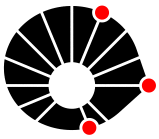

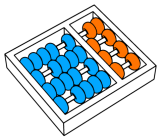

Tolerância a falhas em sistemas de Internet das Coisas - Estudo de caso em um estacionamento inteligente.

Gabriel Massuyoshi Sato, Juliana Freitag Borin

Relatório Técnico - IC-PFG-22-47
Projeto Final de Graduação
2022 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Tolerância a falhas em sistemas de Internet das Coisas - Estudo de caso em um estacionamento inteligente.

Gabriel Massuyoshi Sato, Juliana Freitag Borin*

Novembro 2022

Resumo

A evolução das tecnologias, principalmente dos dispositivos dentro do que chamamos de Internet das Coisas (IoT), possibilitou o surgimento de sistemas cada vez mais sofisticados, muitas vezes com requisitos críticos de funcionamento. O estudo sobre tolerância a falhas (FT) vem ganhando um espaço extremamente importante no cenário de IoT, pois as existentes técnicas que fazem um sistema ser tolerante a falha muitas vezes não são eficientes ou viáveis para um sistema de dispositivos de Internet das Coisas. Este estudo visa analisar algumas técnicas de FT pensadas para o IoT. Além disso, mostraremos um estudo de caso onde temos uma solução para gerir um estacionamento a partir de câmeras que tiram fotos periodicamente, obtendo-se a quantidade de vagas disponíveis, e apresentaremos uma solução para o sistema ser tolerante a falhas.

1 Introdução

O avanço tecnológico, principalmente das tecnologias de comunicação sem fio (*wireless*), possibilitou que diversos dispositivos (sensores, etiquetas RFID, celulares) se comuniquem entre si de forma eficiente e rápida. A Internet das Coisas apoiada por tecnologias habilitadoras como redes de sensores, protocolos de comunicação, computação pervasiva, entre outras, transforma objetos em dispositivos inteligentes capazes de coletar dados e interagir entre si e com outros sistemas [7].

O estudo sobre tolerância à falha é fundamental para tornar um sistema confiável e na área de Internet das Coisas isso não é diferente. Detectar falhas e aplicar técnicas

*Instituto de Computação, Universidade de Campinas, SP, Brasil.

para recuperar o sistema é uma tarefa difícil no cenário IoT, onde temos aplicações heterogêneas, distribuídas em muitas camadas, muitas vezes dependentes de comunicação sem fio e com problemas de escalabilidade [8]. Adicionalmente, temos também o fato da dinamicidade dos sistemas, pois é comum que dispositivos e funções sejam frequentemente adicionados e removidos para melhoria e atualização da aplicação [5].

Muitos sistemas IoT possuem requisitos críticos de funcionamento, ou tem funcionalidades para as quais uma falha pode gerar consequências graves (como em aplicações da área da saúde) e um estudo de como detectar e lidar com as falhas e erros é fundamental.

As atuais soluções de tolerância a falhas em IoT muitas vezes são inadequadas ou insuficientes, visto que a maioria é desenvolvida sob medida para uma aplicação específica, de modo que uma solução não pode ser reproduzida para diferentes sistemas e uma mudança na estrutura da rede que conecta os dispositivos ou troca de hardware pode tornar o suporte para tolerância a falhas inutilizado.

Dessa forma, o objetivo deste trabalho é analisar algumas técnicas de detecção e recuperação de falhas nos sistemas IoT, mostrando seu nível de escalabilidade de um sistema para outro, sua eficácia e os custos com recursos. Vamos comparar diferentes trabalhos, analisando os pontos citados anteriormente; depois mostraremos um estudo de caso onde temos uma solução para gerir um estacionamento a partir de câmeras que tiram fotos periodicamente, obtendo-se a quantidade de vagas disponíveis.

O restante deste documento está organizado da seguinte maneira: A Seção 2 apresenta algumas técnicas para deixar o sistema tolerante a falhas, assim como alguns trabalhos em que tais técnicas foram aplicadas. A Seção 3.2 mostra uma implementação de um estacionamento inteligente e uma solução parar tornar esse sistema tolerante a falhas. Por fim, a Seção 4 e apresentado uma conclusão do conclui o trabalho.

2 Tolerância a falhas na Internet das Coisas

Existem diversos trabalhos já publicados sobre tolerância a falhas em sistemas IoT. Porém, devido à heterogeneidade de dispositivos e aplicações, cada estudo mostra uma solução para um sistema específico. Vamos apresentar algumas técnicas que tornam sistemas IoT tolerantes a falhas, em seguida mostraremos alguns trabalhos em que tais técnicas foram aplicadas, sejam elas para detectar, recuperar ou amenizar uma falha do sistema.

2.1 Técnicas adotadas para tolerância a falhas em IoT

A seguir, mostraremos algumas das principais técnicas apresentadas por Moghaddam e Muccini [8] para construir um sistema resiliente.

2.1.1 Replicação

Muito utilizada no cenário de Internet das Coisas, essa técnica consiste em replicar os componentes, tanto de software, quanto hardware, dentro do sistema e compartilhar as mesmas informações entre eles. A replicação pode se dar de duas formas:

- **Replicação ativa -** todos os processos e hardwares replicados estão em funcionamento concomitantemente aos principais. Assim que ocorre uma falha, o componente ou processo replicado toma o lugar do principal e o funcionamento ocorre normalmente. Esse método consome muito processamento, mas proporciona a recuperação de forma extremamente rápida, além do tempo de falha ser basicamente determinístico.
- **Replicação passiva -** as partes replicadas permanecem paradas até que uma falha ocorra. Assim que confirmada a falha, o processo secundário recolhe as informações do primeiro e se torna o processo principal para o funcionamento do sistema. Apesar de utilizar menos recursos, oferece uma resposta mais lenta de recuperação.

Essa técnica, apesar de ser extremamente útil para a confiabilidade do sistema, demanda um custo muito grande, já que os hardwares e processos são replicados, consequentemente afetando também a escalabilidade em sistemas com muitos dispositivos ou com restrições energéticas.

2.1.2 Controle de rede

A técnica de controle de rede é utilizada para a detecção de falhas nos dispositivos IoT. É escolhido um *Cluster Head* (CH) na rede e esse requisita periodicamente informações sobre a saúde dos nós. Quando um dispositivo responde à requisição, significa que ele está "vivo" e o sistema segue funcionando normalmente. Caso não haja resposta, dependendo do nível de QoS (*Quality of Service*), podemos enviar mais uma requisição ou entrar em modo de recuperação daquele nó. Essa técnica é muito útil para identificar, por exemplo, problemas com conexão de rede e informações de telemetria dos dispositivos. Não consome muito processamento, porém sofre com o que chamamos de ponto singular de falha (significando que caso o CH falhe, o sistema de FT também falha).

2.1.3 Processamento de eventos complexos (CEP)

Processamento de evento complexo é utilizado para pesquisas e indústrias no intuito de identificar situações complexas, ou seja, definir regras para classificar uma composição de eventos [5]. Com um conjunto de entradas, é possível identificar um padrão no sistema e classificar quando se trata de um comportamento correto ou errôneo, sendo assim utilizado para monitorar o sistema de forma reativa.

2.1.4 Aprendizado de máquina (ML)

O aprendizado de máquina é amplamente citado na literatura para arquiteturas IoT para realizar previsões a partir das quantidades grandes de dados produzidos pelos dispositivos. Com essa técnica, é possível identificar e tomar decisões antecipadas sobre o comportamento errôneo do sistema, dando suporte de forma proativa para a aplicação [5].

2.2 Trabalhos com implementação das técnicas de tolerância a falhas

Esta seção apresenta dois trabalhos, disponíveis na literatura científica, que fazem uso das técnicas de tolerância a falhas descritas na seção anterior.

No estudo de Gia *et al* [1], uma arquitetura de Internet das Coisas para assistência médica é apresentada. O sistema é composto por uma rede 6LoWPAN em topologia estrela onde todos os nós se comunicam uns com os outros. Há um gateway que recolhe os dados dos dispositivos finais através de nós coletores e os transmite para um servidor remoto para processamento e consumo pelos clientes web. A proposta deste trabalho é realizar um controle de rede, onde se observa o fluxo de dados vindos dos nós finais. Quando há uma pausa nesse fluxo, medidas são tomadas no gateway para iniciar um protocolo de descoberta da razão da inatividade desse nó. O primeiro passo tomado é o envio, pelo nó coletor do gateway, de uma mensagem perguntando o *status* ao nó inativo. Se em um período de tempo, escolhido de acordo com a QoS da aplicação, há uma resposta, significa que o dispositivo final está funcionando corretamente. Caso contrário, é enviada uma mensagem de alerta por outro nó coletor para todos os nós ao seu alcance. A escolha de utilizar um nó coletor diferente do primeiro segue a linha da técnica de replicação para poder eliminar uma possível falha do nó coletor inicial. Todos os nós no alcance do coletor verificam se a mensagem é para eles ou não, a partir do seu identificador (ID). Se mesmo assim o dispositivo final não responde, é confirmada a falha.

Power e Kotonya [4] propõem uma estrutura IoT baseada em um micro serviço que oferece suporte em tempo real e preditivo para o sistema ser tolerante a falhas. A implementação responsável por tornar o sistema tolerante a falhas de forma reativa utiliza processamento de eventos complexos. Eles propõem uma série de regras que são aplicadas entre a borda e o banco de dados funcionando com uma espécie de *firewall*. Todos os dados passam por esse mecanismo e somente o conjunto aceitável passa adiante, funcionando como um filtro. Um desafio para essa técnica é classificar quais dados são aceitáveis e corretos e quais não são, dependendo muito do contexto da aplicação. A solução preditiva apresentada pelo autor recebe os dados e passa tanto pelo treinamento, quanto pelo classificador. Com isso, com cada informação recebida, o sistema cria um modelo mais atualizado e consegue prever padrões de

falhas, através do estado da aplicação, além de analisar a efetividade das estratégias de recuperação.

3 Estacionamento inteligente

Com o crescimento da quantidade de carros nas cidades, fica visível a necessidade de um gerenciamento melhor dos estacionamentos em áreas de uso comum. A procura por vagas em estacionamentos aumenta a circulação de carros nas vias, aumentando o congestionamento e a quantidade de carbono emitido pelos veículos [6]. Diante disso, a motivação de tornar esse processo mais eficiente é grande, principalmente pelos conceitos de cidades inteligentes, onde se busca melhor aproveitamento de recursos e energia e aumento na eficiência dos serviços [3].

Em média, 31% do solo das grandes cidades é utilizado por carros estacionados, sendo que em algumas cidades, como Los Angeles, a taxa chega a 81% [2]. Além disso, de acordo com a ONU, com o grande crescimento populacional nas cidades, estima-se que em 2050 cerca de 6 bilhões de pessoas estarão morando em áreas urbanas [3]. Com esses altos números, esse estudo, juntamente com todos os outros relacionados às cidades inteligentes, são de suma importância para o futuro da eficiência dos espaços urbanos.

O estudo de caso a ser demonstrado a seguir é a implementação de um estacionamento inteligente utilizando uma câmera que periodicamente tira fotos do estacionamento localizado no Instituto de Computação (IC) da Unicamp. As imagens são analisadas por uma rede neural resultando numa inferência com o número de vagas livres no bolsão de estacionamento. Após a obtenção, os resultados são enviados para a nuvem e mostrados em um painel de LED posicionado em frente ao IC.

Dessa forma, mostraremos os pontos de falhas que o sistema possui e uma implementação de técnicas para o sistema se tornar tolerante a falhas.

3.1 Tecnologias Utilizadas

A escolha das tecnologias utilizadas no estudo de caso serão explicadas a seguir.

- **Protocolo de comunicação Wi-Fi -** foi escolhido o Wi-Fi como protocolo de comunicação pela facilidade de utilização do padrão e pelo fato da aplicação depender da conexão com a Onternet. Utilizando uma tecnologia de comunicação sem fio, não dependemos de fios (conexão Ethernet), torna o sistema apto para ser colocado em qualquer área que o Wi-Fi alcance. Os contras de utilizar uma conexão sem fio, como a latência elevada, confiabilidade e gasto energético não são pontos relevantes para a escolha de outra alternativa, já que, respectivamente, a aplicação não possui requisitos críticos de latência, os dados não são

sensíveis a ponto de uma perda de pacote ser crucial para o bom funcionamento do sistema e todos os hardwares estão conectados à rede elétrica;

- **Raspberry Pi com câmera acoplada -** a escolha de utilizar uma Raspberry Pi foi pelo seu poder computacional para poder tirar fotos periodicamente do estacionamento e ao mesmo tempo conseguir realizar a análise das fotos. Além disso, como ela possui o módulo de Wi-Fi, é facilmente conectada à Internet para poder enviar os dados à nuvem;

- **Plataforma Konker Labs -** A Konker Labs[1] é uma plataforma em nuvem de gerenciamento para dispositivos de Internet das Coisas de fácil manuseio, ideal para o propósito do projeto. Nela são criados os dispositivos virtuais que se relacionam diretamente com seus pares físicos, obtendo uma melhor visualização dos dados que cada dispositivo recebe ou envia;

- **ESP8266 NodeMCU -** é um microcontrolador de baixo custo com uma capacidade de processamento razoável e capacidade de comunicação por Wi-Fi para receber a quantidade de vagas de estacionamento livres vindas da nuvem. A escolha desse dispositivo se dá justamente pela comunicação com a Internet e processamento suficiente para disparar os circuitos do painel de LED;

3.2 Implementação da solução

Podemos dividir o funcionamento do sistema em 3 etapas, sendo **coleta e cálculo** dos resultados, **armazenamento** dos resultados na nuvem e **apresentação** da quantidade de vagas disponíveis no estacionamento.

3.2.1 Coleta e análise das imagens

Toda a coleta dos dados e análise de imagens é feita na Raspberry Pi. Através da câmera acoplada, fotos do estacionamento são tiradas periodicamente pelo processo *Gateway_pic.py* e são enviados localmente (rede local na própria Raspberry Pi).

Um segundo processo *Gateway_neuralnet.py*, que também roda independente, verifica na rede local se existe alguma imagem nova e caso tenha, passa por uma rede neural pré treinada e realiza uma inferência. O modelo utilizado foi treinado com várias fotos tiradas do estacionamento em diversas condições e climas, seja elas com vários carros, nenhum carro, de dia, de noite, chuva, etc.

Após realizada a inferência, os dados são enviados para a nuvem pelo processo *Gateway_main.py* via MQTT. O formato padrão para comunicação MQTT é em JSON. Um exemplo de dado enviado para a Konker Labs é:

[1] https://iot.konker.me

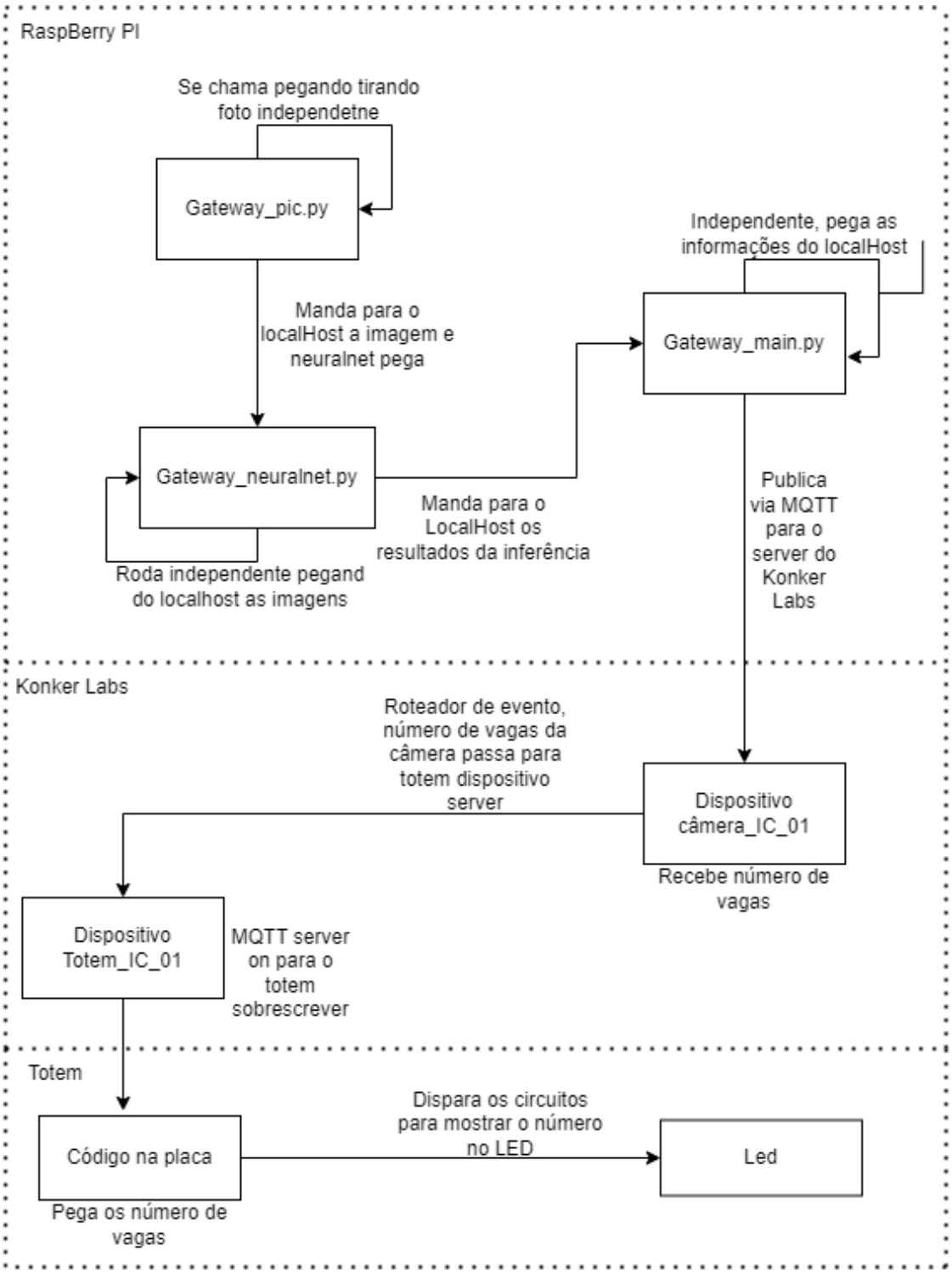

Figura 1: Fluxograma do sistema de estacionamento inteligente.

```
{
    "_ts":1669225448875,
    "veiculos":5,
    "vagas":11
}
```

onde "_ts" é o timestamp em Epoch Unix, "veiculos" é a quantidade de veículos estacionados e "vagas" é o número de vagas livres disponíveis no estacionamento.

Além dos resultados vindos da inferência, dados de saúde do dispositivo, como ping e temperatura interna do circuito, são enviados à nuvem para monitoramento e possível implementação de técnicas de tolerância falha.

Computação em borda

A escolha da utilização de uma Raspberry Pi, como dito na Seção 3.1, foi, principalmente, pela sua capacidade de processamento local. Realizando toda a coleta e análise dos dados localmente podemos desfrutar das principais vantagens [9] [11] que existem atualmente na computação em borda:

- **Utilização de banda -** realizando o processamento das imagens na borda, não necessitamos enviar dados brutos, que no caso possuem tamanho relativamente grande, já que se trata de imagens. Dessa forma, economizamos na utilização de banda, além de diminuir a latência da aplicação e diminuir o tempo em que o sistema está suscetível a falhas de rede;

- **Armazenamento -** as imagens são utilizadas somente para a realização das inferências, tornando-se inúteis após sua análise. Armazená-las na nuvem não é necessário. Assim, realizando o processamento na borda, podemos descartar as fotos logo após passar pela rede neural;

- **Privacidade dos dados -** as fotos tiradas pela câmera contêm informações com certo grau de confidencialidade, como a placa dos carros e até mesmo o rosto das pessoas que estão circulando pelo estacionamento, não sendo interessante o seu armazenamento. Realizando todo o processo localmente evitamos ataques durante a transmissão de dados.

3.2.2 Armazenamento dos dados na nuvem

Todo o armazenamento dos resultados vindos da Raspberry Pi é feito na Konker Labs. Nela, todos os dispositivos reais possuem seus respectivos virtuais que recebem ou enviam os dados. No caso, 2 dispositivos são criados na plataforma, sendo eles

o *camera_IC_01*, que corresponde à Raspberry Pi e *totem_IC_01* que corresponde ao totem onde é visualizada a quantidade de vagas disponíveis do estacionamento. Os dados recebidos via MQTT da Raspberry Pi chegam na plataforma e são armazenados, como mostra a Figura 2.

Eventos de Entrada | Eventos de Saída

Data/Hora	Canal	Mensagem
23/11/2022 16:54:09.650 BRT	data	{"_ts":1669233249650,"veiculos":6,"vagas":10}
23/11/2022 16:44:10.521 BRT	data	{"_ts":1669232650521,"veiculos":5,"vagas":11}
23/11/2022 16:34:11.394 BRT	data	{"_ts":1669232051394,"veiculos":9,"vagas":7}
23/11/2022 16:24:09.221 BRT	data	{"_ts":1669231449221,"veiculos":7,"vagas":9}
23/11/2022 16:14:10.067 BRT	data	{"_ts":1669230850067,"veiculos":7,"vagas":9}
23/11/2022 16:04:10.872 BRT	data	{"_ts":1669230250872,"veiculos":6,"vagas":10}
23/11/2022 15:54:11.744 BRT	data	{"_ts":1669229651744,"veiculos":6,"vagas":10}
23/11/2022 15:44:09.641 BRT	data	{"_ts":1669229049641,"veiculos":7,"vagas":9}
23/11/2022 15:34:10.512 BRT	data	{"_ts":1669228450512,"veiculos":9,"vagas":7}
23/11/2022 15:24:11.318 BRT	data	{"_ts":1669227851318,"veiculos":4,"vagas":12}

Figura 2: Dados recebidos da Raspberry Pi na plataforma Konker Labs.

Depois de recebidos os resultados na nuvem, criamos um **Roteamento de Evento**. Essa funcionalidade é utilizada para que quando o dispositivo *camera_IC_01* receba uma mensagem MQTT, essa mensagem seja enviada para o dispositivo *totem_IC_01* como mostra a Figura 3. Por fim os resultados são enviados da plataforma para o ESP8266 Node MCU via MQTT como mostra a Figura 4.

3.2.3 Apresentação do número de vagas

Após o recebimento dos dados da nuvem, o ESP8266 Node MCU ativa um painel de LED mostrando o número de vagas total do estacionamento. A utilização desse tipo de painel de 7 segmentos da uma boa visualização de longe para motorista que está chegando ao local.

3.3 Tolerância a falhas

De acordo com o trabalho de Moghaddam e Muccini [8], temos diversas arquiteturas de sistemas de Internet das Coisas. No caso da aplicação de estacionamento

Figura 3: Roteamento dos dados recebidos do dispositivo *camera_ic_01* para o dispositivo *totem_ic_01* na plataforma Konker Labs.

Eventos de Entrada | Eventos de Saída

Data/Hora	Canal	Mensagem
23/11/2022 16:54:09.650 BRT	data	{"_ts":1669233249650,"veiculos":6,"vagas":10}
23/11/2022 16:44:10.521 BRT	data	{"_ts":1669232650521,"veiculos":5,"vagas":11}
23/11/2022 16:34:11.394 BRT	data	{"_ts":1669232051394,"veiculos":9,"vagas":7}
23/11/2022 16:24:09.221 BRT	data	{"_ts":1669231449221,"veiculos":7,"vagas":9}
23/11/2022 16:14:10.067 BRT	data	{"_ts":1669230850067,"veiculos":7,"vagas":9}
23/11/2022 16:04:10.872 BRT	data	{"_ts":1669230250872,"veiculos":6,"vagas":10}
23/11/2022 15:54:11.744 BRT	data	{"_ts":1669229651744,"veiculos":6,"vagas":10}
23/11/2022 15:44:09.641 BRT	data	{"_ts":1669229049641,"veiculos":7,"vagas":9}
23/11/2022 15:34:10.512 BRT	data	{"_ts":1669228450512,"veiculos":9,"vagas":7}
23/11/2022 15:24:11.318 BRT	data	{"_ts":1669227851318,"veiculos":4,"vagas":12}

Figura 4: Dados enviados da plataforma Konker Labs para o ESP8266 Node MCU.

Figura 5: Totem mostrando o número de vagas livres do estacionamento.

inteligente, trata-se de uma arquitetura distribuída, onde há praticamente todo o processamento dos dados na borda. Esse tipo de arquitetura, de acordo com os estudos feitos pelos autores [8], são extremamente propícios a falhas no sistema, sendo necessário um estudo bastante aprofundado sobre possíveis pontos de falhas e técnicas para detectá-los e corrigi-los. A Figura 6 mostra o esquema do estacionamento inteligente para melhor visualização das comunicações entre os processos.

3.3.1 Pontos de falhas

Para mapear os momentos em que o sistema pode vir a falhar, utilizaremos uma técnica de análise de falhas como forma de facilitar e cobrir o sistema todo.

Análise de árvore de falhas (FTA)

A técnica da análise de árvore de falhas surgiu da Bell Telephone Laboratories em 1962 para análise do sistema de controle do lançamento do míssil intercontinental Minuteman [10]. Se trata de uma técnica analítica onde é definida uma falha e, a partir desse evento de topo, procuramos todas as combinações para chegar nele, sendo elas primárias ou não [10]. Uma árvore de falhas nos dá uma representação gráfica da relação lógica entre os eventos falhos que podem acontecer no sistema [10]. A Figura 7 mostra a FTA feita do sistema de estacionamento inteligente.

Os retângulos representam um evento não primário, ou seja, uma falha que pode ser destrinchada em outras falhas. Os círculos representam os eventos primários, onde

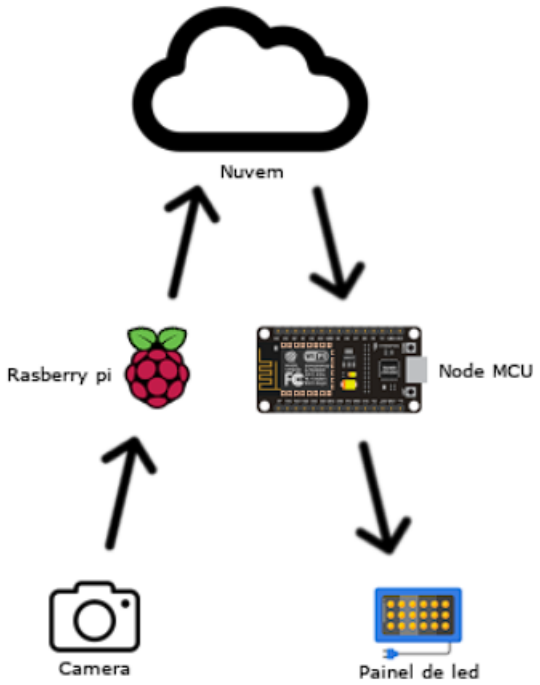

Figura 6: Esquema de funcionamento do estacionamento inteligente.

na análise a ser feita se mostra suficiente para ser considerada básica. Esses círculos são os principais elementos a serem considerados para o entendimento das possíveis falhas do sistema e posteriormente a construção de métodos que tornam o sistema tolerante a falhas. Os conectivos entre os retângulos e círculos são portas lógicas, que podem significar "E" ou "OU", servindo para entender se todos os eventos de entrada necessitam ser verdade para ocorrer a falha acima ou se basta um evento ocorrer para que a falha ocorra.

3.3.2 Solução para recuperação de falhas do sistema

Como todos os sistemas IoT, esse também possui as características de ter dispositivos heterogêneos, com características diferentes e pontos de falhas diferentes. Para isso, a solução proposta para tornar o sistema mais confiável é dividida em implementações separadas.

Falha de conexão com a Internet

Para lidar com as falhas de conexão com a Internet, vamos propor uma solução envolvendo uma comunicação entre a Raspberry Pi e o micro controlador ESP8266. A Figura 8 mostra o esquema de comunicação entre ambos.

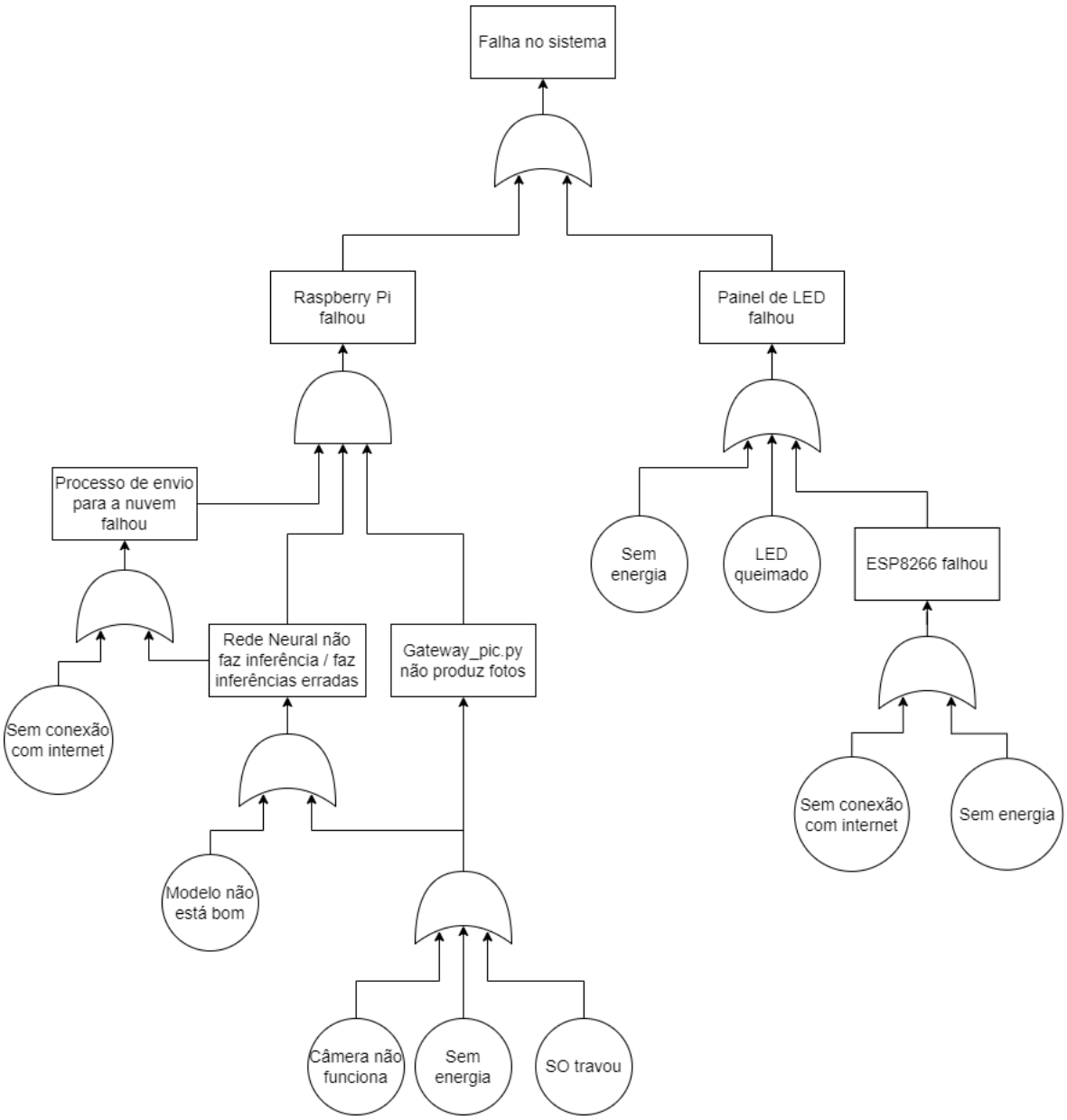

Figura 7: Árvore de falhas do sistema de estacionamento inteligente.

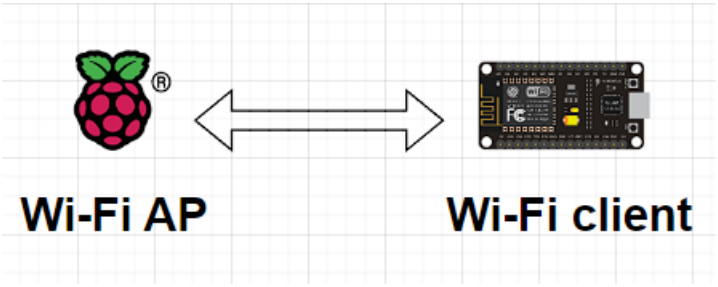

Figura 8: Árvore de falhas do sistema de estacionamento inteligente.

Como o sistema todo está conectado à Internet por meio de uma mesma rede no mesmo roteador Wi-Fi, a solução viável é realizar a comunicação diretamente entre os dispositivos. Como temos somente a camada de armazenamento na nuvem e todo o processamento ocorre na Raspberry Pi localmente, é possível disponibilizar o número de vagas no estacionamento diretamente para o totem. Dessa forma, o processo da Raspberry Pi de envio de dados para a Konker Labs se tornará um Wi-fi Acess Point, onde outros dispositivos podem se conectar localmente. Assim que ambos os dispositivos identificarem que estão sem conexão com a Internet, ativam o modo AP (Raspberry Pi) e cliente (ESP8266) para comunicação entre si. Essa é uma medida temporária, visto que normalmente o problema é transiente.

Para o problema de armazenamento, os dados podem ser guardados localmente na Raspberry Pi para posteriormente serem enviados à nuvem, porém com limitações quanto ao armazenamento interno. Assim que o sistema recupera a conexão com a Internet, novamente os dispositivos se conectam à rede e o serviço se normaliza.

Falhas na Raspberry Pi e na ESP8266

Ambos os dispositivos enviam de tempos em tempos um *heartbeat* mostrando a "saúde", como temperatura do circuito interno e ping (latência da rede). Isso serve para verificar se o dispositivo está vivo e não apresenta falhas de hardware. Essas mensagens, somadas aos dados vindos da inferência, nos permite tirar algumas conclusões sobre o sistema:

- **Recebe *heartbeat*, mas não recebe dados -** significa que a Raspberry Pi está funcionando, porém não está produzindo dados. Podemos tirar a conclusão de que a câmera não está funcionando. Uma solução seria reiniciar o dispositivo e esperar novamente pelo próximo *heartbeat*. Se ocorrer a mesma coisa, significa que uma manutenção no hardware deve ser feita;

- **Recebe *heartbeat*, mas recebe dados errôneos -** significa que o modelo está ruim e deve ser atualizado.

Para receber os *heartbeats*, devemos escolher um cluster head [8], sendo uma opção um script simples de controle de rede na nuvem. Nela, implementamos também a reinicialização dos dispositivos via MQTT ou SSH. Vale também a tentativa de reiniciar os dispositivos quando não recebemos nenhuma mensagem, pois é possível que os processos travaram, não produzindo nenhum tipo de resultado. Ademais, apesar de detectáveis quando a reinicialização não funciona, os problemas como falta de energia não são solucionáveis via aplicação, sendo necessário uma manutenção do equipamento.

4 Conclusão

O avanço tecnológico dos dispositivos de Internet das Coisas permitiu o surgimento de aplicações cada vez maiores e com mais funcionalidades. Com isso, o desafio de deixar os sistemas mais robustos e confiáveis é cada vez maior. O estudo de tolerância a falhas na área de IoT ganhou muita importância, visto que as técnicas e implementações já existentes são insuficientes, pelas características de heterogeneidade dos dispositivos, distribuição do sistema e dinamicidade. Nesse trabalho, analisamos algumas técnicas utilizadas no contexto de IoT, assim como estudos de caso onde essas técnicas são implementadas. Mostramos uma implementação de um estacionamento inteligente onde mapeamos os pontos de falhas do sistema com uma análise de árvore de falhas e apresentamos uma solução para tornar o sistema tolerante a falhas. Para implementações futuras, podemos adicionar um micro serviço realizando, através de aprendizado de máquina, predições para amenizar e prover suporte para uma possível falha antes que ela ocorra.

Referências

[1] Tuan Nguyen Gia, Amir-Mohammad Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Fault tolerant and scalable iot-based architecture for health monitoring. In *2015 IEEE Sensors Applications Symposium (SAS)*, pages 1–6, 2015.

[2] Trista Lin, Herve Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18:3229 – 3253, 04 2017.

[3] Ebenezer Okai, Xiaohua Feng, and Paul Sant. Smart cities survey. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1726–1730, 2018.

[4] Alexander Power and Gerald Kotonya. A microservices architecture for reactive and proactive fault tolerance in iot systems. In *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 588–599, 2018.

[5] Alexander Power and Gerald Kotonya. Providing fault tolerance via complex event processing and machine learning for iot systems. In *Proceedings of the 9th International Conference on the Internet of Things*, IoT 2019, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Sandeep Saharan, Neeraj Kumar, and Seema Bawa. An efficient smart parking pricing system for smart city environment: A machine-learning based approach. *Future Generation Computer Systems*, 106:622–640, 2020.

[7] Sajjad Hussain Shah and Ilyas Yaqoob. A survey: Internet of things (iot) technologies, applications and challenges. In *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pages 381–385, 2016.

[8] Mahyar T. Moghaddam and Henry Muccini. *Fault-Tolerant IoT: A Systematic Mapping Study*, pages 67–84. 09 2019.

[9] Yinhao Xiao, Yizhen Jia, Chunchi Liu, Xiuzhen Cheng, Jiguo Yu, and Weifeng Lv. Edge computing security: State of the art and challenges. *Proceedings of the IEEE*, 107(8):1608–1631, 2019.

[10] Liudong Xing and Suprasad V. Amari. *Fault Tree Analysis*, pages 595–620. Springer London, London, 2008.

[11] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.