Manual do Usuário

Última revisão

Autor	Data
Rafael Sampaio	Maio / 2021

Salvador, 2021

1. Sobre o IoTFogSim

IoTFogSim é um simulador orientado a eventos desenvolvido em python, que utiliza o paradigma da Simulação Paralela e Distribuída (PADS) para permitir que pesquisadores e desenvolvedores de sistemas para IoT, Fog computing e Cloud possa simular suas aplicações com grandes quantidades de nós e produção massiva de dados. O projeto iniciou em 2019 e faz parte de uma proposta de mestrado.

2. Sobre este documento

O presente documento traz informações sobre o instalação e o uso do simulador IoTFogSim. Deve ser utilizado como guia para o usuário.

3. Instalação

O IoTFogSim é multiplataforma, sendo validado no sistema operacional Windows e no Linux Ubuntu. A ferramenta requer a instalação do Python 3.5 ou superior.

Passos:

- *Clonar ou baixar a ferramenta no repositório oficial no github.*
- *Instalar Python 3.5 ou superior.*
- *Instalar o gerenciador de dependências 'pip' compatível com a versão do Pyhton utilizada.*
- *No terminal, acessar a pasta raiz do projeto.*
- *Instalar dependências utilizando o comando:*

```
pip install -r requirements.txt
```

4. Estrutura de pastas do simulador

Neste documento falaremos apenas das pastas interessantes aos usuários do simulador. Desenvolvedores que desejarem contribuir com a ferramenta devem

o consultar manual do desenvolvedor publicado a parte. A figura 1 apresenta a estrutura de pasta na qual está organizado o simulador.

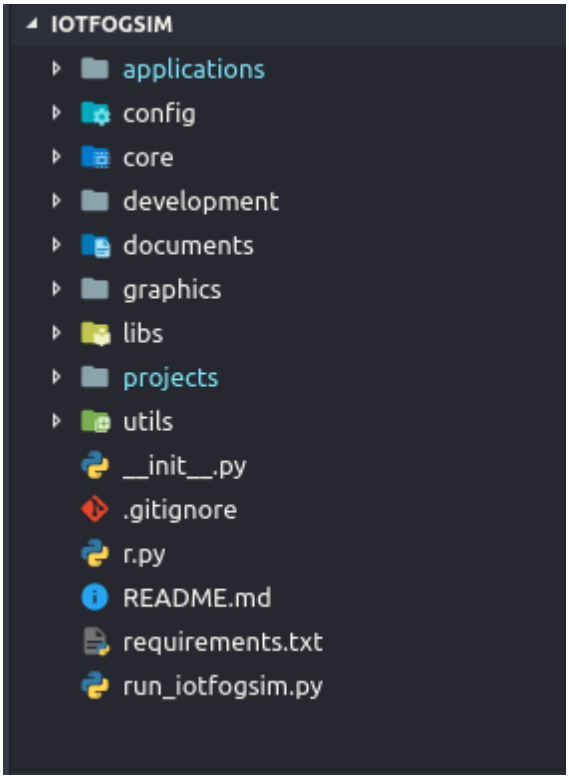

Figura 2. Estrutura de pastas do simulador.

applications – Pasta que contém os protocolos e algoritmos padrão do simulador. Nessa pasta também devem ser colocados os códigos criados e/ou personalizados pelos usuários.

config – Pasta que contém o arquivo *settings*, onde ficam configurações do simulador, por exemplo o caminho para a pasta de imagens e para a pasta de ícones. Recomenda-se que configurações desejadas pelo usuário estejam nessa pasta.

projects – Pasta que contém os projetos. Cada projeto representa uma simulação. Existem projetos de exemplo que acompanham o simulador. Ao criar uma subpasta em ‘projects’ o usuário inicia um novo projeto. O nome da subpasta será também o nome do projeto.

5. Inicialização do simulador

Para inicializar o simulador, no terminal, navegue até a pasta raiz do projeto e execute o comando:

```
python run_iotfogsim.py
```

Após executar o comando acima, você deverá ver a tela inicial do simulador, conforme a figura 2.

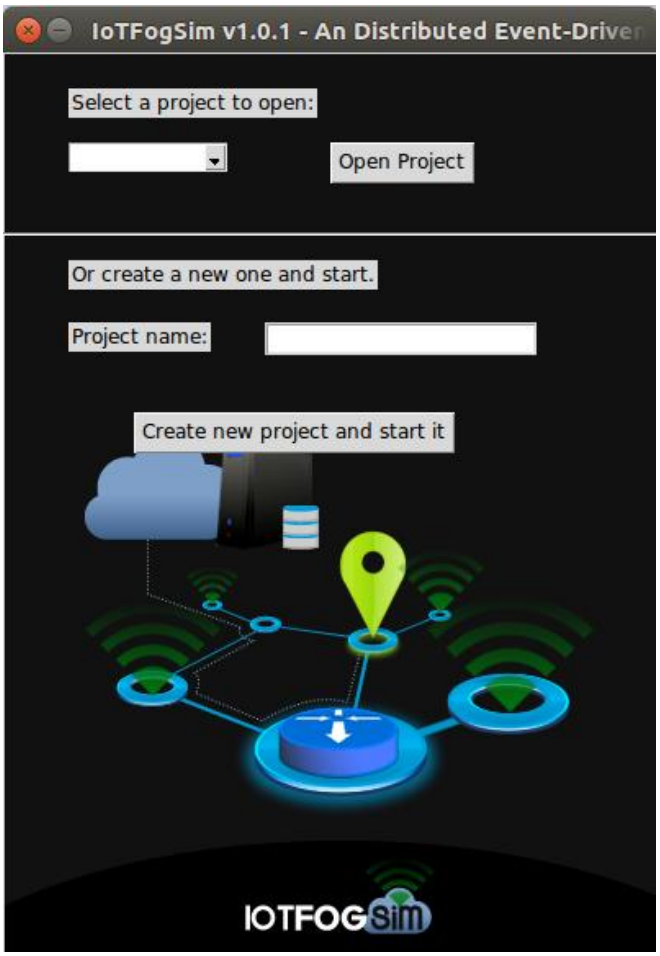

Figura 2. Tela inicial do simulador.

Nessa tela o usuário pode inicializar uma simulação já existente clicando no Botão ‘open project’ ou criar uma nova simulação clicando no botão ‘Create new project and start’. Caso opte pela segunda opção, a simulação irá iniciar vazia.

6. Primeira simulação

É recomendado que antes de criar suas próprias simulações, os usuários executem as simulações de exemplo que acompanham o simulador. Por ser a mais simples, a simulação do projeto ‘simple_wifi_http_client_server’ é a mais indicada para ser a primeira a se iniciar. A figura 3 apresenta a tela de simulação desse projeto.

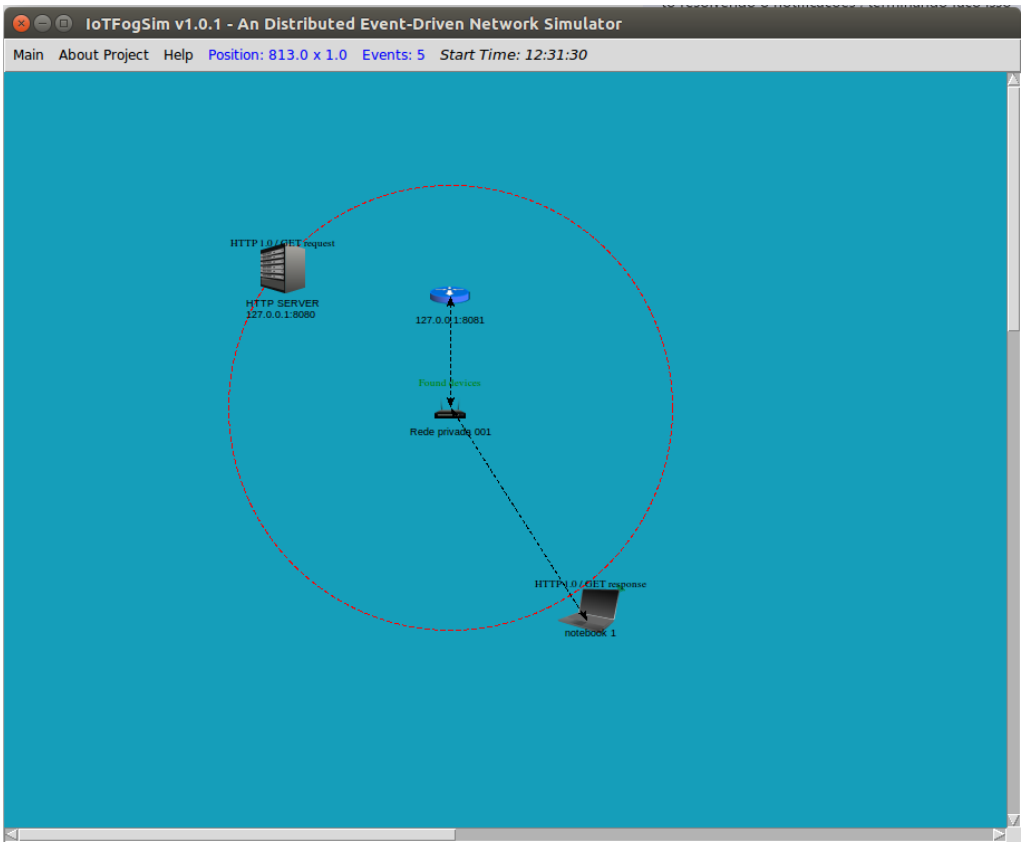

Figura 3. Tela de simulação do projeto simple_wifi_http_client_server.

7. Cirando uma nova simulação

Como dito na seção 4, cada pasta dentro da pasta ‘projects’ corresponde a um projeto, que equivale a uma simulação. O nome dado pasta será o mesmo nome do projeto.

7.1. O arquivo Nodes.js

Dentro de cada pasta/projeto deve haver um arquivo JSON chamado nodes.js obedecendo a seguinte estrutura:

```
{
        "fog": {
            "servers": [],
            "clients":[],
            "wireless_computers": [],
            "routers": [],
            "wireless_sensor_networks": []
          },

        "cloud": {
            "servers": [],
            "clients":[],
            "wireless_computers": [],
            "routers": [],
            "wireless_sensor_networks": []
          },

          "iot": {
            "servers": [],
            "clients":[],
            "wireless_computers": [],
            "routers": [],
            "wireless_sensor_networks": []
          }
}
```

Onde, serão descritos os dispositivos que estarão em cada uma das camadas (i.e. IoT,fog e cloud). Os dispositivos podem ser do tipo *server*, *client*, *wireless computer*, *router* e *wireless sensor network. Para esse exemplo, vamos criar apenas um cliente e um servidor, e executar em ambos aplicações HTTP que permita a comunicação entre os nós. O nosso arquivo nodes.js deve ficar da seguinte maneira:*

```
{
    "fog": {
        "servers": [
            {
                "id": 2,
                "name": "Web Server",
                "real_ip": "127.0.0.1",
                "simulation_ip": null,
                "icon": "broker_icon",
                "type": "server",
                "port": 8080,
```

```
                "is_wireless": false,
                "coverage_area_radius": 0,
                "application": "applications.httpapp.HttpServerA
pp",
                "x": 250,
                "y": 175
            }
        ],
        "clients":[
            {
                "id": 1,
                "name": "Web Client",
                "real_ip": "127.0.0.1",
                "simulation_ip": null,
                "icon": "cloud_icon",
                "type": "client",
                "is_wireless": false,
                "coverage_area_radius": 0,
                "application": "applications.httpapp.HttpClientA
pp",
                "x": 532,
                "y": 64
            }
        ],
        "wireless_computers": [
        ],
        "routers": [
            {
                "id": 3,
                "name": "Router 001",
                "real_ip": "127.0.0.1",
                "simulation_ip": null,
                "icon": "router_icon",
                "type": "router",
                "port": 8081,
                "coverage_area_radius": 0,
                "accesspoint_addr": "127.0.0.1",
                "accesspoint_port": 8082,
                "application": "applications.routerapp.RouterApp
",
                "is_wireless": false,
                "x": 400,
                "y": 200,
                "access_points": [

                ]
```

```
            }
        ],
        "wireless_sensor_networks": []
    },

    "cloud": {
        "servers": [],
        "clients":[],
        "wireless_computers": [],
        "routers": [],
        "wireless_sensor_networks": []
    },

    "iot": {
        "servers": [],
        "clients":[],
        "wireless_computers": [],
        "routers": [],
        "wireless_sensor_networks": []
    }
}
```

Ao estrutura nosso código dessa maneira, teremos uma simulação simples onde um cliente envia 'HTTP 1.0 / GET Request' e o servidor responde 'HTTP 1.0 / GET Response'. Execute o simulador e selecione o projeto que você criou na tela inicial. A simulação executará rapidamente pois nosso cliente e servidor enviam apenas uma mensagem cada. A figura 4 mostra a simulação que criamos.

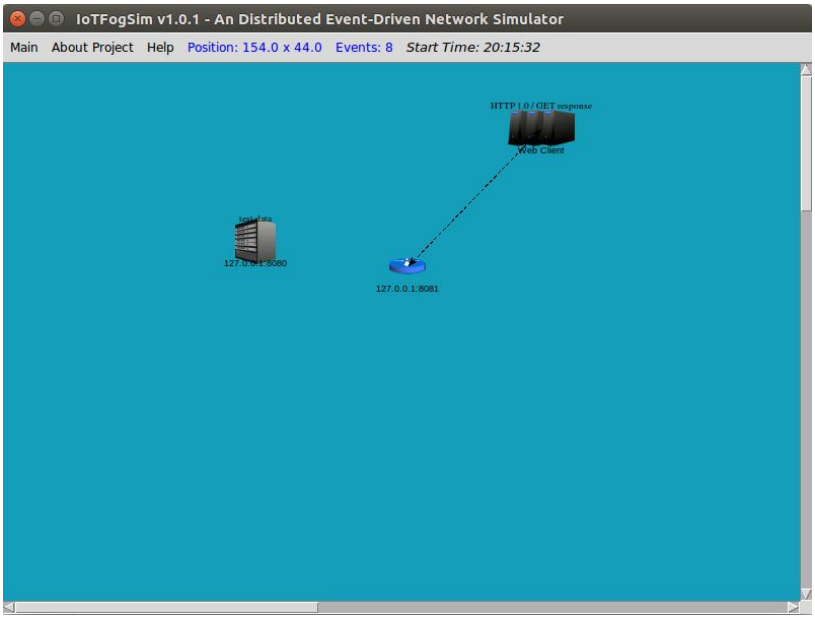

Figura 4. Simulação de Exemplo.