
838         int globalReputation ;
839         char* true_str = "true";
840         char* false_str = "false";
841         char* value;
842
843
844         /*
845         * insiders
846         */
847
848         for(nodeList::iterator iter = grpList.begin(); iter != grpList.end(); iter++)
849     {
850
851         printf("node %i : ",*iter);
852
853         //wants to remove : localReputation
854         localReputation = (* repMatrix[*iter]) [here_.addr_];
855         value = (localReputation < REPUTATION_REMOVING) ? true_str : false_str;
856         printf(", remove = %s (%i) \n",value, localReputation);
857
858         //believe in : globalRecommandation : should be the same for every node
859         globalRecommandation = recTable[*iter];
860         globalReputation = buildGroupReputation(*iter);

```

```

861         value = (globalRecommandation > RECOMMANDATION_THRESHOLD) ? true_str : false_str;
862         printf("believe (recommandation) group messages = %s (%i)",value, globalRecommandation);
863         value = (globalReputation > RECOMMANDATION_THRESHOLD) ? true_str : false_str;
864         printf(", believe (reputation) group messages = %s (%i)\n",value, globalReputation);
865     }
866
867     /* outsiders
868     */
869
870     for(nodeList::iterator iter = outsidersList.begin(); iter != outsidersList.end(); iter++)
871     {
872         printf("node %i : ",*iter);
873
874         //wants to add :localReputation
875         localReputation = (* repMatrix[*iter]) [here_.addr_];
876         value = (localReputation > REPUTATION_ADDING) ? true_str : false_str;
877         printf("add = %s (%i) \n",value, localReputation);
878     }
879 }

```

3.2.3.8 int DGrRAgent::diff (int *viewa*, int *viewb*) [private]

Recommendation evaluation, reputation differences function.

Parameters:

viewa the first view

viewb the second view

Returns:

the difference evaluation of the two views, result in [0,100]. If *viewb* = *viewa*, then [diff\(\)](#) return the maximal value

Definition at line 433 of file DGrR.cc.

```

433     {
434         return 100 - abs(viewa-viewb);
435     }

```

3.2.3.9 void DGrRAgent::display ()

Print information about the current node.

Definition at line 608 of file DGrR.cc.

```

608     {
609         printf("[informations of %i : interval = %li]\n", here_.addr_, intervall);
610
611         #ifdef DEBUG_MODE
612             printRepMatrix();
613         #endif
614
615         decisionsValues();
616         printf("-----\n");
617     }

```

3.2.3.10 void DGrRAgent::freeMemory () [private]

Clean the memory when a node is released.

Definition at line 791 of file DGrR.cc.

```

791         {
792     reputationMatrix::iterator matrixIterator = repMatrix.begin();
793     while(matrixIterator != repMatrix.end()){
794         free((*matrixIterator).second);
795         matrixIterator++;
796     }
797 }
```

3.2.3.11 int DGrRAgent::getMajorityRep (nsaddr_t node) [private]

Return the majoritar reputation value of the node in parameter.

Parameters:

node the node whose reputation is required

Definition at line 451 of file DGrR.cc.

```

451         {
452
453     reputationMatrix::iterator matrixIterator = repMatrix.begin();
454     map<int, int> tab = map<int, int>();
455
456     while(matrixIterator != repMatrix.end()){
457         tab[((*repMatrix[node])[(*matrixIterator).first])++]++;
458         matrixIterator++;
459     }
460     map<int, int>::iterator iter = tab.begin();
461     int maxnb = 0;
462     int rep = 0;
463     while(iter != tab.end()){
464         if((*iter).second > maxnb){
465             maxnb = (*iter).second;
466             rep = (*iter).first;
467         }
468         iter++;
469     }
470
471     return rep;
472 }
```

3.2.3.12 void DGrRAgent::printRecTable ()

Display the recommandation values.

Definition at line 597 of file DGrR.cc.

```

597         {
598     reputationTable::iterator tableIterator = recTable.begin();
599     printf("recommendations : ");
600     while(tableIterator !=recTable.end()){
601         printf("%i, ", (*tableIterator).second);
602         tableIterator++;

```

```

603     }
604     printf("\n");
605
606 }

```

3.2.3.13 void DGrRAgent::printRepMatrix ()

Display the reputation values.

Definition at line 581 of file DGrR.cc.

```

581     {
582     reputationMatrix::iterator matrixIterator = repMatrix.begin();
583     reputationTable::iterator tableIterator;
584     while(matrixIterator != repMatrix.end()){
585         printf("%i | ", (*matrixIterator).first);
586         tableIterator = ((*matrixIterator).second).begin();
587         while(tableIterator != ((*matrixIterator).second).end()){
588             printf("%i , ", (*tableIterator).second);
589             tableIterator++;
590         }
591         printf("\n");
592         matrixIterator++;
593     }
594     printf("\n");
595 }

```

3.2.3.14 void DGrRAgent::recv (Packet * *packet*, Handler * *handler*) [virtual]

Receive a reputation packet: update the reputation parameters.

Note:

This function is a standard redefinition of the NS2 [recv\(Packet*, Handler*\)](#) function

Parameters:

packet received packet

handler unused handler

Definition at line 699 of file DGrR.cc.

```

699     {
700     // Access the IP header for the received packet:
701     hdr_ip* hdrIp = hdr_ip::access(pkt);
702
703     // Get the OLSR packet
704     //OLSR_pkt* hdrOLSR = OLSR_pkt::access(pkt);
705
706
707     // Access the DGrR header for the received packet:
708     hdr_DGrR* hdrDGrR = hdr_DGrR::access(pkt);
709     //src of the message;
710
711     #ifdef DEBUG_MODE_FULL
712         if (DEBUG_MODE_FULL) printf("%i : receive from %i \n", here_.addr_, hdrIp -> src_.addr_);
713     #endif
714
715     //check 1 : node src àĉñ groupe
716     //check 2 : node src àĉñ region of hdrIp -> src_ -> node_

```

```

717
718     map<nsaddr_t, int>::iterator miter = (hdrDGrR -> tab_).begin();
719     for( int i = 0; i < hdrDGrR -> tabsize; miter++, i++)
720     {
721         reputationMatrix::iterator matrixIterator = repMatrix.find((*miter).first);
722         reputationTable* map;
723
724         if(matrixIterator == repMatrix.end()){
725             map = new reputationTable();
726             repMatrix[(*miter).first] = map;
727         }else map = (*matrixIterator).second;
728         (*map)[hdrRip -> src_.addr_] = (*miter).second;
729     }
730 }

```

3.2.3.15 void DGrRAgent::sendDGrR ([reputationTable](#) *tab*) [private]

Send a reputations update message.

Parameters:

tab the nodes we are watching over

Definition at line 733 of file DGrR.cc.

```

733                                     {
734     //(nsaddr_t node){
735     // Create a new packet
736
737     map<nsaddr_t, int>::iterator iter = tab.begin();
738
739
740     for(nodeList::iterator iter = grpList.begin(); iter != grpList.end(); iter++)
741     {
742
743         if( ! (here_.addr_ == * iter)){
744
745             Packet* pkt = allocpkt();
746             // Access the DGrR header for the new packet:
747             hdr_DGrR* hdr = hdr_DGrR::access(pkt);
748             hdr -> tab_ = tab;
749             hdr -> tabsize = tab.size();
750             hdr_ip* iph = HDR_IP(pkt);
751             iph->dport() = iph->sport();
752
753             iph->saddr() = here_.addr_;
754             iph->daddr() = * iter;
755             #ifdef DEBUG_MODE_FULL
756                 if(DEBUG_MODE_FULL) printf("%i send to %i ...", here_.addr_, * iter);
757             #endif
758             Scheduler::instance().schedule(target_, pkt, 0.0);
759
760         }
761     }
762
763     /* Packet* pkt = allocpkt();
764     // Access the DGrR header for the new packet:
765     hdr_DGrR* hdr = hdr_DGrR::access(pkt);
766     hdr -> reputation_ = reput;
767     hdr->node_ = node;
768
769     hdr_ip* iph = HDR_IP(pkt);
770     iph->dport() = iph->sport();
771

```

```

772
773         iph->saddr() = here_.addr_;
774         iph->daddr() = IP_BROADCAST;
775         Scheduler::instance().schedule(target_, pkt, 0.0);
776         */
777 }

```

3.2.3.16 void DGrRAgent::set_timer () [private]

Definition at line 815 of file DGrR.cc.

```

815         {
816         rep_timer.resched((double)timer_ival());
817 }

```

3.2.3.17 void DGrRAgent::timer () [private]

Definition at line 804 of file DGrR.cc.

```

804         {
805         intervall++;
806         buildGroupRecommandation();
807         //saveReputations();
808         display();
809         updateReputation();
810         broadCastReputations();
811
812 }

```

3.2.3.18 int& DGrRAgent::timer_ival () [inline, private]

Definition at line 419 of file DGrR.h.

```

419 { return timer_ival_; }

```

3.2.3.19 void DGrRAgent::updateEvent () [private]

Update the reputation of each nodes, with TAU_UPDATE_EVENT.

Note:

for now, $R = \text{Universe}$ it would be better if $R = f(\text{tau}) + \text{outsiders seen}$

Warning:

As a node watches over each other members, we have here a diff on each nodes. must be changed if the watch is changed.

Definition at line 439 of file DGrR.cc.

```

439         {
440     reputationMatrix::iterator matrixIterator = repMatrix.begin();
441     while(matrixIterator != repMatrix.end()){
442         ((*matrixIterator).second)[here_.addr_] += TAU_UPDATE_EVENT;
443         if((*matrixIterator).second)[here_.addr_] > 100)
444             ((*matrixIterator).second)[here_.addr_] = 100;
445         matrixIterator++;
446     }
447
448 }

```

3.2.3.20 void DGrRAgent::updateReputation () [private]

Update the reputation of all nodes it watches.

Warning:

As a node watches over each other members, we have here a diff on each nodes. must be changed if the watch is changed.

Definition at line 550 of file DGrR.cc.

```

550         {
551     reputationMatrix::iterator matrixIterator = repMatrix.begin();
552     while(matrixIterator !=repMatrix.end()){
553         //update its value
554         ((*matrixIterator).second)[here_.addr_]
555             = buildLocalReputation((*matrixIterator).first);
556         matrixIterator++;
557     }
558
559     updateEvent();
560 }

```

3.2.4 Friends And Related Function Documentation

3.2.4.1 friend class DGrR_Timer [friend]

Define the friend classes

Definition at line 178 of file DGrR.h.

3.2.5 Member Data Documentation

3.2.5.1 betaTable DGrRAgent::bTable [private]

Reputation decrease rates Table.

Definition at line 351 of file DGrR.h.

3.2.5.2 const int DGrRAgent::DEFAULT_RECOMMANDATION = 50 [static]

Define the different default values : recommandation.

Definition at line 190 of file DGrR.h.

3.2.5.3 `const int DGrRAgent::DEFAULT_REPUTATION = 50` [static]

Define the different default values : reputation.

Definition at line 186 of file DGrR.h.

3.2.5.4 `nodeList DGrRAgent::grpList` [private]

List of the nodes who belong to the group.

Definition at line 361 of file DGrR.h.

3.2.5.5 `long DGrRAgent::intervall` [private]

The current interval ID.

Definition at line 374 of file DGrR.h.

3.2.5.6 `const int DGrRAgent::LOCALPRCT = 100` [static]

Define the different paramaters.

LOCALPRCT: how do we consider our judgement, regarding others judgements, for personal decisions

Definition at line 205 of file DGrR.h.

3.2.5.7 `int DGrRAgent::NRmax_` [private]

The maximal bad behavior supported.

Definition at line 329 of file DGrR.h.

3.2.5.8 `int DGrRAgent::NSrec` [private]

The history conservation value.

Definition at line 324 of file DGrR.h.

3.2.5.9 `nodeList DGrRAgent::outsidersList` [private]

List of the external nodes.

Note:

OutsidersList U grpList is the considered universe. The other nodes are not studied by the algorithm

Definition at line 369 of file DGrR.h.

3.2.5.10 `const int DGrRAgent::RECOMMANDATION_THRESHOLD = 90`
[static]

Define the different thresholds.

Recommandation threshold

Definition at line 226 of file DGrR.h.

3.2.5.11 `map<nsaddr_t, int> DGrRAgent::recTable` [private]

Recommandation table.

Definition at line 340 of file DGrR.h.

3.2.5.12 `DGrR_Timer DGrRAgent::rep_timer` [private]

Timer for sending reputation messages.

Definition at line 356 of file DGrR.h.

3.2.5.13 `reputationMatrix DGrRAgent::repMatrix` [private]

Reputation table.

Note:

in the project, this matrix is used in the following way: `matrix[i][j]` = declared reputation of i by j (j opinion)

Definition at line 346 of file DGrR.h.

3.2.5.14 `const int DGrRAgent::REPUTATION_ADDING = 80` [static]

Define the different thresholds Adding threshold.

Definition at line 212 of file DGrR.h.

3.2.5.15 `const int DGrRAgent::REPUTATION_REMOVING = 10` [static]

Define the different thresholds.

Eviction threshold

Definition at line 219 of file DGrR.h.

3.2.5.16 `int DGrRAgent::TAU` [private]

The supported percentage of malicious nodes.

Definition at line 313 of file DGrR.h.

3.2.5.17 `const int DGrRAgent::TAU_UPDATE_EVENT = 4` [static]

Define the different paramaters.

`TAU_UPDATE_EVENT` = alpha

Definition at line 197 of file DGrR.h.

3.2.5.18 int [DGrRAgent::timer_ival_](#) [private]

The update interval value.

Definition at line 318 of file DGrR.h.

The documentation for this class was generated from the following files:

- [bin/ns-2/ns-2.31/DGrR/DGrR.h](#)
- [bin/ns-2/ns-2.31/DGrR/DGrR.cc](#)

3.3 DGrRClass Class Reference

NS2 required class.

```
#include <DGrR.h>
```

Public Member Functions

- [DGrRClass](#) ()
DGrRClass constructor.
- TclObject * [create](#) (int, const char *const *)
DGrRClass create method.
This function is required for the otcl files be able to create DGrR agents.

3.3.1 Detailed Description

NS2 required class.

Define the tcl class for the OLSR reputation implementation in NS2

See also:

TclClass

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition at line 491 of file DGrR.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DGrRClass::DGrRClass () [inline]

[DGrRClass](#) constructor.

The constructor is an immediate extension of the TclClass constructor

Note:

This method used the DGrR term define in NS2 common files

Definition at line 501 of file DGrR.h.

```
501 : TclClass("Agent/DGrR"){}
```

3.3.3 Member Function Documentation

3.3.3.1 TelObject* DGrRClass::create (int, const char *const *) [inline]

[DGrRClass](#) create method.

This function is required for the otcl files be able to create DGrR agents.

Definition at line 507 of file DGrR.h.

```
507                                     {
508         return (new DGrRAgent());
509     }
```

The documentation for this class was generated from the following file:

- [bin/ns-2/ns-2.31/DGrR/DGrR.h](#)

3.4 DGrRHeaderClass Class Reference

NS2 required class.

```
#include <DGrR.h>
```

Public Member Functions

- [DGrRHeaderClass \(\)](#)
DGrRHeaderClass constructor.

3.4.1 Detailed Description

NS2 required class.

Define the packet header class for the OLSR reputation implementation in NS2

See also:

[PacketHeaderClass](#)

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition at line 465 of file DGrR.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DGrRHeaderClass::DGrRHeaderClass () [inline]

[DGrRHeaderClass](#) constructor.

The constructor is an immediate extension of the [PacketHeaderClass](#) constructor

Note:

This method used the DGrR term define in NS2 common files

Definition at line 475 of file DGrR.h.

```
475                                     : PacketHeaderClass("PacketHeader/DGrR",
476                                     sizeof(hdr_DGrR)) {
477                                     bind_offset(&hdr_DGrR::offset_);
478     }
```

The documentation for this class was generated from the following file:

- [bin/ns-2/ns-2.31/DGrR/DGrR.h](#)

3.5 `hdr_DGrR` Struct Reference

DGrR packet header.

```
#include <DGrR.h>
```

Static Public Member Functions

- static int & `offset` ()
Return the header offset This method is required by PacketHeaderManager and is a standard in NS-2 packet definitin.
- static `hdr_DGrR` * `access` (const Packet *packet)
Return the OLSR Reputation packet.

Public Attributes

- `reputationTable` `tab_`
The reputation values, stored in a reputation table.
- int `tabsize`
Determine the table size.

Static Public Attributes

- static int `offset_`
Offset value in the global packet header This is an automanaged parameter required by Packet-HeaderManager.

3.5.1 Detailed Description

DGrR packet header.

Define the packet header for the OLSR reputation implementation in NS2

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition at line 77 of file `DGrR.h`.

3.5.2 Member Function Documentation

3.5.2.1 static [hdr_DGrR*](#) [hdr_DGrR::access](#) (const Packet * *packet*) [inline, static]

Return the OLSR Reputation packet.

Parameters:

packet the packet pointer the NS2 agent receive

Returns:

[hdr_DGrR*](#) a pointer on the reputation header

Definition at line 110 of file DGrR.h.

```

110                                     {
111         return (hdr_DGrR*) packet->access(offset_);
112     }
```

3.5.2.2 static [int&](#) [hdr_DGrR::offset](#) () [inline, static]

Return the header offset This method is required by PacketHeaderManager and is a standard in NS-2 packet definitin.

See also:

[access\(const Packet* packet\)](#)

Definition at line 103 of file DGrR.h.

```

103 { return offset_; }
```

3.5.3 Member Data Documentation

3.5.3.1 [int](#) [hdr_DGrR::offset_](#) [static]

Offset value in the global packet header This is an automanaged parameter required by PacketHeaderManager.

Definition at line 95 of file DGrR.h.

3.5.3.2 [reputationTable](#) [hdr_DGrR::tab_](#)

The reputation values, stored in a reputation table.

Definition at line 82 of file DGrR.h.

3.5.3.3 [int](#) [hdr_DGrR::tabsize](#)

Determine the table size.

Note:

for an unknown reason, the table is not well propagated in NS2. So, this parameter is required for the moment.

Definition at line 89 of file `DGrR.h`.

The documentation for this struct was generated from the following files:

- [bin/ns-2/ns-2.31/DGrR/DGrR.h](#)
- [bin/ns-2/ns-2.31/DGrR/DGrR.cc](#)

Chapter 4

STGDH - NS2 File Documentation

4.1 bin/ns-2/ns-2.31/DGrR/DGrR.cc File Reference

implementation of the reputation mechanism specifications

```
#include "DGrR.h"  
#include <trace.h>
```

4.1.1 Detailed Description

implementation of the reputation mechanism specifications

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition in file [DGrR.cc](#).

4.2 bin/ns-2/ns-2.31/DGrR/DGrR.h File Reference

definition of the reputation mechanism specifications

```
#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"
#include <timer-handler.h>
#include "olsr/OLSR_pkt.h"
#include "modes.h"
#include <map>
#include <list>
```

Namespaces

- namespace [std](#)

Classes

- struct [hdr_DGrR](#)
DGrR packet header.
- class [DGrR_Timer](#)
DGrR timer class for reputation interval updates.
- class [DGrRAgent](#)
DGrR agent implementation.
- class [DGrRHeaderClass](#)
NS2 requiered class.
- class [DGrRClass](#)
NS2 requiered class.

Typedefs

- typedef `std::map< nsaddr_t, int >` [reputationTable](#)
Reputation list: node -> reputation value use a std::map object.
- typedef `std::map< nsaddr_t, int >` [betaTable](#)
Decreasetlist: node -> decrease threshold value (beta) use a std::map object.
- typedef `std::map< nsaddr_t, reputationTable * >` [reputationMatrix](#)

Reputation matrix: node -> reputation list use a std::map object.

- typedef std::list< nsaddr_t > [nodeList](#)

Defintion for a list of nodes.

Functions

- int [min](#) (int a, int b)

Variables

- [DGrRHeaderClass class_DGrRhdr](#)

NS2 requiered class.

- [DGrRClass class_DGrR](#)

NS2 requiered class.

4.2.1 Detailed Description

definition of the reputation mechanism specifications

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition in file [DGrR.h](#).

4.2.2 Typedef Documentation

4.2.2.1 typedef std::map<nsaddr_t, int> [betaTable](#)

Decreaselist: node -> decrease threshold value (beta) use a std::map object.

Definition at line 55 of file DGrR.h.

4.2.2.2 typedef std::list<nsaddr_t> [nodeList](#)

Defintion for a list of nodes.

Definition at line 65 of file DGrR.h.

4.2.2.3 typedef std::map<nsaddr_t, reputationTable*> reputationMatrix

Reputation matrix: node -> reputation list use a std::map object.

Definition at line 60 of file DGrR.h.

4.2.2.4 typedef std::map<nsaddr_t, int> reputationTable

Reputation list: node -> reputation value use a std::map object.

Definition at line 48 of file DGrR.h.

4.2.3 Function Documentation

4.2.3.1 int min (int a, int b)

Definition at line 448 of file DGrR.h.

```

448         {
449         if (a<b) return a;
450         return b;
451 }
```

4.2.4 Variable Documentation

4.2.4.1 DGrRClass class_DGrR [static]

NS2 required class.

Define the tcl class for the OLSR reputation implementation in NS2

See also:

TclClass

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

4.2.4.2 DGrRHeaderClass class_DGrRhdr [static]

NS2 required class.

Define the packet header class for the OLSR reputation implementation in NS2

See also:

PacketHeaderClass

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

4.3 bin/ns-2/ns-2.31/DGrR/modes.h File Reference

Define the different behaviors implemented in NS2.

Defines

- `#define DEBUG_MODE false`
- `#define DEBUG_MODE_FULL false`
- `#define NOBEHAVIORCASE`
- `#define LYINGMODE`

4.3.1 Detailed Description

Define the different behaviors implemented in NS2.

Author:

Julien Thomas

Version:

1.0

Date:

2007/05/09

Definition in file [modes.h](#).

4.3.2 Define Documentation

4.3.2.1 `#define DEBUG_MODE false`

Debug information: reputation outputs

Note:

the different defined default value is false Set to true if an advanced debug mode is required

Definition at line 17 of file [modes.h](#).

4.3.2.2 `#define DEBUG_MODE_FULL false`

Debug information: messages transmission outputs

Note:

the different defined default value is false Set to true if an advanced debug mode is required

Definition at line 24 of file [modes.h](#).

4.3.2.3 `#define LYINGMODE`

Recommendation evaluation mode

Definition at line 38 of file [modes.h](#).

4.3.2.4 `#define NOBEHAVIORCASE`

Worst cases simulations

Definition at line 29 of file modes.h.