Internet das coisas

[1] **Antonio Nicassio**

[1]Departamento de Tecnologia – Universidade Estadual de Feira de Santana (UEFS)
44036–900 – Feira de Santana – Bahia

antonio.nicassio007@hotmail.com

Resumo. *A internet das coisas(Internet of things,IoT), vem revolucionando a integração entre dispositivos.Entretanto com tantos dispositivos diferentes se faz necessário um middleware distribuído para permitir tal comunicação.Este relatório descreve a implementação de um dispositivo virtual para dados fictícios, uma aplicação REST para interação com dispositivo, aliado ao serviço broker que funciona como o intermediador na comunicação entre dispositivo e aplicação.Os sistemas foram implementados utilizando recursos da linguagem python como sockets, frameworks para aplição bem como uma interface em javascrispt e html para acessar as rotas.O usuário através da interface pode ver informações e enviar comandos para os dispositivos conectados.*

1. Introdução

A internet das coisa(*Internet of Things*, IoT), tem alavancado a propagação de dados entre dispositivos diferentes de forma acelerada,permitindo novas funcionalidades a partir disto[Santos 2018].

Dados podem ser enviados entre dispositivos, entretanto com a diversidade de aparelhos, se faz necessário um intermediador para que se comuniquem adequadamente.Observando este fato, foi designado o desenvolvimento de um serviço intermediário de mensagens(*message broker*), para realizar permitir que haja comunicação entre dispositivos e aplicações [Magnoni 2015].

Foram implementados 3 produtos, um dispositivo virtual caracterizado como uma tranca de porta *smart*, um serviço *broker* que suporta protocolos de comunicação TCP, UDP [Alves 2008] e HTTP [LIMA and PETRICA], e uma interface que acessa as rotas HTTP do *broker* através de um navegador.Para construção dos produtos foram utilizados recursos nativos do python os *sockets* [Alves 2008], para realizar tanto a comunicação TCP como UDP.Na construção da API REST [Junior et al. 2021] fez se a utilização do *framework* python *Flask* para confecção das rotas e uma interface com html e javascript para acessá-las.

O usuário final poderá ver as informações das trancas conectadas, bem como atualizar alguns estados do dispositivo, através de comandos que são enviados pela interface ao *broker* e do broker para o dispositivo.

Este relatório sera organizado da seguinte maneira.A seção 2 apresentará os fundamentos teóricos para solução do problema. A Seção 3 para aspectos de metodologia e implementação.A seção 5 destinada a Resultados e Conclusões.

2. Fundamentação Teórica

Para construção dos produtos é necessário o conhecimento em alguns conceitos principais, para melhor organização este tópico será dividido em 2 subtópicos sendo eles Protocolos de comunicação e API.

2.1. Protocolos de Comunicação

Para fazer um sistema que faça a comunicação entre sistemas é necessário entender sobre protocolos de comunicação que são conjunto regras que fazem com que a informação enviada de um lugar ao outro possua padrões que garantam seu entendimento em ambas as partes [Brito et al. 2018].Tendo isso em mente é de suma importância entender alguns conceitos como sockets, TCP, UDP e HTTP.

2.1.1. Sockets

Os sockets funcionam como uma interface que pode ser utilizada para estabelecer uma comunicação entre dois pontos pela rede, de desta maneira se pode trocar informações entre dois processos através da rede que estão separados, podendo serem utilizados para implementar protocolos [Alves 2008].

2.1.2. TCP

O Protocolo *Transmission Control Protocol*(TCP), é um protocolo que preza a confiabiliade dos dados, com maneiras de garantir que um dado enviado chegará ao destino.O TCP oferece condições para o estabelecimento para o controle do fluxo de dados, impedindo assim uma inundações de dados por uma parte, além de caso algum dado ser pedido tem a possibilidade de identificar isso e retransmiti-lo.Isso pode ser feito devido ao fato do TCP estabelecer uma comunicação entre o Transmissor e o receptor [Alves 2008]

2.1.3. UDP

O Protocolo *User Datagram Protocol*(UDP), é um protocolo não confiável que preza pela velocidade e o desempenho, as camadas de confiabilidade empregadas fazem com que a comunicação fique mais lenta, por este motivo acaba por ser mais rápido.Desta Maneira um dispositivo pega um determinado dado e envia para o outro dispositivo sem nenhuma garantia que chegará lá ou não [Alves 2008].

2.1.4. HTTP

O *Hypertext Transfer Protocol*(HTTP) é um protocolo que atua na camada da internet, dividido em duas partes, o cliente responsável por querer algum dado no primeiro momento e o servidor, aquele que espera para prover algum dado ou serviço no primeiro momento.Para que essa comunicação ocorra utilizam protocolos HTTP que são compostos por uma requisição e uma resposta seguindo determinados padrões.

São diversos métodos para as requisições HTTP, como por exemplo, o método GET que solicita algo ao servidor e o método POST que envia algum arquivo, ambos esperam alguma resposta com valores que indicam sucessos e erros específicos [LIMA and PETRICA]

2.2. API

A *Application Programming Interface*(API), a API é um sistema que possuem um conjunto de regras que permitem que vários softwares utilizem de suas funcionalidades e serviços sem a necessidade que esses softwares saibam o seu funcionamento interno para que as utilizem, uma das Maneiras de se utilizar a API é através dos protocolos HTTP fazendo assim a comunicação entre dispositivos com o desacoplamento da linguagem utilizada desde que respeitem os padrões do HTTP.

A API ainda pode seguir uma arquitetura chamada *Representational State Transfer*(REST) que definem algumas restrições para que os recursos pudessem funcionar de forma independente sendo algumas principais[Junior et al. 2021]:

-A definição de URIS para acesso a recursos

-Definir ações sobre os recursos através dos Métodos HTTP

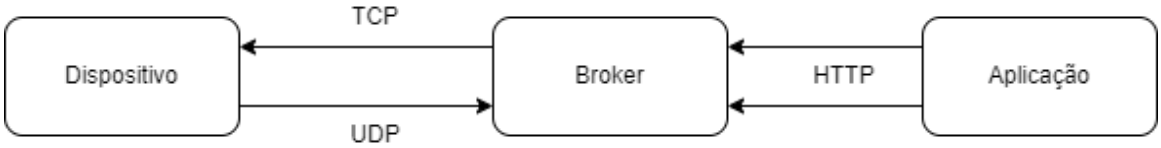
-Representação dos recursos definidos pelas partes

-Ter Códigos definidos para Resposta

3. Metodologia e Implementação

O projeto desenvolvido é composto por três produtos principais, sendo eles esquematizados na Figura 1.

Figura 1. Diagrama do Geral



Para entender melhor o funcionamento das comunicações e dos produtos desenvolvidos, está sessão será dividida em Produtos construídos e Protocolo de comunicação envolvido.

3.1. Produtos construídos

3.1.1. Dispositivo

Foi construído um dispositivo virtual para geração de dados fictícios.O dispositivo escolhido foi uma tranca de porta **smart**, que simula a berta e o fechamento da porta na qual ela está instalada, e a partir deste fato pode gerar alguns dados, sendo eles:

-O estado da porta: fechada ou aberta

-tempo total desde que ela foi aberta

-O estado da trava na porta: trancada ou destrancada

-O estado de funcionamento da tranca: ligada ou desligada

Sabendo desses dados é possível extrair algumas peculiaridades do dispositivo desenvolvido, como o fato da tranca não poder ser trancada se a porta estiver aberta, e caso a tranca esteja desligada, ela não gera dados e não recebe comandos.

O recebimento dos dados é feito via TCP e cada comando recebido via *broker*, e para cada comando recebido a Tranca envia algum comando de resposta via TCP a partir do comando recebido.

Assim que a tranca é ligada é iniciada a geração de dados na qual a porta é aberta e fechada de tempos em tempos desde que não esteja trancada,o tempo em que ela está aberta é contado a partir disso, e com os dados sendo gerados a cada 2 segundos são enviados os dados gerados pela Tranca ao Broker utilizando UDP

3.1.2. Broker

O *broker*, foi construído para realizar o intermédio da aplicação e o dispositivo, ele possui uma lista interna que contém as informações dos dispositivos conectados, sendo cada dispositivo sendo representado como um dicionário, que contém o socket do dispositivo conectado, seu ip, a porta TCP que usou para se conectar, bem como chaves que representam os dados gerados por ele que chegam via UDP são decodificados e utilizando o ip do dispositivo que enviou o dado como identificação, é percorrida a lista de dispositivos conectados e atualizada as informações do dispositivo correspondente.

Do lado da aplicação o *broker* oferece duas rotas HTTP,uma rota GET que permite que ele envie uma lista que contém informações dos dispositivos conectados e outra POST responsável por fazer que ele receba comandos da aplicação, ambas feitas utilizando o *Framework Flask*.

3.1.3. Aplicação

A aplicação foi feita uma interface, utilizando a linguagem de marcação *html*, e *javascript* para construção das funções que acessam determinada rota ao clicar em determinado botão sendo mostrados na Figura 2.

Existe um único botão em que acessa a rota GET do *broker* chamado ”Obter Lista de Trancas”,e ao aperta-lo a cada 2 segundos a aplicação faz um GET pegando as informações dos dispositivos conectados ao *broker*, e assim gerando um campo em que o usuário pode interagir com o dispositivo.

Tendo o campo para interação com o Dispositivo a aplicação possui botões que acessam uma mesma rota POST, para o *broker* com mudando apenas o conteúdo da mensagem que será enviado ao **broker**.Caso o dispositivo esteja desligado o único comando que efetivamente sai da aplicação para o broker é o botão que contém a rota POST com uma mensagem que pode liga-lo.

Figura 2. Interface da Aplicação

3.2. Protocolo de comunicação envolvido

Alguns protocolos de comunicação forma implementados para que ambas as partes possam manter a troca de informações

3.2.1. Dispositivo e Broker

O *broker* e o dispositivo podem manter a troca de informações tanto em TCP quanto quanto em UDP tendo algumas características no formato em que as informações são enviadas,

Em TCP, o *broker* é sempre o responsável por enviar a primeira mensagem enviada pelo *broker* é uma string que pode ter essas configurações:

Broker para Dispositivo:

- ”comando-comando recebido”:O comando recebido varia dentre as opções que aparecem na Figura 2, sendo eles ”ligar”, ”desligar”, ”trancar”e ”destrancar

- ”verificando”: é um comando enviado pelo *broker* para verificar se o dispositivo ainda está conectado ou não

Para esses comando enviados pelo *broker*, existem comandos de resposta:

Dispositivo para Broker:

-”comando recebido-comando enviado broker-estado atual da parte afetada pelo comando-Estado atual da porta”

-”online”:comando de resposta para o comando ”verificando”

No UDP apenas existe a relação de envio Dispositivo para Broker, seguindo dois formatos:

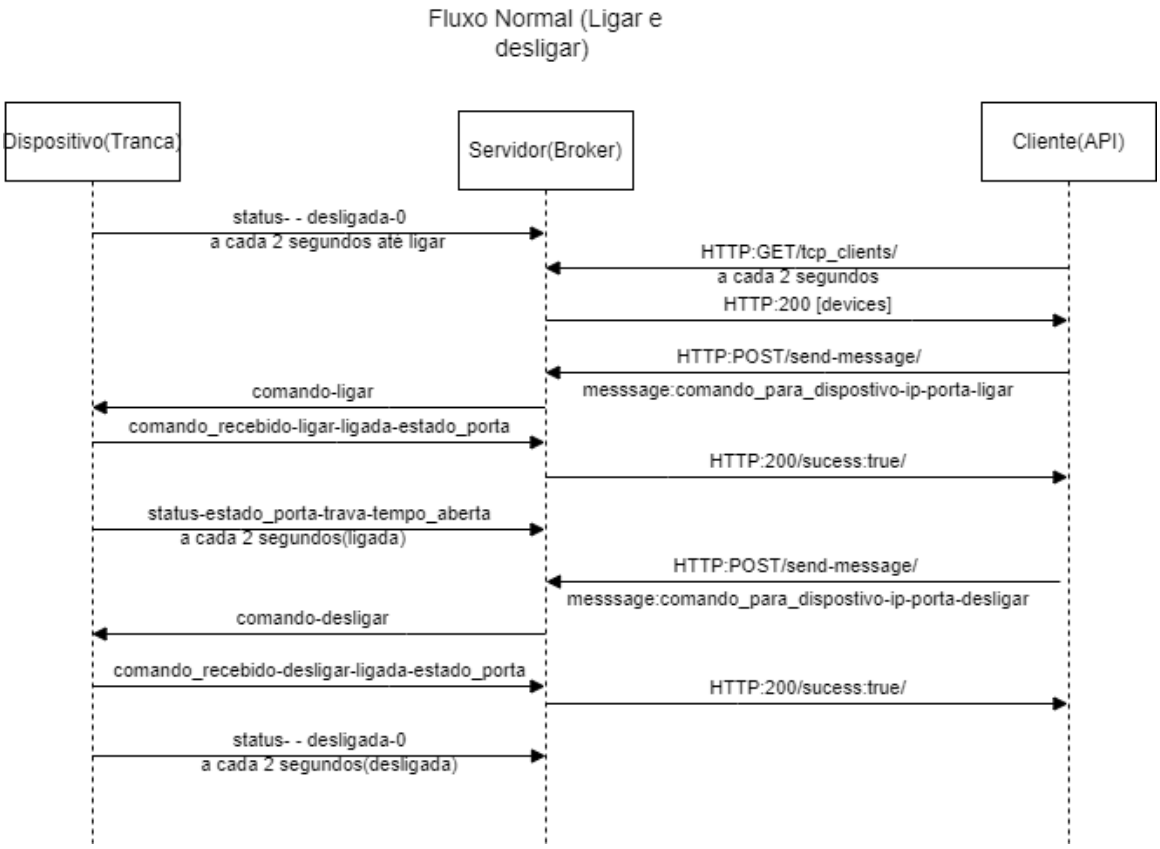
Dispositivo Desligado:

-”status- -desligada-0”:enviando dessa maneira o dispositivo entende a interface final entendera que o dispositivo está desligado

Dispositivo Ligado:

-”status-estado porta-estado Tranca-tempo aberta”

Figura 3. Diagrama comunicação comandos Ligar e Desligar



O fluxo das informações pode ser visto através da Figura 3

3.2.2. Broker e Aplicação

A comunicação entre *broker* e a aplicação é feita utilizando protocolos HTTP que segue no formato em que uma requisição é feita do da aplicação para o *broker* e uma resposta é enviada do *broker* para a aplicação.

Para o método GET:

-requisição:/tcp-clients/

-resposta: [{estado:string, ip: string, porta-tcp: int, tempo aberta: string, trava: string}, ...]

para o método POST:

-requisição: /send-message/

Json: {message: "comando para dispositivo-ip-porta$_tcp - comando$"}

-resposta: json: message:"Mensagem enviada com sucesso", sucess: "True

4. Resultados e Conclusões

Através dos produtos implementados, foi possível realizar um sistema *broker*, que permite escalar a conexão e gerenciamento de dispositivos, através de uma interface de uma aplicação, desde que tais dispositivos e interface atendam os protocolos de comunicação requeridos do *broker*.

A solução permitiu a comunicação entre interface e dispositivo de forma assíncrona, podendo receber comandos a qualquer momento sem a estarem disponíveis um para o outro a todo momento.Além disso os produtos contam com tratamentos de possíveis erros que podem acontecer como por exemplo encerramentos de processos envolvidos ou desconexão entre as partes, através de um sistema de verificação de funcionamento através de envio de mensagens periódicas, junto com limitação de tempo sem atividade.

O problema foi de grande importância para entender a base da conexão entre dispositivos IoT, bem como trocam informações, aliado ao conhecimento de uma confecção de um *middleware* distribuído, aliado aos fundamentos de uma API REST, junto com o conhecimento sobre protocolos de comunicação de baixo nível como TCP/IP.

Referências

Alves, M. M. (2008). *Sockets Linux*. Brasport.

Brito, L. L. F., Neto, M. M., Oliveira, M. R. F., Moraes, I. A., and Muniz, V. A. D. O. (2018). Protocolos de comunicação para internet of things (iot). *Intercursos Revista Científica*, 17(1).

Junior, E. A. G., Rocha, R. D., and de Souza Maciel, R. (2021). Desenvolvimento de api rest com spring boot.

LIMA, L. and PETRICA, E. Protocolo http.

Magnoni, L. (2015). Modern messaging for distributed sytems. In *Journal of Physics: Conference Series*, volume 608, page 012038. IOP Publishing.

Santos, S. (2018). *Introdução à IoT: desvendando a internet das coisas*. SS Trader Editor.