Eficiência Energética em Redes de Sensores sem Fios

Relatório Final

Agentes e Inteligência Artificial Distribuída

Mestrado Integrado em Engenharia Informática e Computação

T11_1
João Guilherme Routar de Sousa - ei12042@fe.up.pt
Luís Telmo Soares Costa - ei08089@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

21 de Dezembro de 2016

Conteúdo

1 Objetivo

1.1 Descrição do Cenário

No ãmbito da unidade curricular de Agentes e Inteligência Artifial Distribuída, propusémo-nos a desenvolver um programa, cujo objetivo é **otimizar a eficiência energética em redes de sensores sem fios** (*WSN - Wireless Sensor Networks*).

A instalação das *WSN* em locais remotos e/ou inacessíveis implica que a sua substituição seja extremamente difícil (ou até mesmo impossível) e custosa. Como tal, é **essencial** que o consumo energético da *network* seja o **mínimo possível**. Esta otimização apenas poderá ser atingida através da utilização de algoritmos de perceção e comunicação, estabelecendo, idealmente, **interações entre os agentes** (sensores) envolvidos.

Por outro lado, é essencial que a informação captada pelos sensores seja fidedigna. Aliar a qualidade da mesma à eficiência energética é o *core* deste trabalho. A figura seguinte esquematiza o que foi mencionado anteriormente.

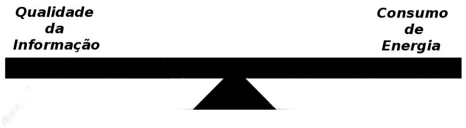

Figura 1: balanço

1.2 Objetivo do Trabalho

Tradicionalmente, os processos de transmissão de dados através de sensores são rudimentares, na medida em que não contemplam a eficiência energética da rede sensorial, apesar de assegurarem a qualidade da informação recolhida. Como tal, o desenvolvimento de métodos de Inteligência Artificial, nomeadamente Sistemas Multi-Agentes (*MSA*) é essencial para que a eficiência energética da rede seja um fator a ter em conta.

Consequentemente, implementámos um algoritmo de **formação de coligações (*CF*)**, denominado ***COSA***, ***Coalition Oriented Sensing Algorithm*** que, para além da vertente básica, inclui outra estratégia, ***sampling frequency*** (***COSA-SF***). Esta última, permite a variação da frequência com que as amostras são recolhidas pelos sensores.

Este algoritmo estabelece as coligações entre nós (sensores) mediante negociação *peer-to-peer*, sendo que a estrutura obtida dependerá diretamente do tipo da rede formada, o estado dos nós e o ambiente para cada instante da análise.

Para o caso em questão, o *environment* é representado por um rio com dimensões 500x20 *px*, em que os sensores podem ser distrbuídos de 3 formas diferentes:

- Uniformemente ao longo do rio
- Aleatoriamente
- Fim do rio

2 Especificação

2.1 Descrição dos Agentes

A aplicação contempla 2 tipos de agentes. O ***Sensor*** e o ***Sink Node Agent***.

Os agentes *Sensor* constituem a rede sensorial instalada no *environment* (rio) responsável pela recolha de amostras (de poluição), enquanto que o agente *Sink Node*, situado no início do rio, é responsável por gerir a informação recebida.

2.2 Arquitetura

2.2.1 Sensor Agent

A tabela seguinte carateriza os parâmetros dos agentes do tipo ***Sensor***:

Parâmetro	**Descrição**
x, y	Coordenadas da posição do sensor no rio
color	Cor (coligação ou livre)
status	Estado (on, sleep, off)
battery	Nível de energia atual
stdDev	Desvio padrão do sensor
maxAdh	Aderência máxima registada
maxLead	Liderança máxima registada
leader	Boleano para diferenciar lider-dependente
leaderSensor	Identificador do líder do sensor
sleepCounter	Coordenadas da posição do sensor
strategy	Define estratégia do algoritmo (variação da frequência da recolha de amostras: COSA, COSA_SF)
sinkNode	Identificador do *sink node*
pollutionSamples	Amostras (de poluição) recolhidas pelo agente
neighboursLastSampleMap	Estrutura que mapeia amostras (de poluição) dos vizinhos do sensor
neighboursAdherenceMap	Estrutura que mapeia a aderência dos vizinhos do sensor
dependantNeighbours	Lista de vizinhos dependentes do sensor

2.2.2 SinkNode Agent

A tabela seguinte carateriza os parâmetros dos agentes do tipo ***SinkNode***:

Parâmetro	**Descrição**
x, y	Coordenadas da posição do *sink* no rio
samplesPollutionLevelsSum	Somatório dos valore de poluição recebidos
actualPollutionLevelsSum	Somatório dos nível atuais de poluição
lastSamplePollutionLevels	Valor de poluição da última amostra enviada

2.3 Comportamento

O **MAS** é constituído por agentes com um comportamento padrão de amostragem diferente do normal, por consequência da implementação do *COSA*. Tal implica que os agentes sejam **autónomos**, **proativos** e **reativos**, enquadradando-se na arquitetura de **agentes reativos simples**.

Os agentes ***Sensor*** são caraterizados por um *behaviour* assegurado através de um protocolo de negociação simples, aliado a funções de modelação entre os nós: **aderência** e **liderança**.

Os resultados obtidos determinam o diálogo assíncrono protagonizado pelos agentes enquanto negoceiam entre si.

O esquema seguinte demonstra o comortamento dos agentes ***Sensor*** na rede:

COSA: Sensor basic behaviour
while *battery >0* ***do***
environment sampling;
environment model update;
to neighbours update
social network update
end

Por outro lado, o agente ***SinkNode*** é apenas responsável pela receção de mensagens (com informações de poluição), com o objetivo de calcular o erro associado à qualidade da informação.

O esquema seguinte demonstra o comortamento do agente ***SinkNode*** na rede:

COSA: SinkNode basic behaviour
while *battery >0* ***do***
environment sampling;
sensor sample reception;
error calculation;
end

2.4 Protocolos de Interação

Os nós da rede sensorial (sensores) recolhem amostras do ambiente envolvente (amostras de poluição) que são enviadas, posteriormente, através de ***ACL Messages***, para os vizinhos envolventes.

Após a receção das *samples*, **os destinatários trocam valores de aderência e liderança com o remetente**, sendo avaliada a necessidade de formar uma coligação num modelo **bottom-up**.

Em caso afirmativo, o sensor remetente torna-se o **líder** da coligação, sendo que o(s) restante(s) são considerados **dependentes** e, como tal, poderão **cessar a sua atividade** (incluindo recolha de amostras e comunicação com o *sink*.

É essencial que a organização dos nós seja bem estruturada, de forma a ser possível rentabilizar energia, derivado da poupança de transmissões de longa distância (as que os nós dependentes não têm que fazer).

Paralelamente, todos os sensores não dependentes, ou seja, em atividade, **enviam continuamente *samples* para o *sink***. Este, que se encontra no início do rio, é responsável pelo **cálculo do erro** e **qualidade da informação** transmitida ao longo do processo. Para esta aplicação, apenas foi feito o cálculo do erro, através da seguinte fórmula:

$$e^t = \sum_{i \in N^t} || \, xs_i^t - xp_i^t \, ||$$

As imagens seguintes sintetizam todas as interações que ocorrem na aplicação entre os sensores e o *sink*:

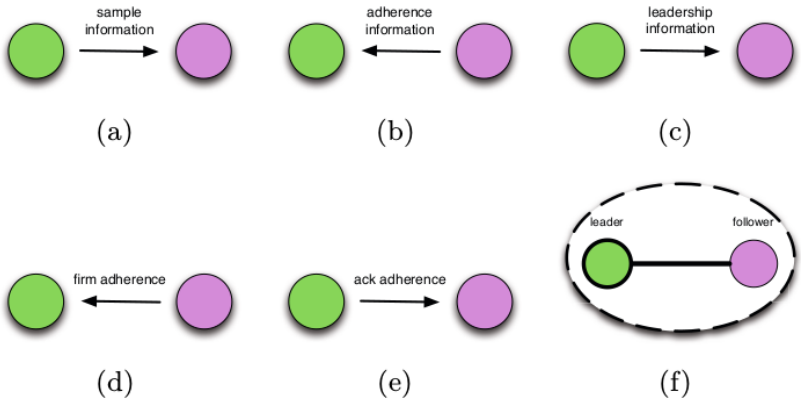

Figura 2: Interações entre sensores e sink (paper)

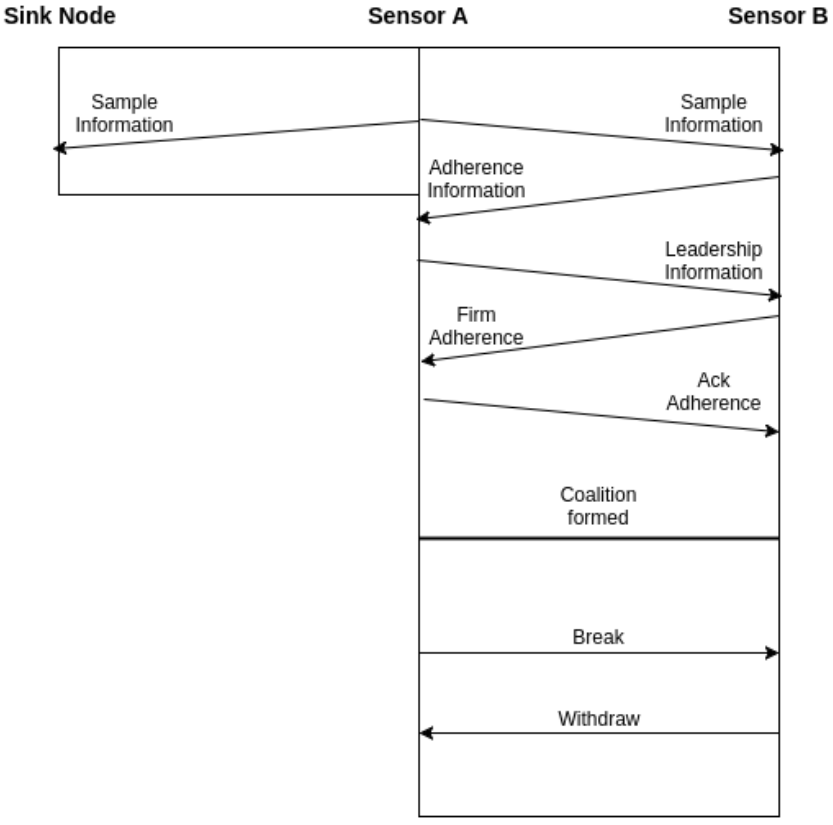

Figura 3: Interações entre sensores e sink

Por cada interação entre sensores, é essencial que o ambiente (dinâmico) e os próprios sensores sejam atualizados. A imagem seguinte, fornecida pelo *paper*, esquematiza os processos envolventes:

Algorithm 2. Message Processing

Data: me: focus node; a_j: generic neighbour; a_l: potential leader; a_r: potential dependant on me; a_p: dependant node on me; a_L: leader node of me; $D(me)$: set of dependant nodes on me

```
1  case rcvd(inform(a_j, me, meas, t))
2      updateNeighbourInfo();
3      adherence2NeighbourEvaluation();
4      updateOwnMaxAdherence();
5      if changesOnOwnMaxAdherence
       then
6          inform(me, a_l, maxAdh, t);
7      end
8  end
9  case rcvd(inform(a_j, me, maxAdh, t))
10     inform(me, a_r, lead);
11     updateNeighbourInfo();
12     adherence2NeighbourEvaluation();
13     updateOwnMaxAdherence();
14     if changesOnOwnMaxAdherence
       then
15         inform(me, a_l, maxAdh, t);
16     end
17 end
18 case rcvd(inform(a_l, me, lead))
19     if checkAgainstOwnLead then
20         firmAdherence(me, a_l);
21     end
22 end
23 case rcvd(firmAdherence(a_r, me))
24     if checkAgainstOwnLead then
25         ackAdherence(me, a_r);
26         updateOwnLeadValue();
27         updateDependentGroup();
28     end
29 end
30 case rcvd(ackAdherence(a_l, me))
31     if !leader ∧ a_l! = a_L then
32         withdraw(me, a_L);
33     end
34     if leader ∧ D(me)! = ∅ then
35         while D(me)! = ∅ do
36             break(me, a_p);
37         end
38     end
39     updateRoleState(dependant);
40     sleep(t);
41 end
42 case rcvd(break(a_L, me))
43     updateRoleState(leader);
44 end
45 case rcvd(withdraw(a_p, me))
46     D(me) ← D(me)\a_p;
47     updateRoleState(leader);
48 end
```

Figura 4: Interações entre sensores e sink

3 Desenvolvimento

3.1 Plataforma e Ferramentas

3.1.1 JADE

O **J**ava **A**gent **D**evelopment **E**nvironment é um *Middleware open-source* que facilita o desenvolvimento de *MAS*, sistemas multi-agente, sob as especificações *FIPA* (*Foundation for Intelligent Physical Agents*). É uma plataforma de agentes distribuídos, onde estes estão restritos a um determinado domínio. Para além disso, possibilita a execução paralela dos comportamentos dos agentes *behaviours* e mobilidade e clonagem dos mesmos.

Esta ferramenta foi essencial na implementação das interações entre os sensores e entre os sensores e o *sink*. Como tal, permitiu a utilização das *ACLMessages* e outros processos.

3.1.2 Repast 3 + SAJaS (com MASSim2Dev)

O **Repast 3** é uma plataforma *open-source*, *object-oriented* que permite fazer simulações baseadas em agentes. Para além disso, possibilita a acessibilidade e modelação dos mesmos em *run time*. Apenas complementa o *JADE*, dado que não segue as especificações *FIPA* e não permite o desenvolvimentos de *MAS*.

O **SAJaS** é uma *API* para simulações baseadas em *JADE*. O seu objetivo é interligar o desenvolvimento e a simulação *MAS*.

Ambas as plataformas contribuíram para o desenvolvimento de uma interface gráfica, capaz de traduzir as necessidades da aplicação, nomeadamente no desenvolvimento de um *environment* (rio) e dos próprios sensores e *sink*.

A ferramenta ***MASSim2Dev*** permitiu a conversão da simulação baseada em *JADE* para *SAJaS*.

3.2 Ambiente de Desenvolvimento

A aplicação foi desenvolvida no Sistema Operativo ***Ubuntu 16.04 LTS (Xenial Xerus)*** com o *IDE* ***Eclipse Neon .1***

3.3 Estrutura da Aplicação

A tabela e a figura seguinte mostram como está estruturada a aplicação, sendo que é constituída por 5 *packages*:

Pacote	Subpacote	Ficheiros	Descrição
ACLMsgContentObjects	CANCEL	Cancel.Java	Classe mãe de Break e Withdraw
		Break.Java	Conteúdo para mensagens enviadas com o intuito de quebrar a coligação
		Withdraw.Java	Conteúdo para mensagens enviadas com o intuito de sair da coligação
	INFORM	Inform.Java	Classe mãe de Adherence, Leadership e Sample
		Adherence.Java	Conteúdo para mensagens enviadas com o intuito de enviar valores de aderência
		Leadership.Java	Conteúdo para mensagens enviadas com o intuito de enviar valores de liderança
		Sample.Java	Conteúdo para mensagens enviadas com o intuito de enviar amostras de poluição
Agents	-	Sensor.Java	Gere o comportamento dos sensores (classe filha de Agent)
		SinkNode.Java	Gere o comportamento do *sink* (classe filha de Agent)
COSA_Calculations	-	COSA.java	Gere os cálculos associados ao COSA
Environment	-	Water.java	Gere o comportamento do rio e poluição
SIMLauncher	-	SIMLauncher.Java	Representa o modelo da simulação

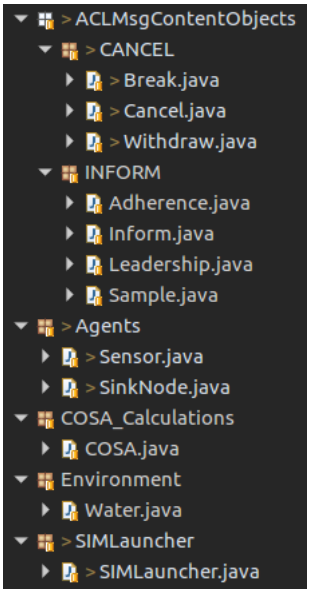

Figura 5: Estrutura da Aplicação

3.4 Diagrama de Classes

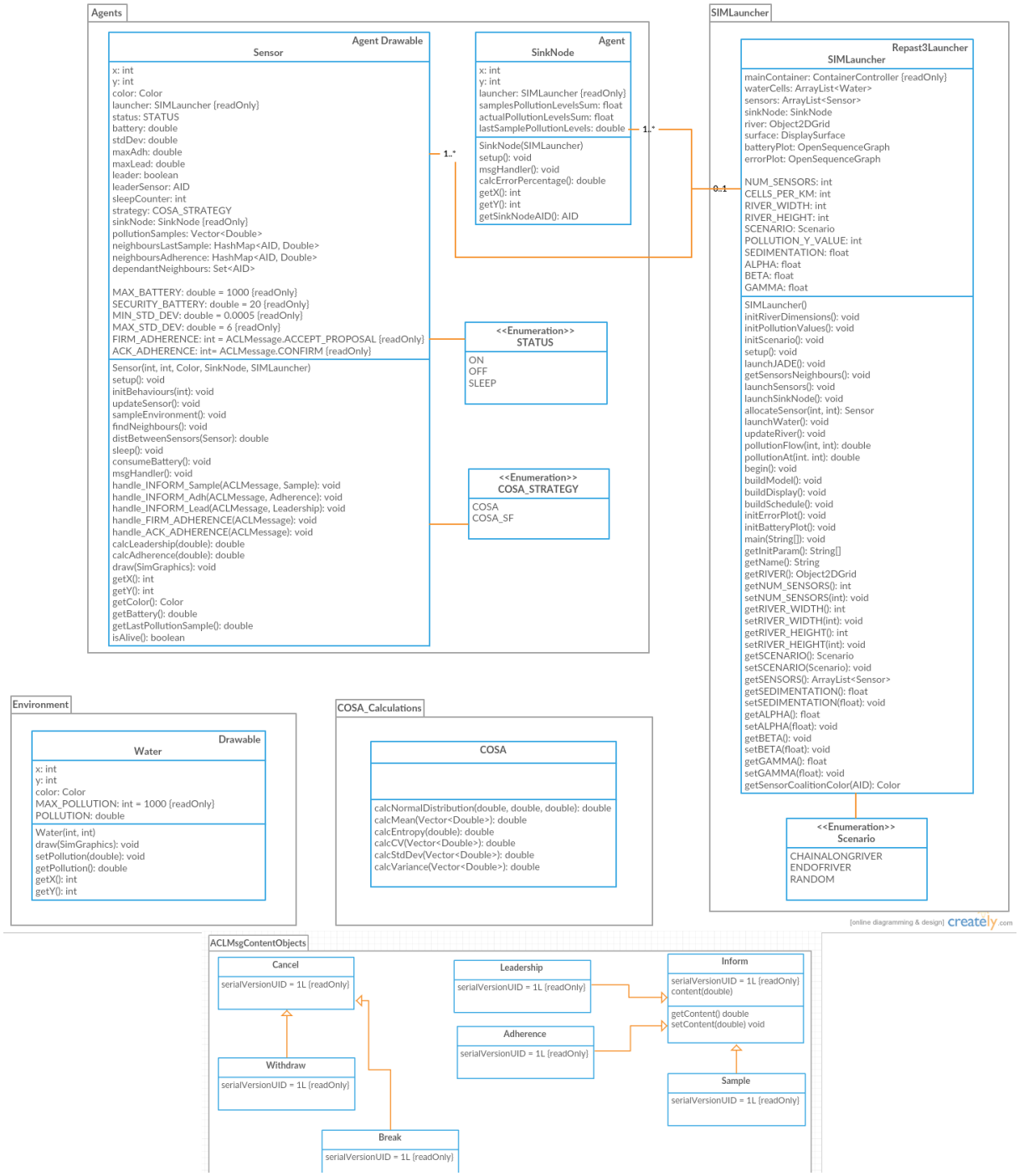

Figura 6: Diagrama de classes UML. Link original: https://s27.postimg.org/9m8jfra37/uml.png

3.5 Detalhes Relevantes da Implementação

Como referido anteriormente, foi utilizado o SAJaS para a representação gráfica da simulação. Como tal, assim que a aplicação corre, são desenhados 2 painéis. Um deles permite **reproduzir, parar, pausar ou fazer *step*** da simulação. O outro painel, contendo os parâmetros, permite regular os mesmos de acordo com as preferências do utilizador. É possível **alterar os valores da dimensão do rio, do fluxo e sedimentação do mesmo, o cenário** (de distribuição dos sensores). A figura seguinte apresenta o painel de simulação:

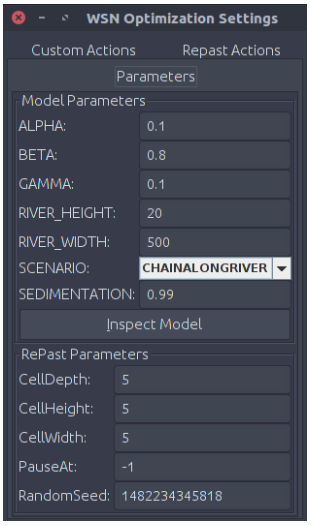

Figura 7: Painel da simulação

A simulação em si é constituída por 3 elementos fundamentais:

- *Environment* (rio, poluição, sensors e *sink*)
- Gráfico Bateria (mediana da bateria dos sensores)
- Gráfico Erro (associado à recolha de amostras)

Environment

O rio é constituído por células de ***Water*** com cor azul, que são distribuídas pelas dimensões pré-definidas (500x20). O fluxo do rio segue a seguinte distribuição:

$$River(x,y) = (1-p)River(X,Y)+p(a(River(X-1,Y-1))+B(River(X,Y-1))+y(River(X+1,Y-1)))$$

A **poluição** corresponde, igualmente, **a células de *Water*, que têm uma cor adulterada**, seguindo uma **distribuição sinusoidal**.

Os sensores são desenhados de acordo com a distribuição pretendida (ao longo do rio, aleatório ou fim do rio), sendo que o desenho do *sink* é *hardcoded* para ser mais percetível a sua visualização (SN).

A simulação tem o aspeto demonstrado na figura seguinte:

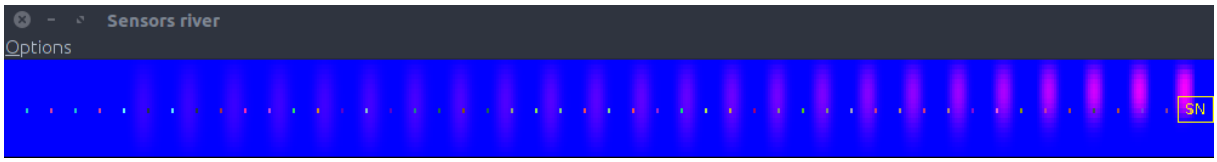

Figura 8: Ambiente da simulação

Gráfico Bateria

O gráfico bateria calcula a mediana da bateria dos sensores em tempo real, apresentando um gráfico com a variação para efeitos de análise da simulação. A figura seguinte demonstra a mediana das baterias num instante da simulação:

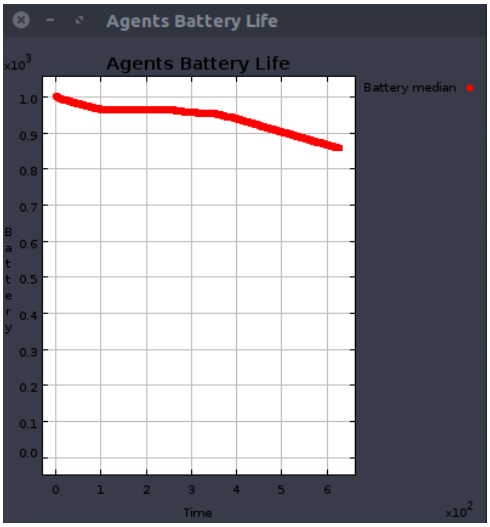

Figura 9: Gráfico da bateria

Gráfico Erro

O gráfico erro calcula o **erro associado à transmissão das amostras de poluição**. Calcula, em tempo real, a **percentagem (%) de erro associada à diferença entre os valores atuais de poluição recolhidos pelo *sink* e os últimos valores de poluição enviados pelos sensores**.

A figura seguinte demonstra o gráfico do erro no instante final da simulação. Os valores apresentados são mais fidedignos, quanto mais próxima está a simulação deste instante.

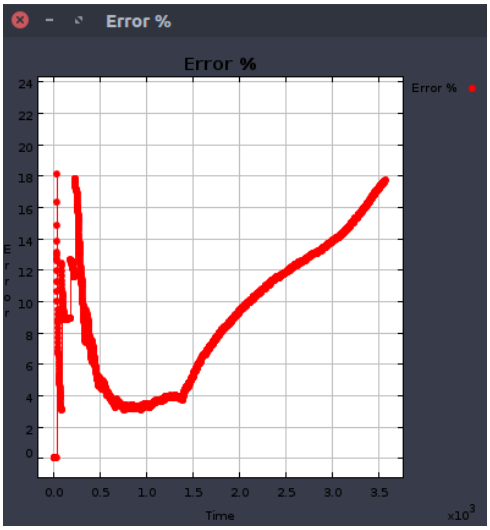

Figura 10: Gráfico do Erro

4 Experiências

Seguidamente, irão ser demonstradas as várias experiências para os diferentes cenários e instantes da simulação. Os parâmetros utilizados para cada exemplo são os apresentados no painel da figura 7.

O algoritmo **COSA_SF** (*sampling frequency*) **altera a frequência de envio de amostras, quando um líder tem pelo menos 4 dependentes**. Como tal, apenas faria sentido testar para o cenário *aleatório*, considerando que os sensores no cenário *ao longo do rio* têm, no máximo, 2 vizinhos e o cenário *fim do rio* implementa o algoritmo *default* (não são formadas coligações).

As experiências são demonstradas de seguida.

4.1 Cenário 1 : Ao Longo do Rio

Neste cenário, os sensores estão distribuídos uniformemente ao longo do rio. São apresentados os resultados para os intantes inicial, intermédio e final.

4.1.1 Instante Inicial

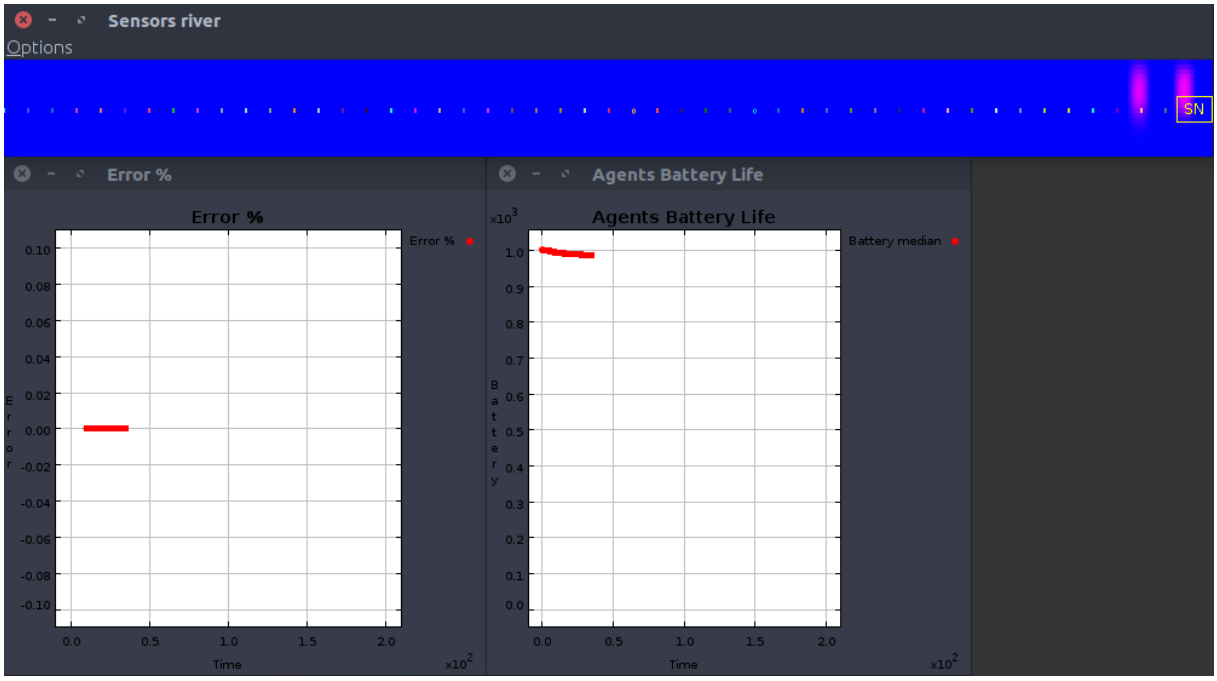

Figura 11: Simulação CHAINALONGRIVER (COSA) no instante inicial

4.1.2 Instante Intermédio

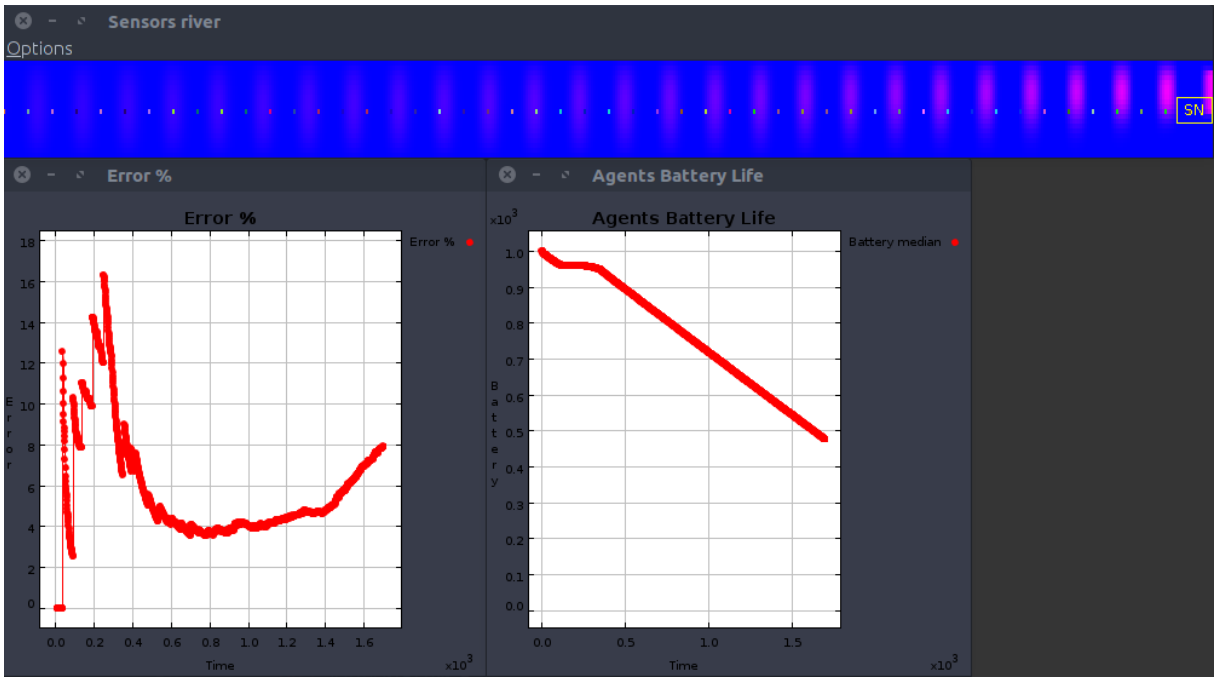

Figura 12: Simulação CHAINALONGRIVER (COSA) no instante intermédio

4.1.3 Instante Final

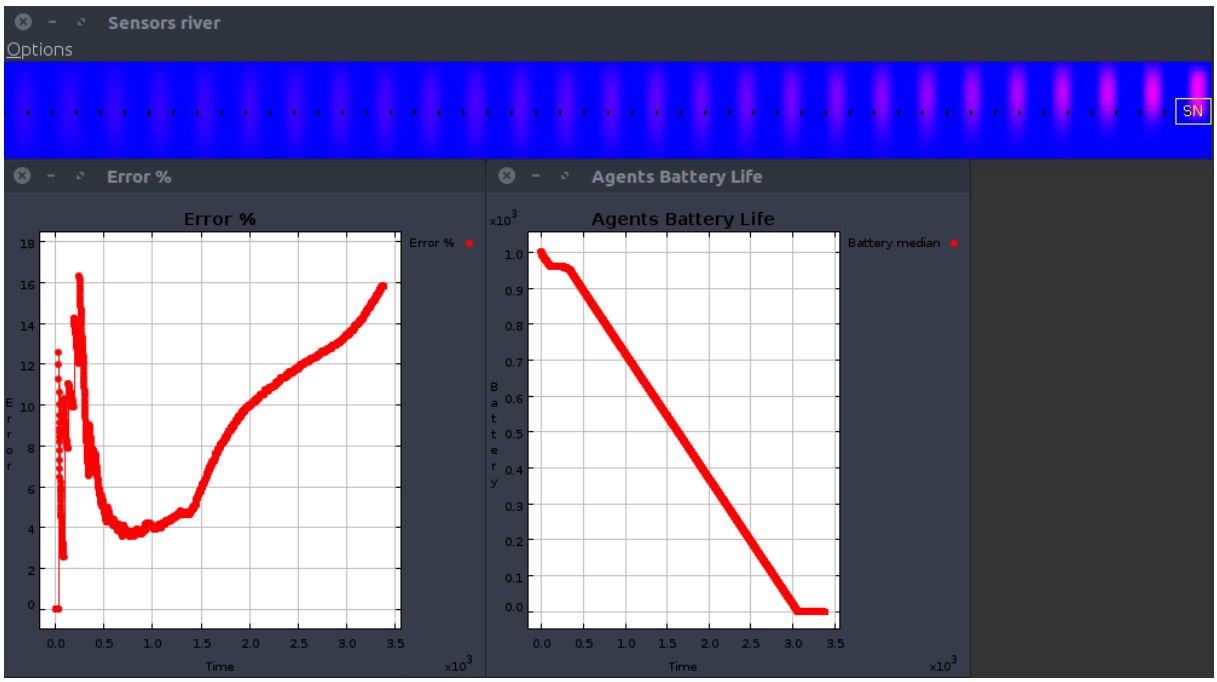

Figura 13: Simulação CHAINALONGRIVER (COSA) no instante final

4.2 Cenário 2 : Aleatório

Neste cenário, os sensores estão distribuídos aleatoriamente ao longo do rio. Dado que corresponde a uma distribuição aleatória, uma simulação apenas não seria conclusivo. Como tal, foram efetuadas 10 experiências aleatórias diferentes, sendo que são apresentados os resultados para 2 das mesmas e uma 3ª para o *COSA-SF*, para os instantes inicial, intermédio e final.

4.2.1 Instante Inicial

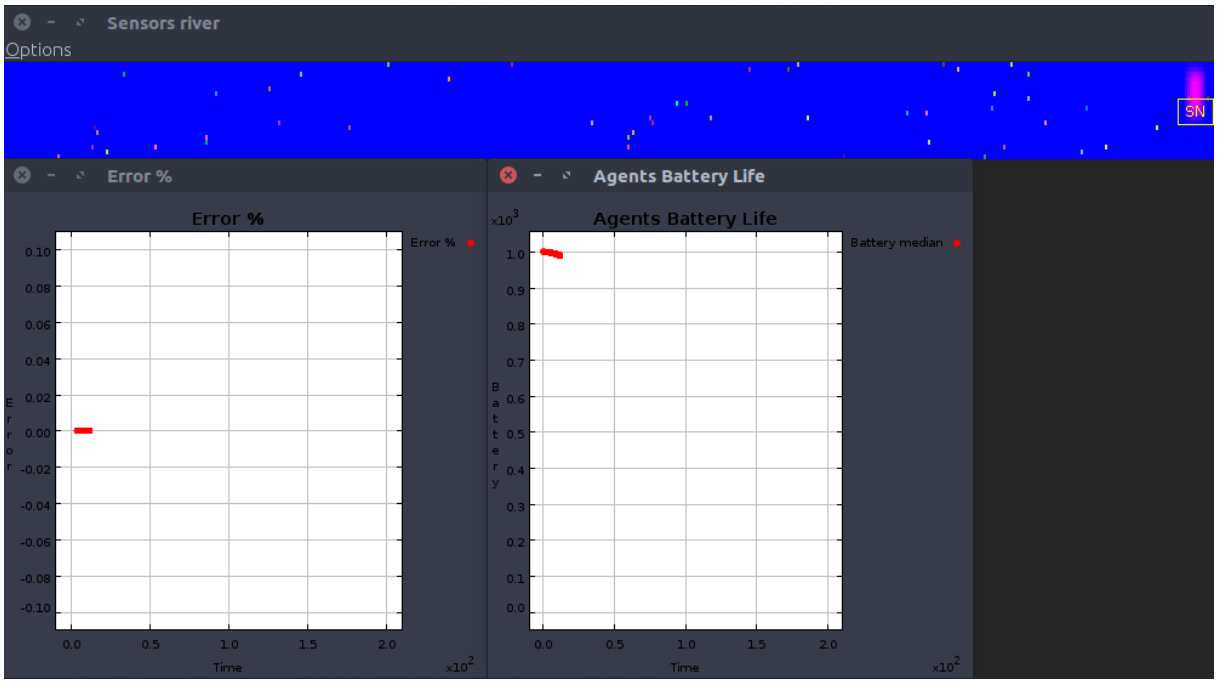

Figura 14: Simulação RANDOM no instante inicial (COSA) - experiência 1

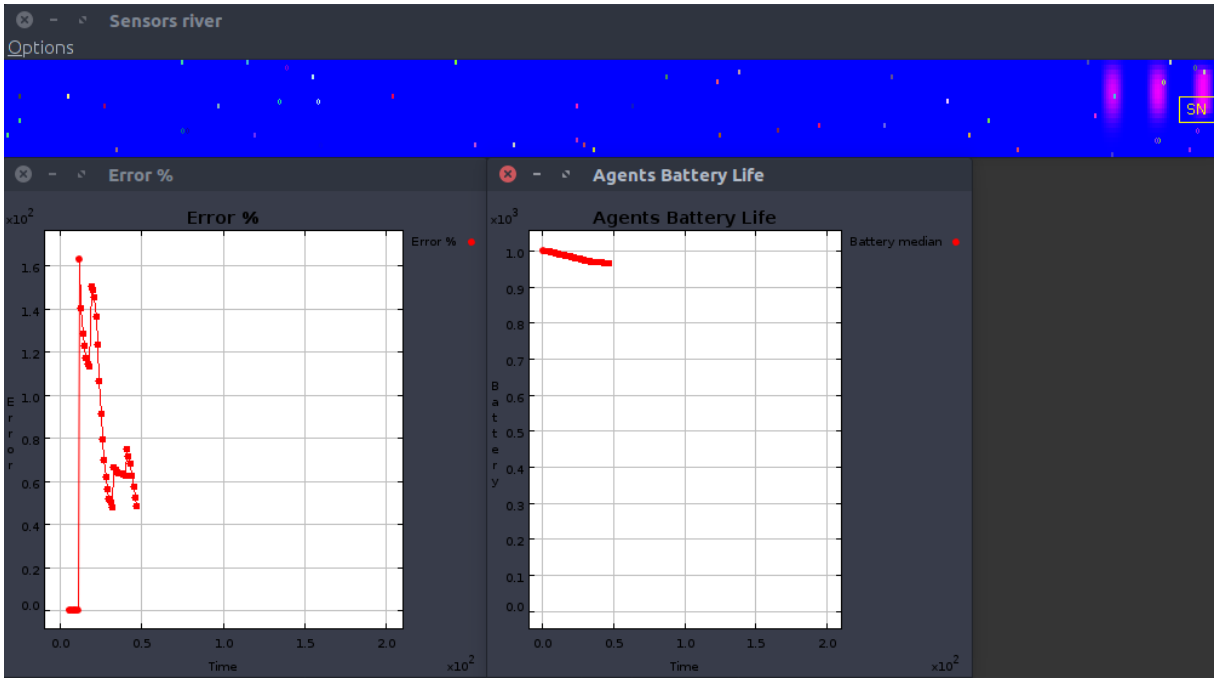

Figura 15: Simulação RANDOM no instante inicial (COSA) - experiência 2

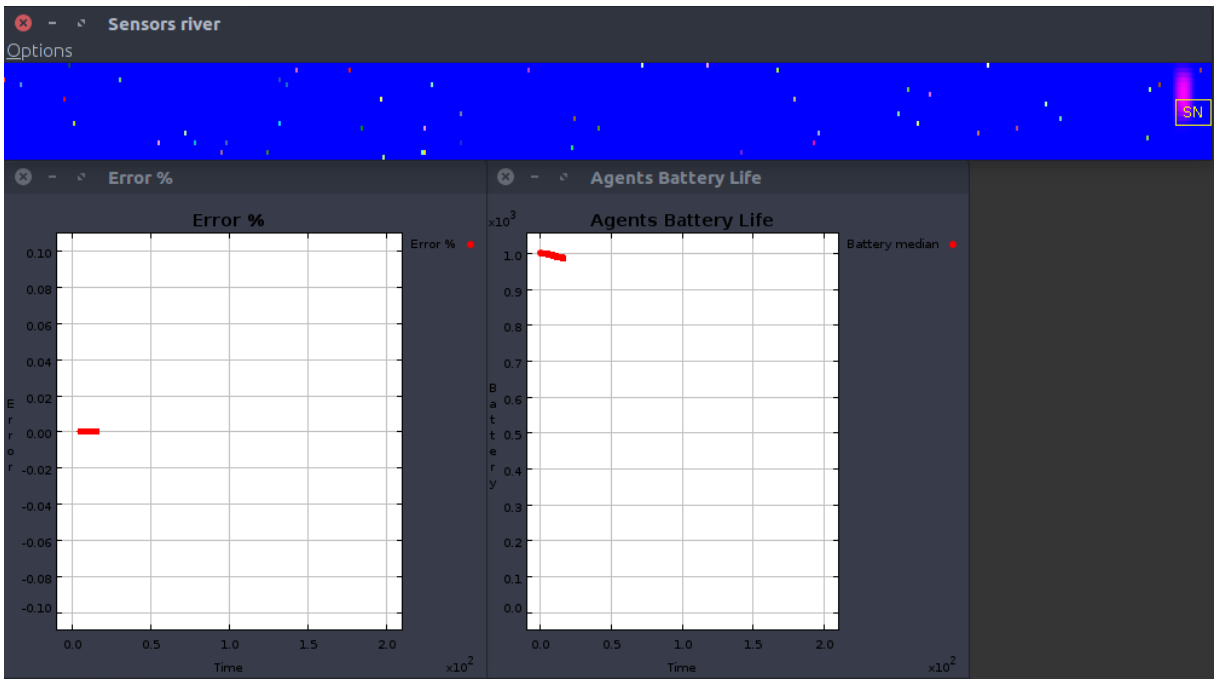

Figura 16: Simulação RANDOM no instante inicial (COSA-SF) - experiência 3

4.2.2 Instante Intermédio

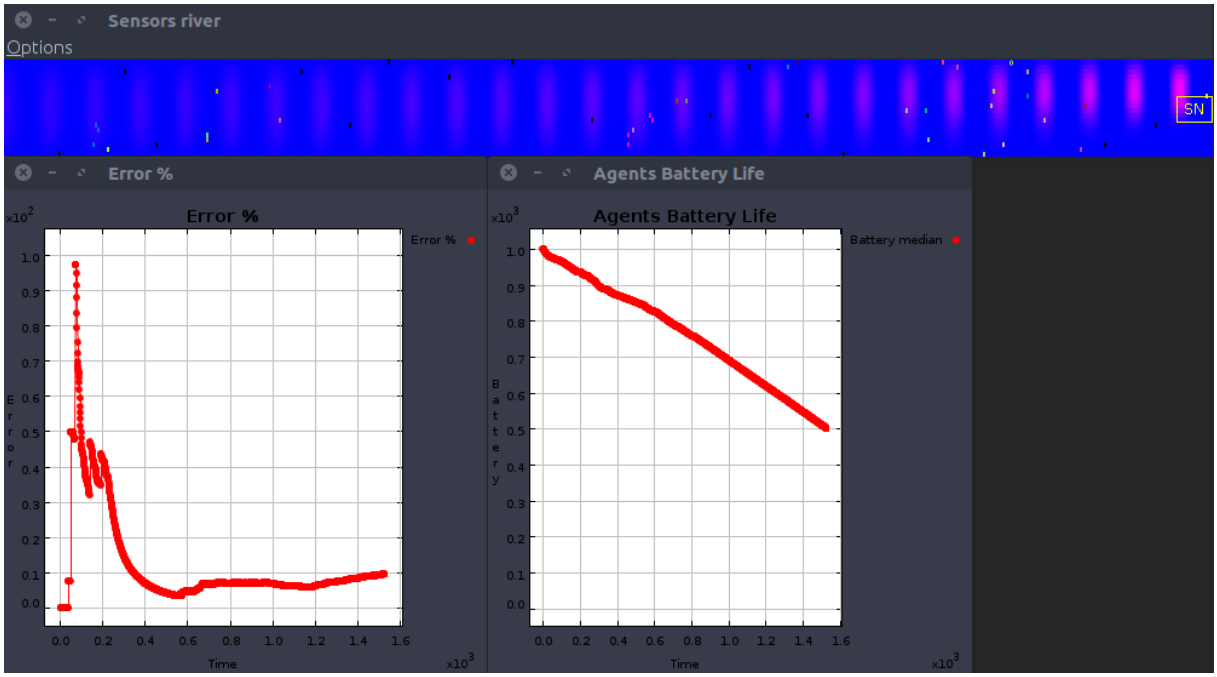

Figura 17: Simulação no RANDOM instante intermédio para o COSA - experiência 1

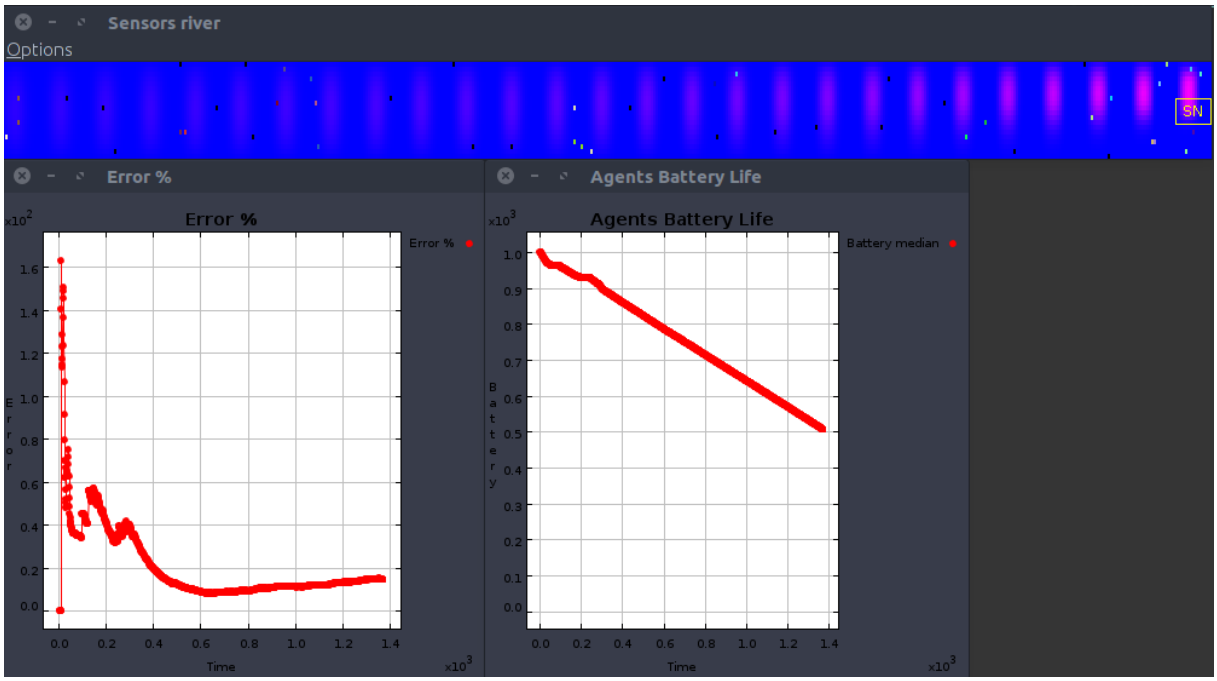

Figura 18: Simulação RANDOM no instante intermédio para o COSA - experiência 2

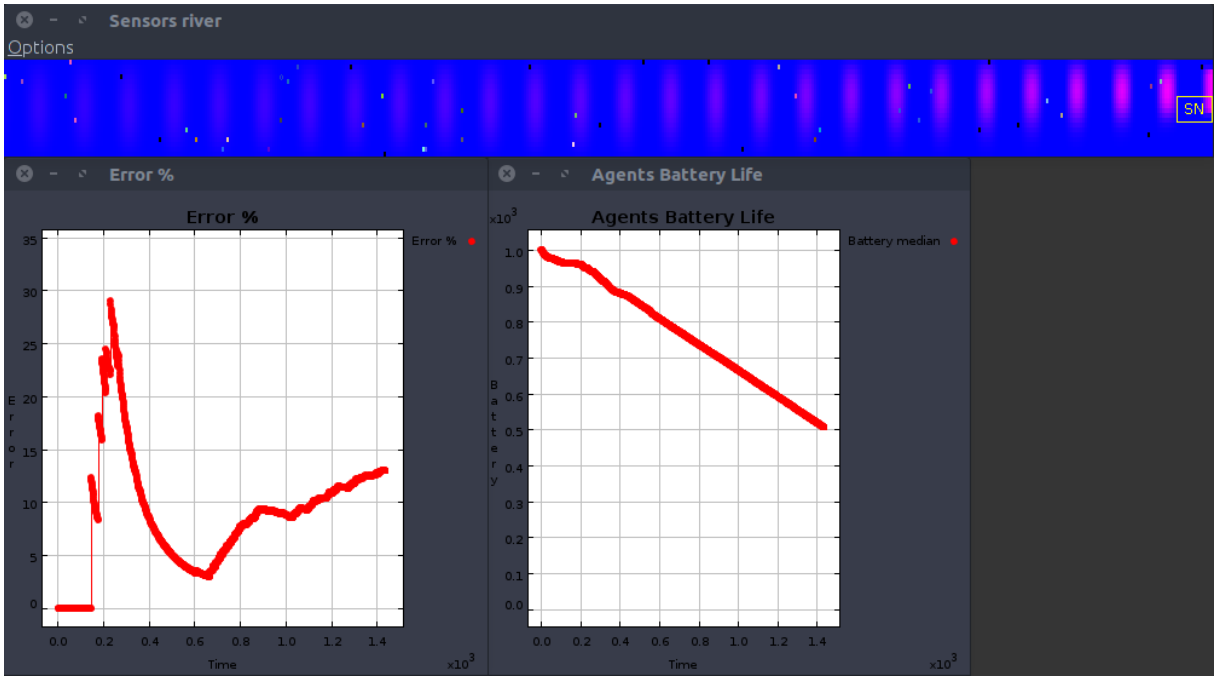

Figura 19: Simulação RANDOM no instante intermédio para o COSA-SF - experiência 3

4.2.3 Instante final

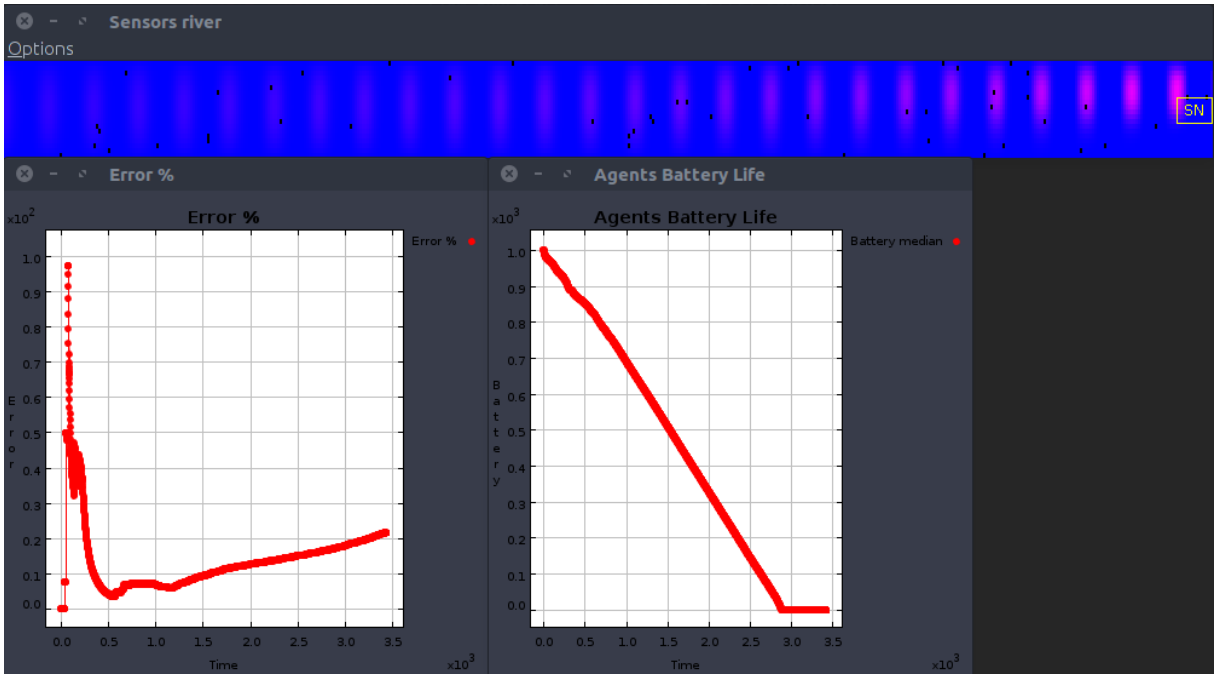

Figura 20: Simulação RANDOM no instante final para o COSA - experiência 1

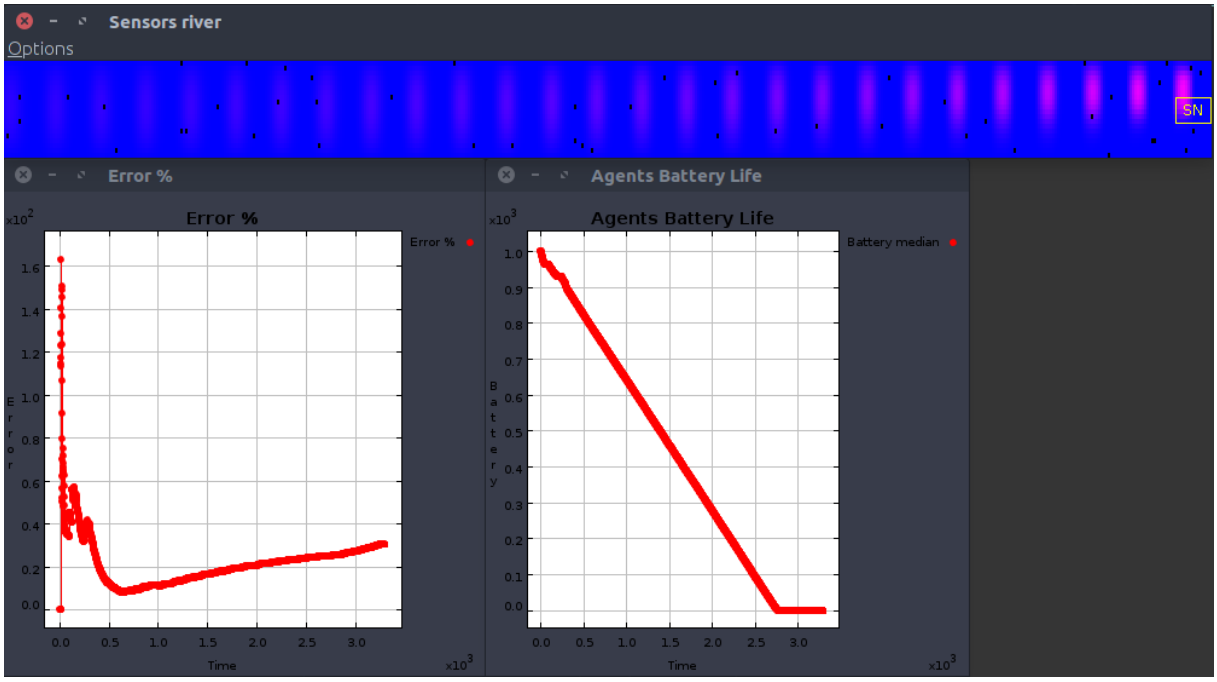

Figura 21: Simulação RANDOM no instante final para o COSA - experiência 2

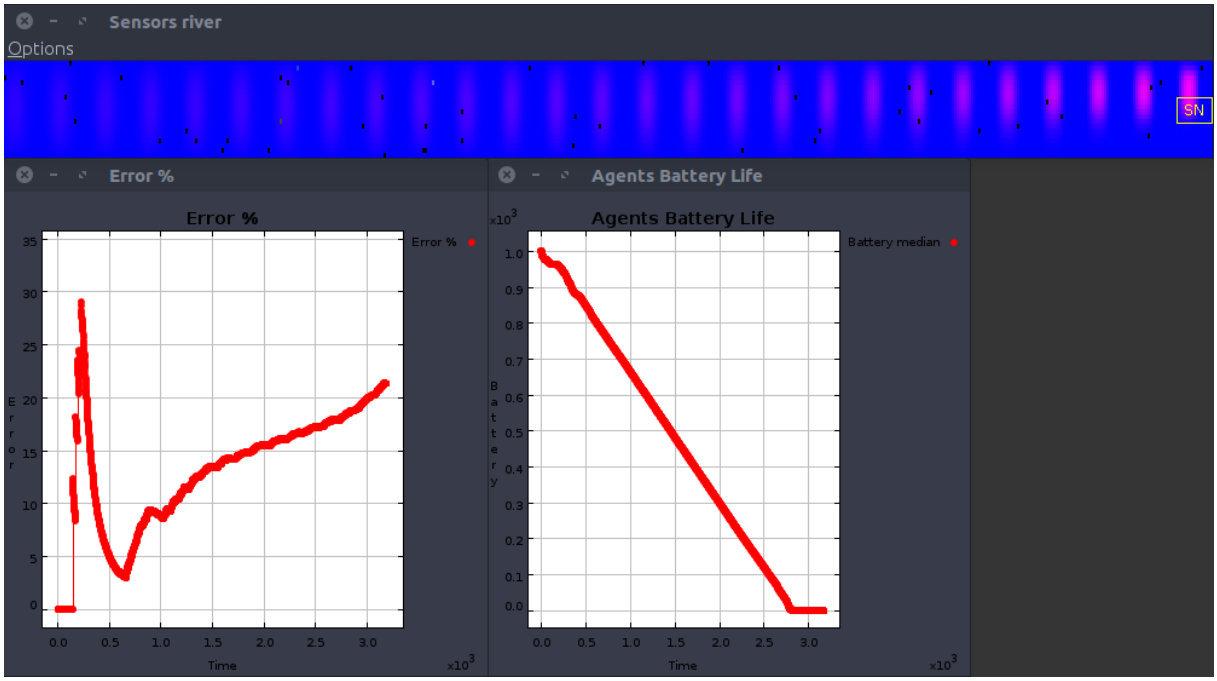

Figura 22: Simulação RANDOM no instante final para o COSA-SF - experiência 3

4.3 Cenário 3 : Fim do rio

Neste cenário, os sensores estão distribuídos em grelha no fim do rio. Estes **não estabelecem coligações entre si**, o que significa que **trabalham autonomamente**. Desta forma, a sua **bateria decresce linearmente**.

Este cenário corresponde aos *algoritmos* básicos utilizados (onde o *COSA* não está incluído).

São apresentados os resultados para os intantes inicial, intermédio e final.

4.3.1 Instante Inicial

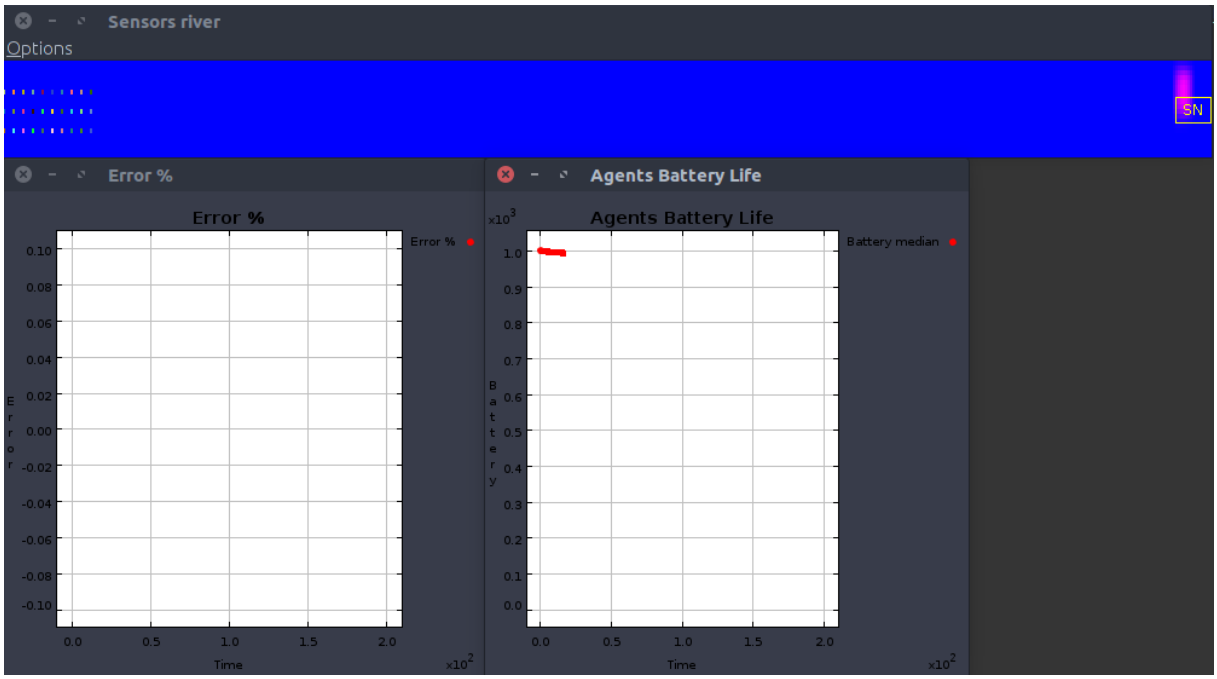

Figura 23: Simulação ENDOFRIVER (Default) no instante inicial

4.3.2 Instante Intermédio

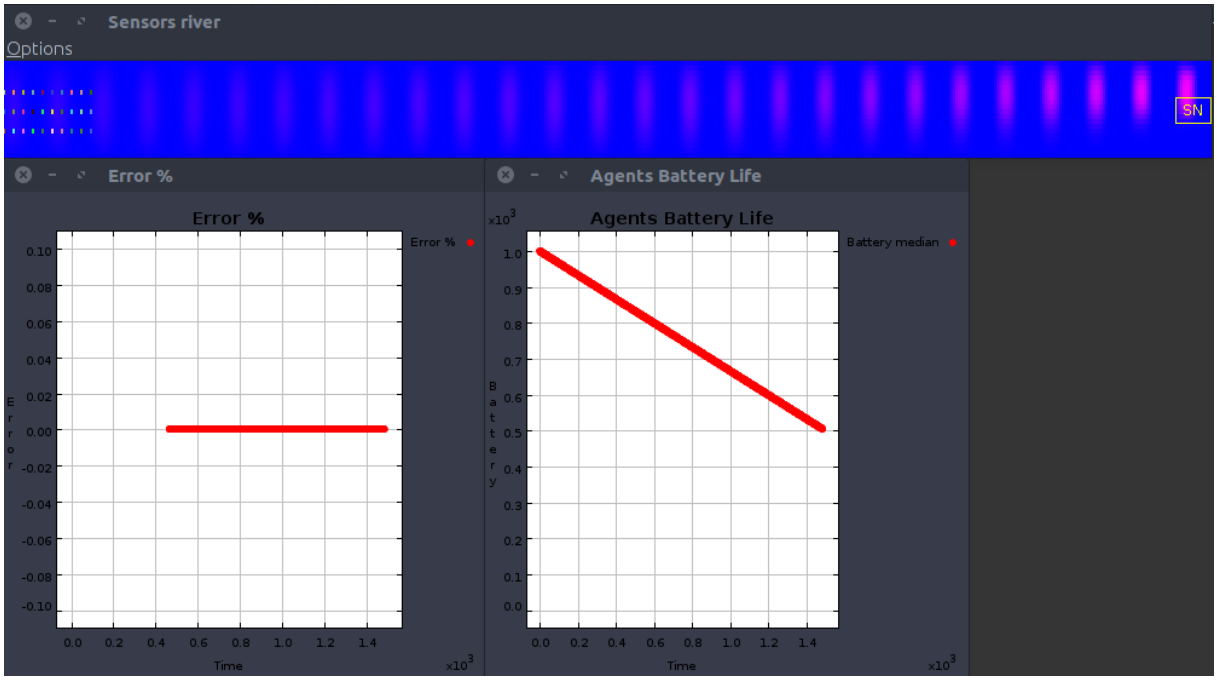

Figura 24: Simulação ENDOFRIVER (Default) no instante intermédio

4.3.3 Instante final

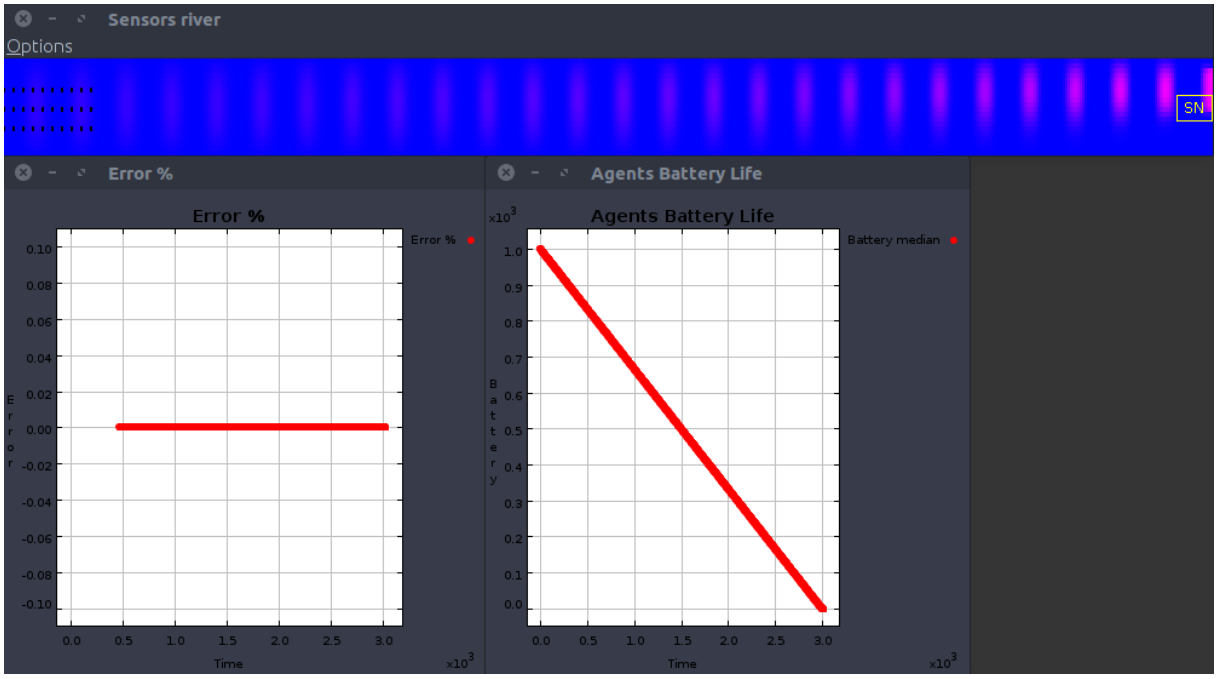

Figura 25: Simulação ENDOFRIVER (Default) no instante final

5 Conclusões

5.1 Análise de Resultados das Experiências

A tabela seguinte sintetiza os resultados das experiências demonstradas na secção anterior:

Algoritmo	**Cenário**	**Mediana da Bateria (ms)**	**Erro (%)**
COSA	CHAINALONGRIVER (ao longo do rio)	$3.4 * 10^3$	16
COSA	RANDOM (aleatório) - Experiência 1	$3.5 * 10^3$	22
COSA	RANDOM (aleatório) - Experiência 2	$3.3 * 10^3$	30
COSA-SF	RANDOM (aleatório) - Experiência 3	$3.3 * 10^3$	22
Default	ENDOFRIVER (fim do rio)	$3 * 10^3$	0

A partir dos resultos apresentados, é possível tirar algumas conclusões. Para o cenário *ao longo do rio* e *fim do rio*, as **conclusões são bastantes precisas, dado que é um ambiente estático**.

Por outro lado, o cenário *aleatório* **não apresenta uma precisão ideal**, dado que a distribuição dos sensores é diferente para cada simulação e seria necessário um elevado número de experiências para chegar a resultados ideais.

Assim, conclui-se que, para **o cenário *fim do rio*, a mediana da bateria apresenta o valor mais baixo (os sensores duram menos tempo) e o menor erro possivel (0%)**. Isto deve-se ao facto de os sensores atuarem **autonomamente, o que significa que nenhum deles estabelece coligações e a bateria decresce linearmente** (morrem todos ao mesmo tempo). Por outro lado, os **valores de poluição enviados são concisos** para cada instante.

Para **o cenário *ao longo do rio* e *aleatório***, que aplicam o algoritmo *COSA*, **a mediana das baterias aumenta consideravelmente**. Aqui, são formadas coligações, o que significa que há um **menor consumo energético**.

Por outro lado, o **erro associado aumenta bastante**, sendo que o objetivo principal é diminuir este e tornar o algoritmo o mais eficiente e eficaz possível (apenas possível com informação mais detalhada do documento).

O cenário *aleatório* apresenta valores energéticos igualmente superiores e inferiores relativamente ao *ao longo do rio*. Isto deve-se, como referido anteriormente, à variação da distribuição dos sensores. **Quando estes duram mais tempo, significa que há um maior número de coligações** e vice-versa. Todavia, o algoritmo *COSA-SF* apresenta valores bastante semelhantes ao *COSA*. Apenas com um maior número de experiências, seria possível concluir que, eventualmente, o ***COSA-SF* apresenta um menor consumo energético, mas um erro maior**.

O erro associado ao cenário *ao longo do rio* é o menor das experiências (para além do algoritmo *default*), apresentando uma mediana da bateria relativamente parecida.

Como tal, **conclui-se que o cenário *ao longo do rio*** (*COSA*) apresenta os valores mais concisos: **Menor erro e eficiência energética semelhante aos restantes**. Por outro lado, à partida, a distribuição ***aleatória* apresentará um menor consumo energético, mas um erro maior**.

5.2 Desenvolvimento e Aplicabilidade

A otimização da eficiência energética (aliada à redução do erro de transmissão) em redes de sensores sem fios é, sem dúvida, um tema extremamente interessante e aplicável a situações da vida real. Como tal, aliar este tema aos conhecimentos da disciplina em questão foi uma mais valia para o desenvolvimento pessoal, profissional e social, na medida em que contribui em grande escala para todos estes fatores.

Todavia, as lacunas existentes no *paper* limitaram o desenvolvimento e a precisão da aplicação, o que criou algumas dificuldades na implementação do algoritmo.

6 Melhoramentos

Existem vários melhoramentos que poderiam ser feitos, parcialmente devido a lacunas do *paper*, tendo em conta um melhor produto final. Estes são apresentados na lista seguinte:

- **Demonstração mais percetível das coligações**

  Apesar das coligações entre sensores serem visíveis através da mesma cor, esta não constitui uma solução ideal. Para além disso, não é possível fazer a distinção entre líderes e dependentes.

- **Maior precisão do algoritmo *COSA***

  Devido a certas lacunas no *paper*, houve certas suposições a ser feitas. Uma melhoria do documento, levaria a que o algoritmo fosse implementado de uma forma mais precisa, de forma a evitar falhas.

- **Cenário *fim do rio***

  Este cenário deveria representar o algoritmo de forma semelhante aos outros, porém os sensores não formam coligações entre si. Por outro lado, o facto de o algoritmo não ser aplicado, faz com que este cenário represente os algoritmos tradicionais aplicados, pelo que foi possível tirar mais conclusões relativamente ao *COSA*. Idealmente, o cenário deveria funcionar de acordo com o esperado e deveria ser implementado o algoritmo base num módulo à parte.

- **Obtenção da qualidade da informação (somatório de entropia dos sensores envolvidos)**

  A informação incompleta e a falta de recursos, fizeram com que este cálculo não fosse possível.

7 Recursos

7.1 Bibliografia

- María del Carmen Delgado; Carles Sierra "A Multi-agent Approach to Energy-Aware Wireless Sensor Networks Organization", Agreement Technologies - Second International Conference, AT. Lecture Notes in Computer Science, vol. 8068, Beijing, China,, Springer, pp. 32-47, 01/08/2013.

7.2 Software

- Ubuntu 16.04 LTS
- Eclipse Neon.1
- JADE
- Repast 3
- SAJaS
- MASSim2Dev

7.3 Elementos do Grupo

- João Guilherme Routar de Sousa - 50%
- Luís Telmo Costa - 50%

8 Apêndice

Para correr a aplicação, basta apenas correr o projeto no Eclipse e clicar no botão *Play* para iniciar a simulação. Os parâmetros da mesma podem ser alterados (antes desta ser iniciada), para uma experiência mais enriquecedora.