Rapport ARA

Mickael Rudek, Oskar Viljasaar

17 janvier 2018

1 Exercice 1

1.1 Question 1

```
Movement protocol:
- if not moving
    - assign random speed lower than max
- moving
    - use ‘PositioningStrategiesFactory‘ to get next destination
    - Calc distance to next destination
    - if too far to reach ‘destination‘ in one hop
        - calculate next ‘x‘ and ‘y‘
        - move to that position
    - if destination reached, stop
    - else continue running
```

L'algorithme utilise le protocole de déplacement suivant : Une valeur de la vitesse est aléatoirement choisie dans l'intervalle [`speed_min`; `speed_max`]. Vu qu'on est en temps discretisé, 'distance_to_next' représente la distance parcourue au Une fois la destination atteinte, le noeud s'arrête pendant un tic.

1.2 Question 2

```
simulation.endtime 50000
random.seed 5
network.size 10
init.initialisation Initialisation
control.graph GraphicalMonitor
control.graph.positionprotocol position
control.graph.time_slow 0.0002
control.graph.step 1
```

1.3 Question 3

La stratégie 1 choisit aléatoirement la prochaine destination dans les intervalles [0; `maxX`] et [0; `maxY`].

1.4 Question 4

La stratégie 2 choisit comme destination l'endroit courant du noeud, celui-ci reste donc immobile.

1.5 Question 5

```
package manet.communication;

import manet.Message;
import manet.positioning.Position;
import manet.positioning.PositionProtocol;
import peersim.config.Configuration;
import peersim.core.Network;
import peersim.core.Node;
import peersim.core.Protocol;
import peersim.edsim.EDSimulator;

public class EmitterImpl implements Emitter {

    private int latency;
    private int scope;
    private int this_pid;
    private int position_protocol;

    private static final String PAR_LATENCY = "latency";
    private static final String PAR_SCOPE = "scope";
    private static final String PAR_POSITIONPROTOCOL = "positionprotocol";

    public EmitterImpl(String prefix) {
        String tmp[]=prefix.split("\\.");
        this_pid=Configuration.lookupPid(tmp[tmp.length-1]);
        this.position_protocol=Configuration.getPid(prefix+"."+PAR_POSITIONPROTOCOL);
        this.latency = Configuration.getInt(prefix + "." + PAR_LATENCY);
        this.scope = Configuration.getInt(prefix + "." + PAR_SCOPE);
    }

    @Override
    public void emit(Node host, Message msg) {
        PositionProtocol prot = (PositionProtocol) host.getProtocol(position_protocol);

        for (int i=0; i < Network.size(); i++) {
            Node n = Network.get(i);
            PositionProtocol prot2 = (PositionProtocol) n.getProtocol(position_protocol);
            double dist =prot.getCurrentPosition().distance(prot2.getCurrentPosition());
            if (dist < scope && n.getID() != host.getID()) {
                EDSimulator.add(latency, new Message(msg.getIdSrc(), n.getID(), msg.getTag(), msg.getConte
            }
        }

    }

    @Override
    public int getLatency() { return latency; }

    @Override
    public int getScope() { return scope; }

    @Override
    public Object clone(){
        EmitterImpl res=null;
        try {
            res=(EmitterImpl)super.clone();
        } catch (CloneNotSupportedException e) {}
        return res;
    }
}
```

1.6 Question 6

```
package manet.detection;

import manet.Message;
import manet.communication.EmitterImpl;
import peersim.config.Configuration;
import peersim.core.Node;
import peersim.edsim.EDProtocol;
import peersim.edsim.EDSimulator;

import java.util.ArrayList;
import java.util.List;

public class NeighborProtocolImpl implements NeighborProtocol, EDProtocol {
    private int this_pid;
    private int period;
    private int timer_delay;
    private int listener_pid;

    private static final String PAR_PERIOD = "period";
    private static final String PAR_TIMERDELAY = "timer_delay";
    private static final String PAR_LISTENER_PID = "listenerpid";
    Integer timeStamp = 0;

    private List<Long> neighbor_list;

    public NeighborProtocolImpl(String prefix) {
        neighbor_list = new ArrayList<>();

        String tmp[]=prefix.split("\\.");
        this_pid= Configuration.lookupPid(tmp[tmp.length-1]);
        this.period = Configuration.getInt(prefix+"."+PAR_PERIOD);
        this.timer_delay = Configuration.getInt(prefix + "." + PAR_TIMERDELAY);
        this.listener_pid = Configuration.getPid(prefix + "." + PAR_LISTENER_PID,-1);
        }

    @Override
    public List<Long> getNeighbors() { return neighbor_list; }

    @Override
    public Object clone() {
        NeighborProtocolImpl res = null;
        try {
            res = (NeighborProtocolImpl) super.clone();
            neighbor_list = new ArrayList<>();
            timeStamp = new Integer(0);
        } catch (CloneNotSupportedException e) {

        }
        return res;
    }

    @Override
    public void processEvent(Node node, int pid, Object event) {
        int emitter_pid = Configuration.lookupPid("emitter");
        EmitterImpl impl = (EmitterImpl) node.getProtocol(emitter_pid);
        Message msg = (Message) event;

        if (event instanceof Message) {
            switch (msg.getTag()) {
                case "Heartbeat":
                    if (msg.getIdSrc() == msg.getIdDest()) {
```

```
                    EDSimulator.add(this.period, event, node, pid);
                    impl.emit(node, new Message(node.getID(), 0,
                                    "Heartbeat",
                                    "Heartbeat", this_pid));
                }
                else {
                    if(!neighbor_list.contains(msg.getIdSrc()))
                        neighbor_list.add(msg.getIdSrc());
                    break;
                }
                break;
            default:
                System.out.println("IN DEFAULT");
        }
    }
    else {
        System.out.println("no good message");
    }
    return;
  }
}
```

1.7 Question 7

Oui.

1.8 Question 8

Strategy3InitNext fait converger les points vers le milieu, et assure que tous les noeuds sont à portée les uns des autres, dans un rayon de `scope − marge`.

SPI	**SD**	**Graphe**	**Commentaires**
`Strategy1InitNext`	`Strategy1InitNext`	Connexe	
`Strategy1InitNext`	`Strategy2Next`		bouge ap normal
`Strategy1InitNext`	`Strategy3InitNext`	Connexe	Converge vers le milieu
`Strategy1InitNext`	`Strategy4Next`		bouge ap, ça bouge seulement si on a exactement deux noeuds dans le réseau
`Strategy3InitNext`	`Strategy1InitNext`	Connexe	Connexe dès le début
`Strategy3InitNext`	`Strategy2Next`	Connexe	spawn au milieu et ne bouge pas
`Strategy3InitNext`	`Strategy3InitNext`	Connexe	
`Strategy3InitNext`	`Strategy4Next`	Connexe	Ça chill dans le scope
`Strategy5Init`	`Strategy1InitNext`	Connexe	Connexe dès le début car apparaissent au même endroit, après les nodes se barrent de ouf
`Strategy5Init`	`Strategy2Next`	Connexe	Connexe et ne bouge pas
`Strategy5Init`	`Strategy3InitNext`	Connexe	cf Protocol3
`Strategy5Init`	`Strategy4Next`	Connexe	Trenquille dans le scope
`Strategy6Init`	`Strategy1InitNext`	Connexe	‘protocol.emitter.latency 0‘
`Strategy6Init`	`Strategy2Next`	Connexe	‘protocol.emitter.latency 0‘
`Strategy6Init`	`Strategy3InitNext`	Connexe	‘protocol.emitter.latency 0‘
`Strategy6Init`	`Strategy4Next`	Connexe	‘protocol.emitter.latency 0‘

1.9 Question 10

10 itérations

Portee	SPI	SD	D	E/D	ED/D
125	1	1	0.0900 +- 0.0148	2.0340 +- 0.2968	0.1940 +- 0.0220
250	1	1	0.3430 +- 0.0344	0.9840 +- 0.0822	0.1850 +- 0.0242
375	1	1	0.7130 +- 0.0447	0.6550 +- 0.0457	0.1790 +- 0.0230
500	1	1	1.1980 +- 0.0473	0.5700 +- 0.0460	0.2040 +- 0.0265
625	1	1	1.7760 +- 0.0967	0.4730 +- 0.0581	0.2120 +- 0.0366
750	1	1	2.2740 +- 0.1165	0.4630 +- 0.0518	0.2320 +- 0.0325
875	1	1	2.7320 +- 0.1012	0.4240 +- 0.0600	0.2270 +- 0.0514
1000	1	1	3.2600 +- 0.0892	0.4600 +- 0.0632	0.2740 +- 0.0541
125	3	3	2.7820 +- 0.1310	0.4280 +- 0.0787	0.2630 +- 0.0639
250	3	3	2.5410 +- 0.0575	0.4630 +- 0.0520	0.2480 +- 0.0366
375	3	3	2.3370 +- 0.1246	0.4380 +- 0.0371	0.2200 +- 0.0241
500	3	3	2.3830 +- 0.0684	0.4320 +- 0.0426	0.2300 +- 0.0335
625	3	3	2.3250 +- 0.0745	0.4730 +- 0.0473	0.2450 +- 0.0472
750	3	3	2.3170 +- 0.0714	0.4590 +- 0.0532	0.2330 +- 0.0484
875	3	3	2.3390 +- 0.1023	0.4400 +- 0.0369	0.2140 +- 0.0201
1000	3	3	2.3450 +- 0.1551	0.4720 +- 0.0654	0.2480 +- 0.0531

100 itérations

Portee	SPI	SD	D	E/D	ED/D
125	1	1	0.0953 +- 0.0155	1.9218 +- 0.2210	0.1943 +- 0.0241
250	1	1	0.3441 +- 0.0302	0.9882 +- 0.0791	0.1949 +- 0.0296
375	1	1	0.7277 +- 0.0490	0.6890 +- 0.0724	0.1982 +- 0.0409
625	1	1	1.7340 +- 0.0904	0.4903 +- 0.0584	0.2188 +- 0.0418
500	1	1	1.1966 +- 0.0595	0.5586 +- 0.0468	0.2049 +- 0.0278
750	1	1	2.2763 +- 0.0774	0.4570 +- 0.0616	0.2306 +- 0.0435
875	1	1	2.7883 +- 0.0979	0.4585 +- 0.0743	0.2551 +- 0.0641
500	3	3	2.3528 +- 0.0868	0.4443 +- 0.0459	0.2220 +- 0.0379
1000	1	1	3.2539 +- 0.0870	0.4583 +- 0.0753	0.2760 +- 0.0793
125	3	3	2.8025 +- 0.1088	0.4299 +- 0.0680	0.2435 +- 0.0559
250	3	3	2.4760 +- 0.0980	0.4381 +- 0.0446	0.2293 +- 0.0342
625	3	3	2.3267 +- 0.1107	0.4584 +- 0.0600	0.2376 +- 0.0425
375	3	3	2.3910 +- 0.1285	0.4527 +- 0.0567	0.2355 +- 0.0429
750	3	3	2.3350 +- 0.1005	0.4686 +- 0.0626	0.2413 +- 0.0490
875	3	3	2.3451 +- 0.0853	0.4703 +- 0.0600	0.2500 +- 0.0516
1000	3	3	2.3639 +- 0.0991	0.4614 +- 0.0698	0.2379 +- 0.0506

A Compilation, lancement du code et jeux de test

A.1 src/Makefile

Un fichier `Makefile` est inclus dans le dossier `src`. Celui-ci reconnaît les directives :
— `make` compile le projet.
— `make run` lance une instance de simulation telle que spécifiée dans `src/manet/cfg_initial.txt`.
— `make clean` nettoie le projet des compilés.
— `make bench_clean` nettoie le dossier `src` des dossiers de résultats des benchmarks.

Le `Makefile` admet par ailleurs deux variables :
— `DIR_PEERSIM=<chemin>` : le dossier d'installation de Peersim, qu'il faudra **obligatoirement** soit modifier, soit spécifier dans `make` et `make run`.
— `CFG=<chemin>` : le chemin d'un fichier de configuration Peersim, initialisé à `src/manet/cfg_initial.txt` par défaut.

A.2 src/bench.pl

Exemple : `./bench.pl <chemin_peersim>`
Le script crée un dossier `bench_<date>` où seront stockés les résultats pour la question 8 de l'exercise 1. Le dossier contiendra les fichiers de configuration pour les expériences sous le nom `cfg_bench_<scope>_<SPI>_<SD>`, les résultats d'autres fichiers, de même nom avec l'extension `.result`. Ces derniers contiennent les résultats de 10 expériences avec autant de différentes graines aléatoires.

A.3 src/bench.py

exemple : `bench.py <chemin_results>`
Dans le dossier `<chemin_results>` doivent se trouver les fichiers résultats créés par `bench.pl`. Utilisé pour les resultats de benchmark, pour remplir les tableaux.
Exemple de sortie :

```
/ara/src/results/cfg_bench_500_3_3.result
| 2.3830 +- 0.0684 | 0.4320 +- 0.0426 | 0.2300 +- 0.0335 |

/ara/src/results/cfg_bench_875_3_3.result
| 2.3390 +- 0.1023 | 0.4400 +- 0.0369 | 0.2140 +- 0.0201 |

/ara/src/results/cfg_bench_750_1_1.result
| 2.2740 +- 0.1165 | 0.4630 +- 0.0518 | 0.2320 +- 0.0325 |

/ara/src/results/cfg_bench_375_1_1.result
| 0.7130 +- 0.0447 | 0.6550 +- 0.0457 | 0.1790 +- 0.0230 |
```

Les valeurs correspondent à des moyennes avec écarts-type, obtenus sur 10 itérations sur chaque combinaison de SPI et SD.