

UNIVERSIDADE ESTADUAL DE CAMPINAS  
Instituto de Computação

---

Avaliação do Consumo de Energia em Rede WSN em Ambiente  
Simulado

MO809A Tópicos de Computação Distribuída

---

*Autor:*  
Eduardo Seiti Ito

*Professor:*  
Leandro Villas



**UNICAMP**

Campinas - SP  
Junho de 2019

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Metodologia</b>	<b>2</b>
2.1	NS3 . . . . .	3
2.2	OMNETPP . . . . .	6
<b>3</b>	<b>Análise</b>	<b>6</b>
<b>4</b>	<b>Conclusão</b>	<b>6</b>

# 1 Introdução

Em Redes WSN (Wireless Sensor Network) se requer uma duração prolongada das baterias de sensores IOT de forma que há um interesse crescente em entender o comportamento do consumo de energia dos mesmos.

De acordo com o estudo realizado por Transviña-Moreno et al [2], o consumo de energia em dispositivos WiFi se deve à máquina de estado finito (FSM Finite State Machine) que descreve os três estados de um nó sensor.

- Estado de Inicialização: Inicialização do hardware (aquecimento do oscilador e periféricos, inicialização de variáveis, etc.)
- Estado ativo. Tarefa de sensoriamento como por exemplo transmissão de dados. Enquanto em operação, o nó sensor habilita o conversor AD periférico e o desabilita na conclusão da mesma. No caso da difusão da mensagem, o módulo de rádio necessitará se juntar à rede, reportar os dados e fechar a conexão de energia para evitar o consumo desnecessário de energia.
- Estado de Baixa Potência. O módulo de rádio será desligado, o microcontrolador será mantido em modo de dormência e todos os periféricos serão desabilitados ou ligados à um estado de menor nível de consumo.

A figura abaixo ilustra o estado FSM:

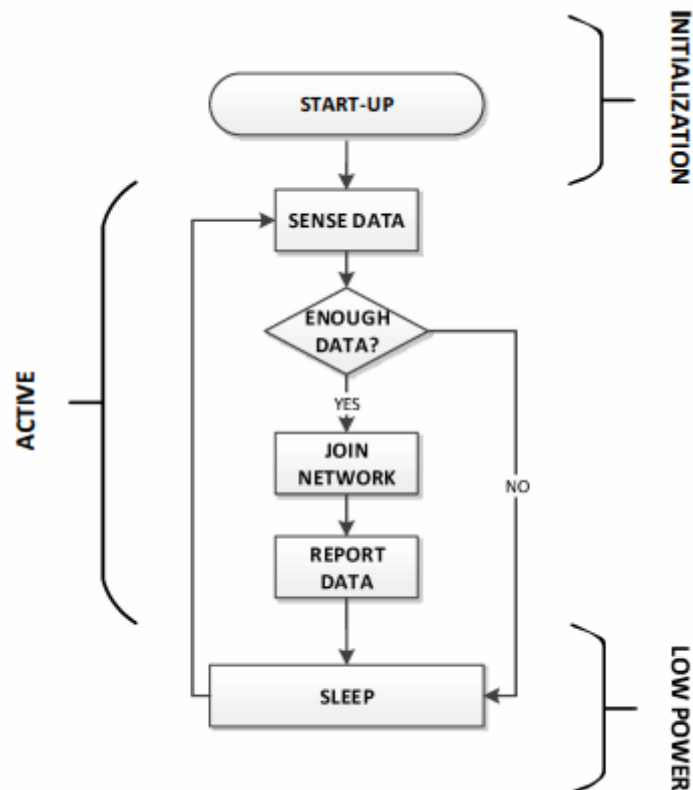


Figura 1: Máquina de Estado Finito (FSM)

Será também abordado o papel do harvester, um dispositivo de alimentação suplementar que é acoplada ao módulo de energia para prolongamento da operação dos sensores. De acordo com o [wikipedia.org](https://en.wikipedia.org/wiki/Energy-harvesting)<sup>1</sup>, a colheita (harvesting) de energia é o processo pelo a energia é derivada de fontes externas (e.g. energia solar, energia térmica, energia eólica, gradiente de salinidade e energia cinética, também conhecido como energia ambiental), capturada, e armazenada para pequenos dispositivos autônomos sem fio, como os dispositivos eletrônicos wearable e rede WSN.

O objetivo deste trabalho é verificar por meio de ambiente simulado o comportamento da energia em um nó sensor, eventualmente, verificar a aplicação do conceito FSM em rede WSN (especialmente a rede WiFi). Os seguintes ambientes simulados estão disponíveis e públicos: NS3<sup>2</sup>, OMNET++<sup>3</sup>, WSNET<sup>4</sup>, SENSE<sup>5</sup> e outros.

Será priorizado os simuladores NS3 e OMNET++, pelo fato de já terem sido apreciado em classe. A intenção deste relatório é reportar as saídas dos simuladores e sumará-los em gráficos e, se possível, reportar quaisquer limitações desses simuladores, se há algum esforço adicional para que os mesmos operem a contento.

Como metodologia, não será necessário observar a rede inteira, apenas o comportamento do consumo de energia de um nó específico em uma rede WSN, onde 2 nós (e.g. 2 nós WiFi) ou 3 nós (1 rede LAN com apenas 1 nó CSMA, 1 nó AP, e 1 nó WiFi) seria o suficiente.

Dessa forma conseguiríamos testar algumas variáveis no comportamento de energia em um nó WSN como por exemplo a distância entre os nós, o tempo de simulação, eventualmente o protocolo de comunicação.

Por limitação de tempo, não será escopo deste trabalho alterar o comportamento core do módulo de energia das bibliotecas padrões dos aplicativos.

Será explicado a metodologia utilizada, análise dos resultados e finalmente uma conclusão final.

## 2 Metodologia

Nesta seção apresentamos os principais conceitos e técnicas que serão abordadas para a realização deste Trabalho Prático.

Todo o material produzido, shell script, programa python e .cc estão armazenados no GitHub<sup>6</sup>

---

<sup>1</sup><https://en.wikipedia.org/wiki/Energy-harvesting>

<sup>2</sup><https://www.nsnam.org/>

<sup>3</sup><https://omnetpp.org/>

<sup>4</sup><http://wsnet.gforge.inria.fr/>

<sup>5</sup><https://www.ita.cs.rpi.edu/>

<sup>6</sup><https://github.com/edbkei/mo809a-trabalhopratICO/>

## 2.1 NS3

O módulo de energia do NS3 é baseado no Energy Model<sup>7</sup>, que é o resultado do modelo de energia proposto por Taparello et all [1], a figura abaixo ilustra o diagrama de energia.

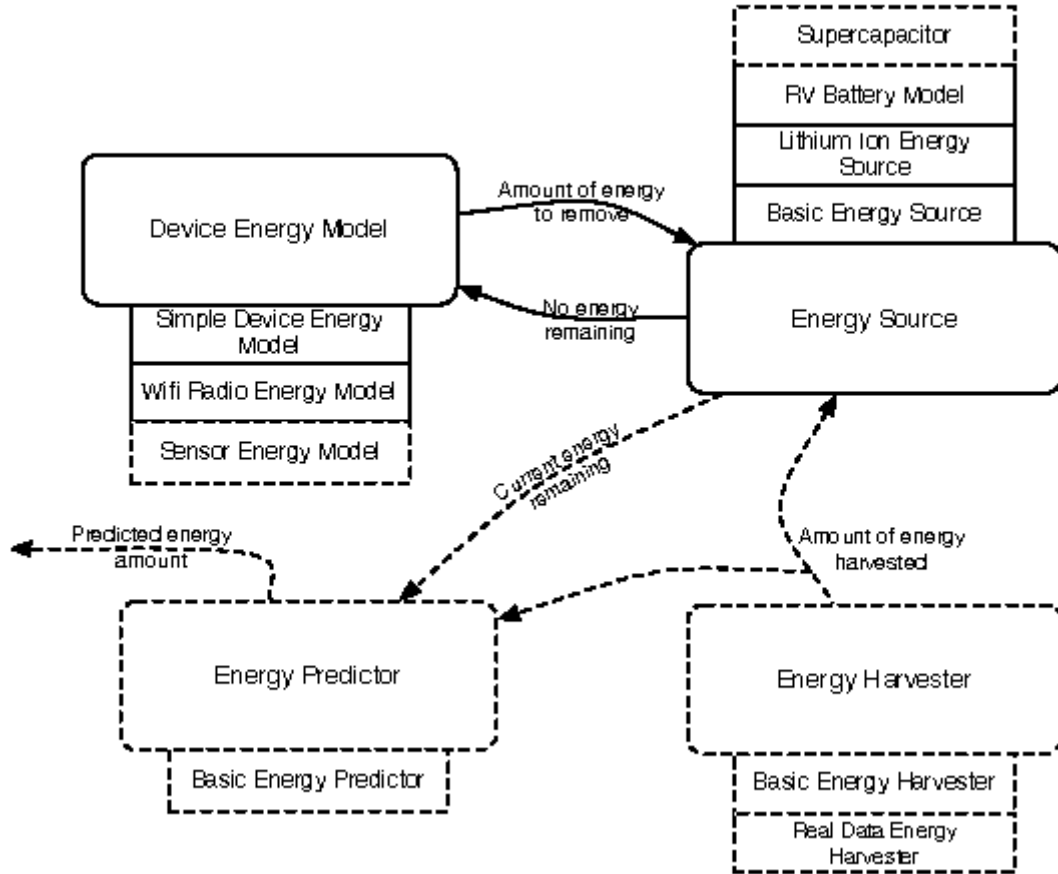


Figura 2: Modelo de Energia no NS3.

A instalação do NS3 foi realizada em uma máquina virtual Ubuntu 18.04.2 LTS e os programas do energy model estão instalados da seguinte maneira:

A lista de programas .cc e .h compõem a biblioteca do Energy Model, o exemplo energy-model-with-harvesting-example.cc será utilizado como base de um novo programa mo809a2019.cc para conectar o modelo de energia à um nó específico (no caso o nó de destino WiFi que recebe pacotes UDP de um nó de origem WiFi). Um módulo de alimentação externa (harvester) já vem conectado ao módulo de energia. Em nosso experimento, vamos mostrar o que acontece quando o módulo de alimentação é desconectado do modelo de energia. Introduziremos novos parâmetros externos na execução do comando ./waf.

<sup>7</sup><https://www.nsnam.org> -> Documents -> development-tree -> Doxygen -> Modules -> Energy Models

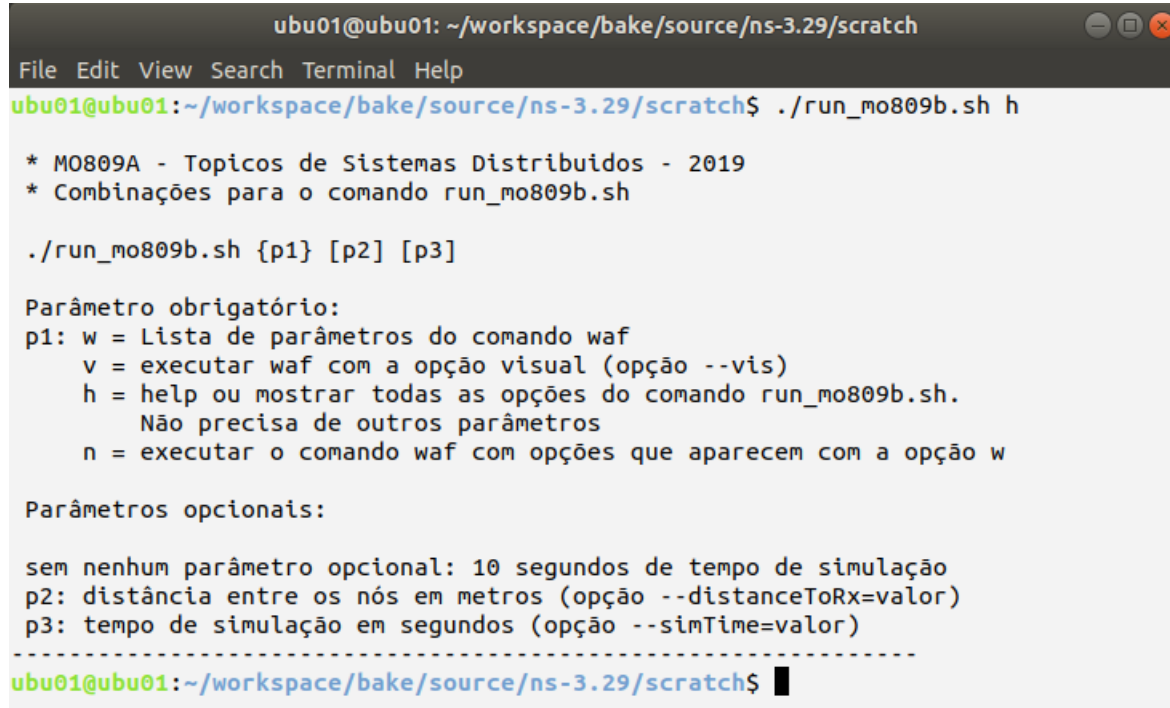
```
ubu01@ubu01: ~/workspace/bake/source/ns-3.29/src/energy/helper
File Edit View Search Terminal Help
bindings doc examples helper model test wscript
ubu01@ubu01:~/workspace/bake/source/ns-3.29/src/energy$ cd model
ubu01@ubu01:~/workspace/bake/source/ns-3.29/src/energy/model$ ls
basic-energy-harvester.cc          energy-harvester.h
basic-energy-harvester.h          energy-source.cc
basic-energy-source.cc            energy-source.h
basic-energy-source.h             li-ion-energy-source.cc
device-energy-model.cc            li-ion-energy-source.h
device-energy-model-container.cc  rv-battery-model.cc
device-energy-model-container.h   rv-battery-model.h
device-energy-model.h             simple-device-energy-model.cc
energy-harvester.cc              simple-device-energy-model.h
ubu01@ubu01:~/workspace/bake/source/ns-3.29/src/energy/model$ cd ../helper
ubu01@ubu01:~/workspace/bake/source/ns-3.29/src/energy/helper$ ls
basic-energy-harvester-helper.cc  energy-model-helper.cc
basic-energy-harvester-helper.h   energy-model-helper.h
basic-energy-source-helper.cc     energy-source-container.cc
basic-energy-source-helper.h      energy-source-container.h
energy-harvester-container.cc    li-ion-energy-source-helper.cc
energy-harvester-container.h     li-ion-energy-source-helper.h
energy-harvester-helper.cc       rv-battery-model-helper.cc
energy-harvester-helper.h        rv-battery-model-helper.h
ubu01@ubu01:~/workspace/bake/source/ns-3.29/src/energy/helper$

ubu01@ubu01: ~/workspace/bake/source/ns-3.29/examples/energy
File Edit View Search Terminal Help
ubu01@ubu01:~/workspace/bake/source/ns-3.29/examples/energy$ ls
energy-model-example.cc          examples-to-run.py
energy-model-with-harvesting-example.cc  wscript
ubu01@ubu01:~/workspace/bake/source/ns-3.29/examples/energy$
```

Figura 3: Programas do Energy Model

A saída do programa mo809a2019b.cc será utilizado pelo programa Python run-mo809b-python.py para a geração de gráficos.

O shell script run-mo809b.sh irá gerenciar esses dois programas. O mesmo irá buscar os valores a partir dos parâmetros para gerar parâmetros ./waf. A localização deste arquivo e a execução do mesmo está mostrado na figura abaixo:



```
ubu01@ubu01: ~/workspace/bake/source/ns-3.29/scratch
File Edit View Search Terminal Help
ubu01@ubu01:~/workspace/bake/source/ns-3.29/scratch$ ./run_mo809b.sh h

* M0809A - Topicos de Sistemas Distribuidos - 2019
* Combinações para o comando run_mo809b.sh

./run_mo809b.sh {p1} [p2] [p3]

Parâmetro obrigatório:
p1: w = Lista de parâmetros do comando waf
    v = executar waf com a opção visual (opção --vis)
    h = help ou mostrar todas as opções do comando run_mo809b.sh.
        Não precisa de outros parâmetros
    n = executar o comando waf com opções que aparecem com a opção w

Parâmetros opcionais:

sem nenhum parâmetro opcional: 10 segundos de tempo de simulação
p2: distância entre os nós em metros (opção --distanceToRx=valor)
p3: tempo de simulação em segundos (opção --simTime=valor)
-----
ubu01@ubu01:~/workspace/bake/source/ns-3.29/scratch$
```

Figura 4: Execução do Shell Script de Energia

Os objetos run-mo809b.sh, mo809a2019b.cc e run-mo809b-python.py estão armazenados no GitHub conforme a figura abaixo:

Quando clonar esses do GitHub, deixar esses arquivos armazenadas no subdiretório /ns-3.29/scratch.

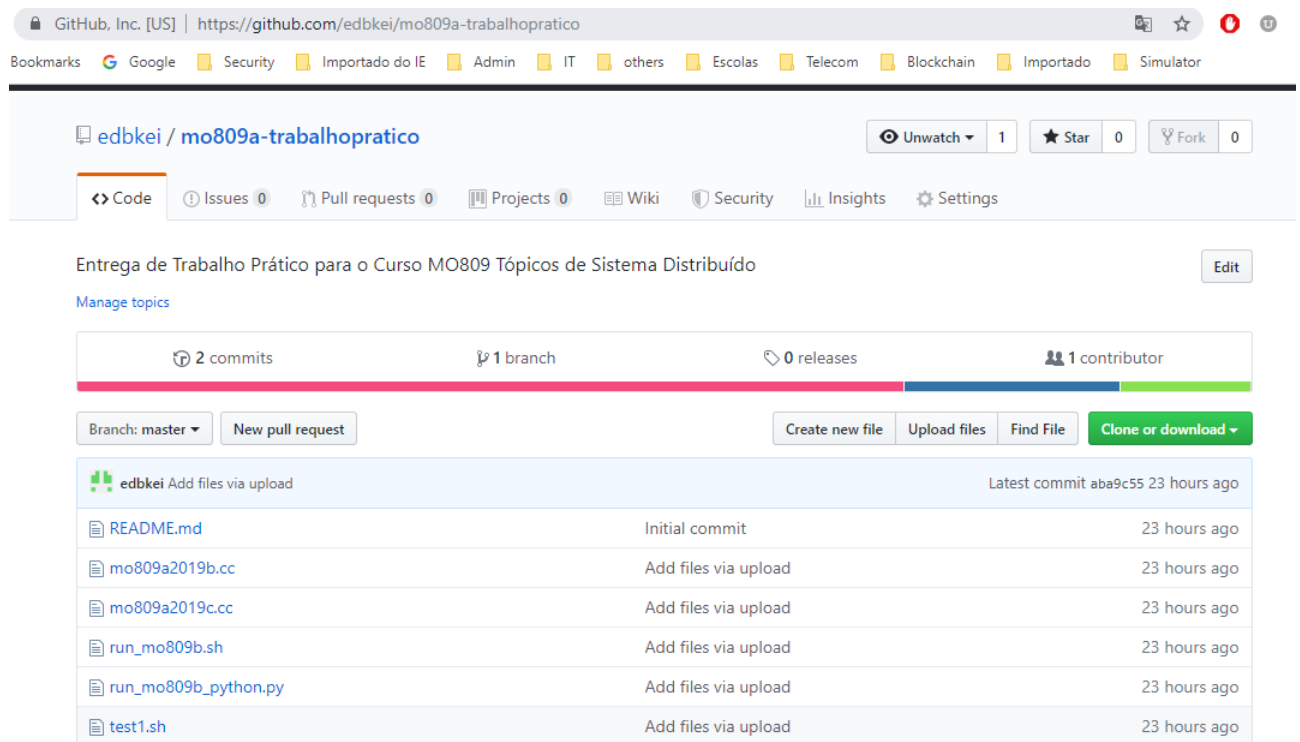


Figura 5: Estrutura do GitHub

## 2.2 OMNETPP

O módulo de energia do OMNETPP será baseado no inet-framework<sup>8</sup>

## 3 Análise

## 4 Conclusão

A conclusão que se quer atingir .....

<sup>8</sup><https://github.com/inet-framework/inet-showcases/tree/master/wireless/power>



## Referências

- [1] C. Tapparello, H. Ayatollahi, and W. Heinzelman. Energy harvesting framework for network simulator 3 (ns-3). 11 2014.
- [2] C. Trasviña-Moreno, A., n. Asensio, R. Casas, R. Blasco, and I. Marco. Wifi sensor networks: a study of energy consumption. *ResearchGate*, Feb. 2014.