Uma Revisão Sistemática da Literatura Sobre Tolerância a Falhas em Internet das Coisas

André F. Pastório*, Luiz A. Rodrigues†, Edson T. de Camargo*†

*Universidade Tecnológica Federal do Paraná (UTFPR) – Toledo – PR – Brasil
†Programa de Pós-Graduação em Ciência da Computação (PPGComp) – UNIOESTE – Cascavel – PR – Brasil
Email: a.f.past@hotmail.com, luiz.rodrigues@unioeste.br, edson@utfpr.edu.br

***Resumo*—A ampla adoção do conceito de Internet das Coisas (IoT) passa por aumentar a confiabilidade e a disponibilidade das suas aplicações perante falhas. A heterogeneidade dos componentes IoT, aliado a dispositivos limitados computacionalmente e integrados à Internet, impõe desafios adicionais para alcançar a tolerância a falhas. Nesse contexto, é essencial investigar como a tolerância a falhas pode ser eficientemente desenvolvida em IoT. Este trabalho apresenta uma revisão sistemática da literatura sobre as principais técnicas e soluções de tolerância a falhas desenvolvidas nesse contexto. Dos 2.004 trabalhos encontrados nos últimos 5 anos, 44 respondem diretamente a questão de pesquisa formulada e são categorizados de acordo com a estratégia de tolerância a falhas utilizada. A análise dos resultados apresenta soluções aplicadas ao dispositivo, conectividade, nuvem e borda, arquiteturas tolerantes a falhas e *blockchain*.**

***Index Terms*—Tolerância a Falhas, Internet das Coisas, IoT, Revisão Sistemática, RSL**

I. Introdução

O mercado global de soluções para usuários finais da Internet das Coisas (IoT) ultrapassou 100 bilhões de dólares em receita de mercado pela primeira vez em 2017 [1]. Há previsão que esse valor crescerá para cerca de 1,6 trilhão de dólares em 2025 [2], chegando a com 41 bilhões de dispositivos em 2027 [3]. Este número deve crescer a medida que a conectividade com a Internet se torna uma característica padrão para uma grande variedade de dispositivos eletrônicos.

A IoT compreende domínios de aplicação diversos, como cidades inteligentes [4], agricultura [5] e saúde [6]. O objetivo é transformar dispositivos comuns do nosso dia a dia em objetos inteligentes ao equipá-los com recursos de identificação, sensoriamento, processamento, comunicação e conexão com a Internet. Com isso, podem se comunicar entre si e com outros dispositivos e serviços pela Internet para alcançar algum objetivo útil [7], [8]. Sensores podem coletar informações do ambiente, como temperatura, umidade, posição geográfica, batimentos cardíacos e imagens de ambientes abertos ou fechados. As informações são enviadas pela rede de comunicação, geralmente sem-fio, para a Internet, onde são processadas e armazenadas em um serviço de nuvem. É possível solicitar remotamente aos objetos inteligentes que executem ações e até mesmo coordenem suas tarefas. A IoT pode ser vista, então, como uma extensão da Internet atual fazendo uso de seus protocolos padrões e de sua característica distribuída.

Embora promissora, concretizar a visão da IoT é uma tarefa árdua devido aos muitos desafios que precisam ser enfrentados, como disponibilidade, confiabilidade, mobilidade, desempenho, escalabilidade, interoperabilidade, segurança e gerenciamento [8], [9]. Entre todos esses desafios, destacam-se a disponibilidade e a confiabilidade, visto que uma falha pode colocar pessoas em perigo, resultar em perda financeira ou danos ambientais [10], por exemplo. De fato, disponibilidade (*availability*) e confiabilidade (*reliability*) são atributos que se referem a confiança no funcionamento de sistemas, ou dependabilidade (*dependability*) [11].

Um dos meios de se alcançar a confiança no funcionamento de um sistema é através de técnicas de tolerância a falhas. Estas técnicas visam garantir o funcionamento correto do sistema mesmo na ocorrência de falhas e são geralmente baseadas em redundância, exigindo componentes adicionais ou algoritmos especiais. Embora a tolerância a falhas em IoT venha sendo explorada em diversos trabalhos [12]–[14], uma questão importante para desenvolver sistemas IoT tolerantes a falhas é conhecer as principais técnicas e soluções capazes de aumentar a confiabilidade e a disponibilidade. Abordagens como redundância e protocolos de consenso são empregadas a várias décadas e tradicionalmente utilizadas em sistemas distribuídos. No entanto, considerando a heterogeneidade e os recursos limitados dos dispositivos (processamento, memória, conectividade, bateria, etc.), é essencial conhecer as soluções que lidam com falhas no contexto de IoT.

O objetivo desse trabalho é identificar as principais abordagens e técnicas de tolerância a falhas que vêm sendo empregadas em IoT. Para tanto, uma revisão sistemática foi executada a fim de encontrar na literatura especializada a resposta para a seguinte pergunta: quais são as principais técnicas e soluções de tolerância a falhas desenvolvidas para IoT? A busca por artigos científicos contendo palavras como "tolerância a falhas" (*fault tolerance*) e "internet das coisas" (*internet of things*) em cinco importantes bases de dados retornou 2.004 trabalhos. Após a primeira e a segunda etapa, nas quais os critérios de exclusão foram aplicados, restaram 86 trabalhos para análise. Dentre esses, buscou-se quais artigos de fato tratam exclusivamente sobre tolerância a falhas em IoT. Um total de 44 foram selecionados para análise e classificadas conforme a solução de tolerância a falhas tem mais impacto.

A organização do trabalho segue da seguinte forma. A

Seção II descreve os conceitos de tolerância a falhas e a arquitetura IoT. A Seção III apresenta a metodologia de pesquisa. A análise dos trabalhos classificados se encontra na Seção IV. Na Seção V conclui-se o trabalho.

II. Conceitos Teóricos

A. Tolerância a Falhas

Com a adoção do conceito de IoT em segmentos como cidades inteligentes, agricultura, segurança e saúde, cresce também a demanda por soluções capazes de lidar com falhas. Em alguns sistemas IoT, uma única falha pode levar a consequências imprevisíveis. De acordo com [11], dependabilidade (*dependability*) é a entrega de um serviço que justifique a sua confiança. É uma propriedade qualitativa que pode ser definida em termos de atributos, meios e ameaças, conforme representado na Figura 1.

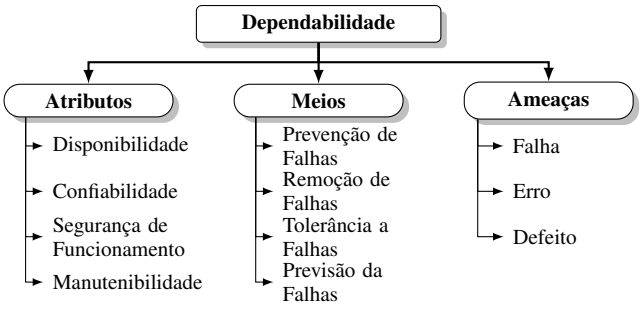


Figura 1: Árvore de dependabilidade. Adaptado de [15].

O atributo de disponibilidade (*availability*) se preocupa com a prontidão do serviço e a capacidade de executar determinada função em um instante de tempo. Confiabilidade (*reliability*) é a capacidade de desenvolver uma determinada função em um determinado intervalo de tempo, dada a importância a continuidade do serviço. Segurança de funcionamento (*safety*) tem como objetivo proteger o ambiente e usuários quanto a perigos causados por falhas. Manutenibilidade (*maintainability*) é a capacidade de restaurar o sistema a um estado anterior de forma a executar uma função, possibilitando também manutenção e reparos.

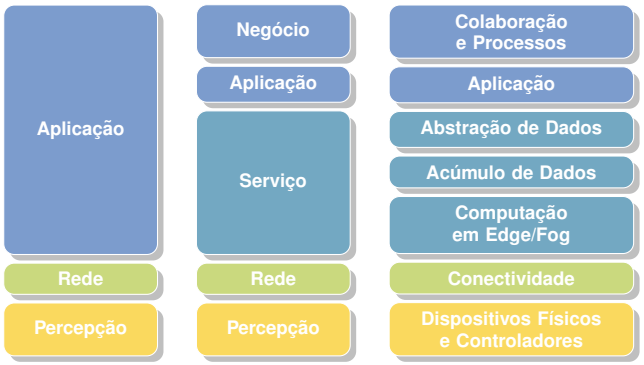
Em relação as ameaças, uma falha (*fault*) pode ocasionar um erro interno (*error*), que pode se manifestar na forma de um defeito ou falha no serviço (*failure*) [15]. Neste sentido, um defeito é uma manifestação da falha no universo do usuário.

Em sistemas baseados em trocas de mensagens, as falhas podem ser categorizadas em dois tipos: falhas de comunicação e falhas de processo [16]. As falhas de comunicação, quando implicam em perda de mensagens, caracterizam-se em omissão e temporização/desempenho [17]. Processos podem ainda falhar por colapso/parada (*crash*), deixando de executar qualquer ação e não respondendo aos estímulos externos. A comunicação pode ainda considerar a possibilidade de falhas de particionamento de rede, que impossibilitam a comunicação entre determinados pares de nós. Em um modelo de falhas mais abrangente, as falhas arbitrárias, também conhecidas como bizantinas, incluem, além das falhas de comunicação e processo, aquelas geradas devido a comportamento malicioso.

Quanto aos meios de se alcançar dependabilidade, a prevenção de falhas diz respeito ao desenvolvimento do sistema de forma a evitar que falhas internas sejam percebidas pela aplicação final. A remoção de falhas é um sistema capaz de detectar e corrigir erros antes que o defeito seja ocasionado. Assim, tolerância a falhas é o uso de técnicas para manter o funcionamento do sistema mesmo após a ocorrência de uma falha e a previsão de falhas é a capacidade de prever uma falha em termos das ameaças [15]. Portanto, a definição do modelo de falhas adotado é crucial para o correto desenvolvimento de uma solução tolerante a falhas.

B. Arquitetura e Padrões IoT

Como afirma [8], existe uma necessidade crítica de uma arquitetura em camadas flexível para IoT capaz de interconectar bilhões ou trilhões de objetos heterogêneos pela Internet. Embora haja diversas arquiteturas de camadas propostas para IoT, nenhuma convergiu para um modelo de referência. Geralmente, a arquitetura de IoT é apresentada em um modelo com três ou cinco camadas [8], [18], embora um modelo de sete camadas tenha sido definido pela Cisco [19] (Figura 2).



(a) 3 camadas (b) 5 camadas (c) 7 camadas

Figura 2: Arquiteturas de camadas para IoT.

No modelo de três camadas [8], [18], a camada de percepção, também chamada de camada de sensoriamento, representa os objetos físicos e é responsável por coletar dados dos sensores e atuadores (informações de temperatura, umidade, peso, movimento, vibração, localização, etc.) e repassá-los para a camada de rede. Os dispositivos IoT apresentam uma arquitetura básica, geralmente composta por unidade de processamento, unidade de comunicação, sensores/atuadores e fonte de energia. A camada de Rede realiza a transmissão dos dados e define o protocolo empregado, a interface de comunicação e o roteamento. É comum o uso de tecnologias sem fio, geralmente voltadas ao baixo consumo de energia, como WiFi, LoRa, Bluetooth Low Energy (BLE), ZigBee e comunicação celular. A terceira camada faz a composição dos dados de forma a entregar um serviço para a aplicação, com foco em protocolos que consomem pouca largura de banda, como CoAP, MQTT, XMPP e AMQP.

No modelo de cinco camadas [20], [21], a camada de Negócio atua como a camada superior, responsável pelo controle e gerenciamento de serviços IoT fornecidos pela camada de Aplicação. A camada de Serviço atua como uma camada intermediária, realizando solicitações de serviços, agregando e processando dados e implementando funcionalidades generalizadas.

O modelo em sete camadas da Cisco [19] mantém as primeiras camadas de percepção e rede, agora como Dispositivos físicos e controladores e Conectividade, respectivamente. A camada de serviço do modelo de cinco camadas é dividida em três. A Computação em Borda ou Névoa, *Edge*/*Fog*, realiza o processamento, análise e filtragem dos dados. Acúmulo de dados se refere ao armazenamento de dados. A Abstração de dados lida com o grande número de dados gerados nas camadas inferiores de forma a fornecer para a aplicação somente os seus dados de interesse. A sétima camada representa as pessoas e processos envolvidos para executar uma tarefa.

III. Metodologia e Processo de Pesquisa

Uma revisão sistemática é uma forma de sumarizar evidências de certa área do conhecimento e identificar questões ainda não pesquisadas, de forma a indicar um caminho a ser seguido [22]. A revisão sistemática executada neste trabalho se baseia na seleção de artigos que respondam a seguinte questão de pesquisa (QP):

QP: Quais são as principais técnicas e soluções de tolerância a falhas desenvolvidas para IoT?

A partir de então, foram definidas as bases de dados de seleção de artigos e as *strings* de busca. As bases de dados foram selecionadas de acordo com os seguintes critérios: (1) mecanismo de busca online; (2) mecanismo de busca avançada e; (3) base reconhecida. As fontes que cumprem os requisitos escolhidas foram IEEEXplore, ACM Digital Library, Springer Link, Science Direct e Portal de Periódicos da Capes. As *strings* de busca relacionam as palavras chave do assunto e foram agrupadas da seguinte forma:

S1 "*Fault tolerance*" AND "*Internet of Things*"

S2 ("*Fault-tolerant techniques*" OR "*fault tolerant technique*" OR "*fault tolerance techniques*") AND "*Internet of Things*"

Note-se que embora S1 e S2 possam formar uma única *string* de busca, S2 é mais restritiva, pois inclui técnicas de tolerância a falhas. A Tabela I apresenta o resultado da pesquisa nas bases de dados selecionadas.

Desde ponto em diante, a metodologia seguiu em três etapas: 1) aplicação de critérios de inclusão ou exclusão, 2) leitura dos títulos e resumos e 3) leitura completa das obras. Na primeira etapa foram excluídos trabalhos anteriores a 2015, revisões (*reviews*, *surveys*), teses e artigos especulativos (*towards*, *new challenges*, etc.). Na segunda etapa, foram selecionados os artigos no qual o título e resumo estão relacionados a questão de pesquisa. Por fim, foram selecionados os trabalhos que tratam do assunto de forma mais incisiva a partir da sua leitura completa.

Tabela I: Quantidade de obras por base (Nov. 2019)

String	IEEE	ACM	Science Direct	Springer Link	Capes
S1	222	433	687	495	101
S2	4	14	28	20	0

A Tabela II apresenta o número de trabalhos aceitos em cada etapa. Inicialmente, 2.004 trabalhos foram encontrados. É possível observar a quantidade expressiva de trabalhos rejeitados de uma etapa para outra. Na primeira etapa, 1.370 trabalhos foram selecionados. Na segunda etapa, após a leitura do título e resumo, restaram 95 trabalhos.

Tabela II: Obras aceitas em cada etapa

Biblioteca	Quantidade	Etapa 1	Etapa 2	Etapa 3
IEEE	226	184	62	55
ACM	447	317	9	9
Science Direct	715	464	11	10
Springer Link	515	359	5	5
Capes	101	46	8	7
Total	2004	1370	95	86

A terceira etapa envolve a leitura completa dos artigos e selecionou 86 trabalhos[1], porém apenas 44 não apenas citam sua proposta como tolerante a falhas, mas também desenvolvem uma solução ou aplicam uma técnica de tolerância a falhas. Por exemplo, o trabalho de [23] consta entre os 86 selecionados na terceira etapa. No trabalho, uma rede sem-fio em malha é a base da proposta. Uma rede em malha inerentemente fornece caminhos redundantes. No entanto, o trabalho não desenvolve ou propõe uma solução de tolerância a falhas. Da mesma forma, o artigo em [24] não foi selecionado pois trata principalmente sobre o posicionamento ideal e seguro do controlador em uma rede de cidades inteligentes baseada em uma Rede Definida por Software (SDN - *Software Defined Network*). A tolerância aparece de forma secundária quando os autores propõem um abordagem já prevista na literatura de SDN para lidar com falhas do controlador.

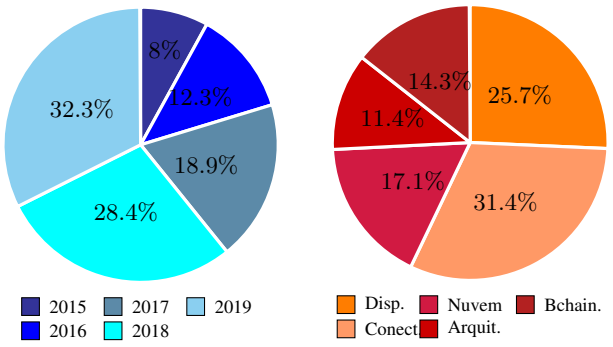
Os trabalhos foram então classificados a partir da camada da arquitetura de IoT na qual a solução de tolerância a falhas tem mais impacto, e analisados na próxima seção.

IV. Análise dos Resultados

Através da revisão sistemática foi também possível identificar o crescimento de pesquisas relacionadas a tolerância a falhas em IoT ao longo dos anos, como mostra a Figura 3(a), referente a etapa 1 da revisão.

Entre os artigos relevantes, apresenta-se os que desenvolvem, propõem ou aplicam uma solução ou técnica de tolerância a falhas. Os trabalhos foram classificados de acordo com a ênfase de atuação da solução de tolerância a falhas empregada dentro das seguintes categorias: dispositivo IoT, conectividade, borda, névoa e nuvem, arquitetura e *blockchain*. A Figura 3(b) ilustra a porcentagem em cada categoria. Devido ao espaço limitado, 35 trabalhos são apresentados nas subseções seguintes de acordo com sua classificação.

[1]Lista disponível em https://github.com/EdanPotter/RevisaoIoT



(a) Por ano (b) Por categoria

Figura 3: Estatística dos trabalhos encontrados.

A. *Dispositivo IoT*

O trabalho em [25] emprega a técnica de replicação de dados em dispositivo de Redes de Sensores Sem Fio (RSSF) homogêneas para IoT. Um arcabouço, chamado DRAW, é apresentado para replicar dados de um dispositivo nos seus dispositivos vizinhos. A solução prevê a troca de mensagens periódica entre nós para disseminar atributos como o espaço disponível em memória de cada dispositivo. O modelo de falha é por colapso e o modelo de sistema adotado não prevê possibilidade de falhas enquanto a replicação acontece.

Em [26], há microsserviços implementados em *containers* no próprio dispositivo IoT. Os microsserviços são responsáveis por interagir com um serviço hospedado na nuvem que, por sua vez, controla os dispositivos por meio dos microsserviços. Cada dispositivo contém um serviço em seu sistema operacional para monitorar as falhas dos microsserviços. Caso a falha de um microsserviço não possa ser reparada, a nuvem desliga o dispositivo e ativa a sua cópia presente em outro dispositivo. O serviço de detecção conta com um servidor NTP para sincronizar os relógios e a solução prevê ainda a transferência de arquivos entre os dispositivos.

O trabalho em [27] apresenta uma abordagem para a implementação do conceito de IoT no domínio militar. O diagnóstico baseado em comparação é usado para detectar as falhas nos processadores dos dispositivos. Nesse método, a mesma entrada é enviada para dois nós e a resposta é comparada. Com base nos resultados das comparações, é possível realizar o diagnóstico. Posteriormente, em um segunda etapa, com a rede particionada em *clusters*, o método é aplicado para diagnosticar em qual subestrutura da rede ocorreu a falha.

Uma abordagem de tolerância a falhas energeticamente eficiente para armazenar os dados do dispositivo em um memória não-volátil é apresentada em [28]. Os autores advogam o empregado de memória não-volátil para salvar os dados do dispositivo (*checkpointing*). Uma vez que falhas ocorram e o dispositivo seja reiniciado, os dados estarão prontamente disponíveis. No entanto, salvar dados em memória não-volátil aumenta a sobrecarga e o consumo energético. Para isso é proposto um algoritmo para selecionar o estado da aplicação com o menor número de dados.

Em [29], os autores criam serviços virtuais como a integração de um ou mais sensores físicos de diferentes modalidades para substituir um sensor físico em caso de falhas. Ou seja, os dados dos diferentes sensores são combinados para prover redundância. Por exemplo, tanto um sensor de presença quanto uma câmera e um microfone podem detectar a presença de pessoas no ambiente. Esses três sensores são então combinados em um sensor virtual. Para combinar diferentes sensores, o trabalho usa métodos de regressão para identificar e gerar os serviços virtuais. O trabalho ainda emprega um algoritmo genético para encontrar as melhores seleções de sensores físicos ou virtuais. Ideia semelhante apresenta o trabalho [30]: um método pressupõe uma lista de recursos do dispositivo e usa essa lista para identificar a sobreposição de recursos entre diferentes dispositivos. As sobreposições podem ser usadas para gerar regras que controlam o estado de energia dos dispositivos, a fim de atingir os objetivos de robustez e eficiência energética, assim como tolerância a falhas.

Uma abordagem de baixo custo para monitorar erros em dispositivos e classificá-los de acordo com a quantidade de erros que exibem para fins de manutenção é apresentada em [31]. Já em [32] há uma abordagem dedicada para confiabilidade em memórias: ao identificar as falhas, uma mapa de memória é criado para evitar acesso a essas regiões. A identificação de sensores falhos é o tema principal do trabalho em [33]. Um sensor quando desligado e retirado da fonte de energia, apresenta uma curva de queda de tensão característica, normalmente devido a capacitância parasita presente no circuito do sensor. Dessa forma é possível identificar a integridade de um sensor através da curva apresentada em leituras periódicas ou ao apresentar uma leitura anormal.

Os trabalhos desta subseção destacam a natureza não confiável dos dispositivos e as condições operacionais adversas. Os dispositivos geralmente são heterogêneos e limitados computacionalmente. Em um mesmo ambiente pode haver uma variedade de *hardware* e interfaces de comunicação, por exemplo. Geralmente, soluções adotadas em RSSF preveem dispositivos homogêneos e nem sempre há comunicação com a Internet. O trabalho [29] explora justamente a heterogeneidade dos sensores. Os trabalhos [25], [26] usam a técnica de redundância no dispositivo. Em [28] a redundância está nos dados que são salvos em memória não-volátil. O trabalho ainda busca diminuir o impacto no consumo de energia na solução proposta. Em [26], apesar de a solução facilitar a gerência dos dispositivos perante falhas, o uso de *containers* exige mais poder computacional do dispositivo e o leva a consumir mais energia. O trabalho [27] emprega uma técnica onde os dispositivos executam testes entre si e comparam os resultados recebidos, o que pode ser uma alternativa para detectar falhas bizantinas. Já os trabalhos em [31]–[33] abordam falhas no dispositivo, memória e sensor.

B. *Conectividade*

Aplicações em redes industriais IoT, incluindo *smart grids*, requerem recursos precisos de sincronização. Nesses sistemas,

a redundância de tempo e dados se torna obrigatória. A solução em [34] é a implementação de protocolo de redundância HSR (*High-availability Seamless Redundancy*) para o *White Rabbit* (WR). O WR é uma tecnologia baseada em Ethernet e uma das suas principais características é a sincronização em sub-nanosegundos. O HSR é usado em topologias de anel e não requer a duplicação de nenhum nó ou enlace, apenas um enlace adicional para fechar o anel.

Em uma rede sem fio, a comunicação pode ser interrompida tanto pela falha do módulo quanto pelo enfraquecimento do sinal, por exemplo. O trabalho em [35] analisa a probabilidade de que todos os nós de uma rede possam se comunicar com todos os outros nós, conceito chamado de *all-terminal reliability*. O estudo é realizado para uma rede sem fio padrão 802.15.4 que assume diversas topologias, incluindo uma rede em malha aleatória. Os autores calculam o tempo médio até uma falha da rede. Para aumentar a confiabilidade é proposto o emprego de redundância de enlace através de uma outra interface de comunicação no dispositivo.

A virtualização de RSSF para IoT é abordada em [36]. O trabalho aborda a falha de comunicação em redes virtuais causadas pela falha de enlace em RSSF físicas. A falha de um enlace nas RSSFs afeta os serviços IoT executados pelas redes virtuais. No trabalho, formula-se um problema de otimização que visa maximizar a tolerância a falhas e minimizar o atraso na comunicação. A solução proposta emprega um algoritmo genético adaptado. Em [37] um algoritmo genético para posicionar deterministicamente nós de uma rede IoT industrial é empregado. O problema abordado se refere a conectividade de sensores em um ambiente com obstáculos. Deseja-se maximizar a vida útil da rede considerando atributos como conectividade, tolerância a falhas, confiabilidade, eficiência energética e capacidade de sobrevivência da rede. Para obter tolerância a falhas considera-se que um conjunto de nós estará ativo enquanto outros ficam inativos. Nós inativos assumirão a computação de nós falhos.

O trabalho em [38] monitora aplicações críticas em RSSF IoT. Apesar de muitos trabalhos na literatura abordarem o monitoramento em RSSF, de acordo com os autores, há poucas pesquisas sobre o monitoramento eficiente de redes IoT em termos de consumo de energia e sobrecarga de comunicação. Os autores afirmam que a abordagem predominante para expandir o tempo de vida da rede é o agendamento do sono (*sleep scheduling*). Nessa abordagem os dispositivos ficam inativos para economizar energia e são ativados para verificar se há ação na rede. Para garantir que cada enlace seja monitorado, os autores propõem um modelo matemático que corresponde a uma otimização multiobjetivo do consumo de energia e a sobrecarga geral de comunicação do monitoramento da rede. A solução é decomposta em três fases onde técnicas da teoria dos grafos, como o problema da cobertura dos vértices e o problema do caixeiro viajante, são empregados na resolução.

Um mecanismo escalável para monitorar as falhas e conservar energia em RSS é o agrupamento (*clustering*) [39]. Na abordagem, os nós dos sensores são organizados em vários grupos. Cada grupo tem um líder, chamado *cluster head* (CH). Os CHs são agregam os dados e os transmitem à estação base (ou *gateway*). O monitoramento é executado separadamente em cada grupo. Ao receber a informação sobre uma falha, o CH também é responsável por substituir o serviço falho por uma cópia. O trabalho em [39] busca minimizar o custo da comunicação do monitoramento de falhas em *clusters*.

O trabalho em [40] lida com a falhas do CH. Para garantir que as mensagens sejam roteadas ao *gateway*, se um CH falhar, seus membros serão gerenciados por CHs sem falhas. Para escolher um novo CH, primeiro os recursos disponíveis de todos os CHs sem falhas são organizados logicamente como um CH virtual para ser o *backup* comum de todos os CHs com falhas. Em seguida, obtém-se a tolerância a falhas com o consumo total mínimo de energia entre todos os CHs sem falhas. Também no contexto de roteamento, o trabalho em [41] propõe um protocolo de roteamento para rede local IoT capaz de encontrar novas rotas caso a rota anterior tenha sido invalidada. Ou seja, o trabalho assume que há enlaces redundantes ou alternativos. Ao reparar as falhas, a rota original é retomada.

Embora haja uma extensa pesquisa sobre protocolos de roteamento em RSSF, a maioria não considera a arquitetura densa de IoT. Em [12] a solução é agrupar nós usando múltiplas estações-base e prioridade de pacotes. A detecção de falhas conta com um pacote especial para notificar o CH caso o nó não tenha dados para enviar. Se o CH não receber nenhum dado ou pacote especial de um dispositivo IoT em um intervalo de tempo, o dispositivo é considerado falho.

Em [42], [43], os autores afirmam que o roteamento tolerante a falhas é frequentemente formulado como um problema de otimização para estabelecer k-caminhos disjuntos que garantam a conectividade mesmo após a falha de até $k-1$ caminhos. A solução gera grande esforço computacional, principalmente se resolvido em dispositivos individuais. No trabalho é desenvolvido um algoritmo de roteamento baseado em otimização de enxame de partículas. O algoritmo é computacionalmente eficiente e capaz de reconstruir os caminhos seguindo parâmetros de qualidade de serviço (QoS) em termos de consumo de energia, atraso e vazão.

Uma características das redes IoT é sua topologia dinâmica e instabilidade devido ao enfraquecimento de bateria, mobilidade, ruído, ou falha no módulo de comunicação. O desafio é ainda maior devido aos enlaces sem fio de curto e/ou longo alcance. Destaca-se o grande número de trabalhos sobre tolerância a falhas no roteamento em IoT. No entanto, a tolerância a falhas impacta em esforço computacional, sobrecarga gerada por troca de mensagens e problemas de escalabilidade conforme o problema aumenta. Trabalhos como [36]–[38], [42] lidam com a tolerância a falhas como um problema de otimização envolvendo o consumo de energia e os custos de comunicação. De fato, muitas soluções de roteamento em IoT tem suas bases em RSSF, como visto em [12], [39], [40].

C. Borda, Névoa e Nuvem

Uma arquitetura confiável e tolerante a falhas para comunicação entre o dispositivo IoT e os vários níveis de computação entre borda e nuvem é definida em [14]. Na arquitetura proposta, a nuvem é distribuída em quatro níveis (nuvem, névoa, neblina e orvalho) com base na capacidade de processamento e distância do dispositivo IoT. Os dados coletados são replicados na borda da rede, ou seja, um dos três primeiros níveis. Na ocorrência de uma falha em um dos servidores da borda, o sistema redireciona a comunicação do dispositivo para um servidor alternativo. O servidor é escolhido no melhor nível possível na hierarquia, com base atraso máximo permitido.

O trabalho em [13] propõe um arcabouço baseado em uma arquitetura de microsserviços. Há dois microsserviços complementares: um que usa processamento de eventos complexos (CEP) para detectar falhas em tempo real e outro que emprega algoritmos de aprendizado de máquina para detectar padrões e mitigar futuras falhas antes que elas ocorram. O primeiro é hospedado na névoa para oferecer rápida detecção e recuperação perante falhas e o segundo é hospedado na nuvem. Os microsserviços recebem informações dos dispositivos e suas ações são baseadas na análise dos dados recebidos. Os mesmos autores também empregam o CEP aliado a ciência do contexto para propor um arcabouço que descreve genericamente as falhas e seus efeitos no sistema [44].

Em um contexto *smart home*, [45] propõe uma plataforma distribuída para aplicações heterogêneas. A ideia central é retirar o ponto único de falha representado na comunicação entre o *gateway* e a nuvem movendo a computação para os dispositivos na rede interna. Os dispositivos são capazes de criar cópias virtualizadas de cada elemento, que são processadas pelos elementos reais e então transmitidas aos virtuais. Na plataforma, as aplicações podem escolher entre serviços de entrega de melhor esforço e entrega confiável. Já [46] trabalha com tolerância a falhas em monitoramento de atividades do cotidiano. Nesse ambiente muitas vezes são utilizados vários sensores para monitorar somente uma atividade. Com o objetivo de diminuir o período de manutenção e manter o funcionamento do sistema com a presença de sensores falhos é proposto um arcabouço para agregar os dados e extrair a redundância funcional presente nos sensores e agendar uma possível manutenção.

Falhas bizantinas na alocação de serviços na névoa é um dos objetivos de [47]. Quando um usuário requisita um serviço o nós principal envia os dados para as réplicas. O número de réplicas é $n \geq 3f + 1$, sendo f o número de nós falhos.

Os trabalhos desta categoria buscam distribuir o processamento centralizado da nuvem em equipamentos mais próximos dos dispositivos IoT. Além da redundância, a estratégia aumenta o desempenho, diminuindo a latência de comunicação e de detecção de falhas locais.

D. Arquitetura

Uma arquitetura escalável e tolerante a falhas no contexto da saúde é proposta em [48]. Para contornar a baixa confiabilidade dos protocolos do padrão IEEE 802.15.4, os autores empregam o protocolo 6LowPAN e propõem uma arquitetura personalizada. Na arquitetura os *gateways* são compostos por nós de processamento, chamados de *sink nodes*, que monitoram o estado do sensores. Os sensores enviam dados para um dos nós de processamento do *gateway*. Ao perceber um sensor inativo, o *gateway* inicia um protocolo para descobrir o motivo da inatividade. Se após a conclusão do protocolo o nó ainda não responder, outro nó de processamento envia uma mensagem de alerta a todos os dispositivos. Se o dispositivo receber a mensagem, passa então a se comunicar com o outro nó de processamento.

A falha e atraso entre a comunicação entre dispositivos, composto por sensor e/ou atuador, e nuvem, é apresentado por [49]. De fato, o trabalho apresenta uma arquitetura de sistema para garantir a continuidade de uma ação de controle, apesar da perda de de comunicação entre o atuador e a nuvem. Basicamente, a solução conta com redundância de nós e enlace. Entre as soluções propostas, caso a falha de um enlace seja detectada na comunicação com a nuvem, um controle local assume e começa a direcionar os dados para o atuador.

Uma arquitetura IoT orientada a serviço para prevenção e previsão de desastres usando aprendizado de máquina é apresentada em [50]. A implementação inclui redundância modular tripla para garantir a disponibilidade e confiabilidade dos dados: três sensores, um principal e dois auxiliares. As leituras dos sensores são comparadas e um esquema de votação é executado na borda para assegurar que os dados encaminhados para a nuvem sejam confiáveis. Em um contexto semelhante, [51] apresenta uma arquitetura e avaliação do mecanismo SENDI, *System for dEtecting and forecasting Natural Disasters based on IoT*. SENDI é um sistema tolerante a falhas baseado em IoT e aprendizado de máquina. A arquitetura é composta por três níveis: Nuvem, névoa e RSSF. A comunicação entre os níveis é projetada para ser tolerante a falhas: quando um dispositivo de um nível falhar, o outro nível pode substituir as suas funções.

Nesta subseção, as soluções encontradas buscam prover tolerância a falhas por meio da organização dos dispositivos e a sua comunicação com a Nuvem, seja pelo monitoramento de atividade ou pelo uso de enlaces e sensores redundantes.

E. Blockchain

Blockchain é uma solução de DLT (*Distributed Ledger Technology*) que define uma rede de transações distribuídas em que os nós transferem blocos de informação, contendo identidade e conteúdo, de forma P2P (*Peer-to-Peer*) [52], [53]. Devido a ser uma rede descentralizada, que provê privacidade, rastreabilidade e controle de acesso, vem sendo usada em várias áreas, especialmente criptomoedas e contratos inteligentes (*smart contracts*).

LVChain [54] é uma solução de *blockchain* para controle de acesso em IoT através de um algoritmo de consenso por votação, mais leve que as soluções tradicionais que utilizam prova de trabalho (PoW - *proof-of-work*) e prova de

participação (PoS - *proof-of-stake*). O objetivo é disponibilizar uma solução descentralizada, escalável, tolerante a falhas e que possa ser executada em modo *off-line*, levando também em consideração a limitação energética dos dispositivos.

O trabalho de [55] propõe uma arquitetura de consenso de duas camadas. A camada base é formada por *end* e *edge-devices*. Alguns dispositivos *edge* com maior capacidade de processamento são escolhidos para formar a camada superior. A camada base lida com blocos de transações, enquanto a camada superior trabalha os blocos de conjunto de transações. A tolerância a falhas é garantida por um algoritmo de consenso não-bizantino. De acordo com os resultados, mesmo com mais de 51% dos nós comprometidos, os membros confiáveis ainda podem convergir.

Já em [56] é definida uma arquitetura multicamadas com *Cloud* e *Fog* integradas. O objetivo é obter o consenso com um número mínimo de troca de mensagens, com garantias de tolerante a falhas. Para isso foi proposto um protocolo de consenso a ser executado na *Fog* (mais próximo do dispositivo final) e, posteriormente, na *Cloud*. Um protocolo de consenso baseado em consistência interativa (*interactive consistency*), que se baseia na maioria, é utilizado.

Em [57] é proposta uma arquitetura de descobrimento de serviços baseada em *blockchain* denominada BlockONS. BlockONS tem o objetivo de cobrir as falhas de segurança existentes no protocolo ONS (*Object Name Service*), que apresenta vulnerabilidades parecidas com as existentes no protocolo DNS (*Domain Name System*). A solução para tolerância a falhas propõe o armazenamento de dados *off-chain* (transações registradas fora da *blockchain*), que podem ser sincronizados com *on-chain* mesmo quando o serviço de um nó falha.

Em sistemas de transporte inteligente, o problema de consenso bizantino é abordado por [58] em uma proposta de protocolo para VANETs (*Vehicular Ad-hoc NETwork*). O consenso em uma rede VANET resulta em uma rede tolerante a falhas, uma vez que veículos são suscetíveis a falhas e o meio de transmissão pode sofrer interferências e ataques. Para isso são realizadas trocas de mensagens em *clusters* através do protocolo proposto, RTAP (*Reliable and Trustworthy Agreement Protocol*), finalizando em um acordo entre os dispositivos operantes, respeitando-se os requisitos do acordo bizantino.

Conforme observado nos trabalhos apresentados, a tolerância a falhas para IoT utilizando a tecnologia de *blockchain* é fornecida por meio de protocolos de consenso, nos quais os valores de entrada dos participantes devem resultar um acordo. Apesar das vantagens, os recursos computacionais limitados presentes em dispositivos IoT fazem com que seja um desafio implementar tal tecnologia na área [55].

V. Conclusão

Este trabalho buscou responder quais são as principais técnicas e soluções de tolerância a falhas desenvolvidas para IoT por meio de uma revisão sistemática da literatura. Inicialmente foram encontradas 2.004 trabalhos. Desses, 86 (4.29%) foram selecionados na terceira etapa. Durante a análise, 44 artigos responderam diretamente a questão de pesquisa formulada. Os artigos foram ainda classificados de acordo com estratégica de atuação utilizada.

A ampla adoção do conceito de IoT passa por impedir que falhas gerem consequências indesejadas ou mesmo desastres. Como visto, dispositivos IoT são, em geral, limitados computacionalmente. A rede, geralmente sem fio, é instável e a comunicação entre os dispositivos e a nuvem gera latência. A revisão encontrou trabalhos desenvolvidos em diversos contextos, desde soluções residenciais à aplicações industriais críticas. A solução de tolerância a falhas empregada corresponde ao contexto da aplicação. Quanto mais crítico, maior é o nível de confiabilidade requerido e, consequentemente, o impacto da solução de tolerância a falhas no custo computacional e energético. Observa-se que uma questão importante de pesquisa na área é desenvolver protocolos e abordagens leves sem abrir mão do nível requerido de confiabilidade e disponibilidade. A técnica de redundância, geralmente empregada em sistemas distribuídos foi encontrada em diversos trabalhos: seja redundância de enlace ou de dispositivo, apesar de acarretar maior custos. Reunir dados heterogêneos de diversas fontes para prover tolerância falhas sem duplicar os recursos pode ser uma estratégia promissora.

Em se tratando de sistemas distribuídos, a soluções encontradas apresentam poucos detalhes em relação ao modelo de comunicação e de falhas. A definição destes modelos têm impacto direto na garantia da corretude dos algoritmos propostos. Como trabalhos futuros, são possibilidades investigar como os protocolos padrões de IoT lidam com falhas, quais técnicas de RSSF são mais promissoras ou adequadas no contexto de IoT e *middlewares* que forneçam tolerância a falhas, interoperabilidade e transparência.

Referências

[1] Bain, "Unlocking opportunities in the internet of things," Online, 2018, acesso em 23/06/2019. [Online]. Available: https://www.bain.com/insights/unlocking-opportunities-in-the-internet-of-things/

[2] Statista, "Forecast end-user spending on iot solutions worldwide from 2017 to 2025," 2020, acesso em 30/03/2020. [Online]. Available: https://www.statista.com/statistics/976313/global-iot-market-size/

[3] vXchnge, "Comprehensive guide to iot statistics you need to know in 2020," 2020, acesso em 30/03/2020. [Online]. Available: https://www.vxchnge.com/blog/iot-statistics

[4] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios," *IEEE Wirel. Comm.*, vol. 23, no. 5, pp. 60–67, Oct. 2016.

[5] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, "Internet of things in agriculture, recent advances and future challenges," *Biosystems Engineering*, vol. 164, pp. 31 – 48, 2017.

[6] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[7] A. Whitmore, A. Agarwal, and L. Xu, "The internet of things–a survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, p. 261–274, Apr. 2015.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[9] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[10] D. Macedo, L. A. Guedes, and I. M. D. Silva, "A dependability evaluation for internet of things incorporating redundancy aspects," *Proc. 11th IEEE Int'l Conf. Net., Sensing and Control*, pp. 417–422, 2014.

[11] A. Avizienis, J. . Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dep. and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[12] N. Nasser, L. Karim, A. Ali, M. Anan, and N. Khelifi, "Routing in the internet of things," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[13] A. Power and G. Kotonya, "A microservices architecture for reactive and proactive fault tolerance in iot systems," in *IEEE 19th WoWMoM*, 2018, pp. 588–599.

[14] J. Grover and R. M. Garimella, "Reliable and fault-tolerant iot-edge architecture," in *2018 IEEE SENSORS*, 2018, pp. 1–4.

[15] K. S. Trivedi, A. Bobbio, and J. Muppala, *Reliability and Availability Engineering: Modeling, Analysis, and Applications*. Cambridge: Cambridge University Press, 2017.

[16] L. Lamport and N. Lynch, "Distributed computing: Models and methods," in *Formal Models and Semantics*, ser. Handbook of Theoretical Computer Science, J. VAN LEEUWEN, Ed. Amsterdam: Elsevier, 1990, pp. 1157 – 1199.

[17] P. Jalote, *Fault Tolerance in Distributed Systems*. USA: Prentice-Hall, Inc., 1994.

[18] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev, and V. H. C. de Albuquerque, "A reference model for internet of things middleware," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 871–883, 2018.

[19] CISCO, "The internet of things reference model," in *WhitePaper*, 2014.

[20] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in *10th Int'l Conf. on Frontiers of Inf. Technology*, 2012, pp. 257–260.

[21] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du, "Research on the architecture of internet of things," in *3rd ICACTE*, vol. 5, 2010, pp. V5–484–V5–487.

[22] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, 08 2004.

[23] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in internet of things," *IEEE Access*, vol. 5, pp. 16 441–16 458, 2017.

[24] S. K. Tarai and S. Shailendra, "Optimal and secure controller placement in sdn based smart city network," in *2019 International Conference on Information Networking (ICOIN)*, 2019, pp. 254–261.

[25] W. B. Qaim and O. Ozkasap, "Draw: Data replication for enhanced data availability in iot-based sensor systems," in *Int'l Conf. DASC/PiCom/DataCom/CyberSciTech*, 2018, pp. 770–775.

[26] A. Celesti, L. Carnevale, A. Galletta, M. Fazio, and M. Villari, "A watchdog service making container-based micro-services reliable in iot clouds," in *IEEE 5th FiCloud*, 2017, pp. 372–378.

[27] J. Chudzikiewicz, J. Furtak, and Z. Zielinski, "Fault-tolerant techniques for the internet of military things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 496–501.

[28] Teng Xu and M. Potkonjak, "Energy-efficient fault tolerance approach for internet of things applications," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[29] S. Zhou, K. Lin, J. Na, C. Chuang, and C. Shih, "Supporting service adaptation in fault tolerant internet of things," in *IEEE SOCA*, 2015, pp. 65–72.

[30] C. Chilipirea, A. Ursache, D. O. Popa, and F. Pop, "Energy efficiency and robustness for iot: Building a smart home security system," in *IEEE ICCP*, 2016, pp. 43–48.

[31] M. D. Gutierrez, V. Tenentes, T. J. Kazmierski, and D. Rossi, "Low cost error monitoring for improved maintainability of iot applications," in *IEEE DFT*, 2017, pp. 1–6.

[32] M. Gottscho, I. Alam, C. Schoeny, L. Dolecek, and P. Gupta, "Low-cost memory fault tolerance for iot devices," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, Sep. 2017.

[33] T. Chakraborty, A. U. Nambi, R. Chandra, R. Sharma, M. Swaminathan, Z. Kapetanovic, and J. Appavoo, "Fall-curve: A novel primitive for iot fault detection and isolation," in *16th ACM SenSys*, ser. SenSys '18. New York, NY, USA: ACM, 2018, p. 95–107.

[34] J. L. Gutiérrez-Rivas, J. López-Jiménez, E. Ros, and J. Díaz, "White rabbit hsr: A seamless subnanosecond redundant timing system with low-latency data capabilities for the smart grid," *IEEE Trans. Ind. Informatics*, vol. 14, no. 8, pp. 3486–3494, 2018.

[35] J. Park, "All-terminal reliability analysis of wireless networks of redundant radio modules," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 219–230, 2016.

[36] O. Kaiwartya, A. H. Abdullah, Y. Cao, J. Lloret, S. Kumar, R. R. Shah, M. Prasad, and S. Prakash, "Virtualization in wireless sensor networks: Fault tolerant embedding for internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 571–580, 2018.

[37] L. Dai, B. Wang, L. T. Yang, X. Deng, and L. Yi, "A nature-inspired node deployment strategy for connected confident information coverage in industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9217–9225, 2019.

[38] B. Mostafa, A. Benslimane, M. Saleh, S. Kassem, and M. Molnar, "An energy-efficient multiobjective scheduling model for monitoring in internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1727–1738, 2018.

[39] S. Zhou, K. Lin, and C. Shih, "Device clustering for fault monitoring in internet of things systems," in *IEEE WF-IoT*, 2015, pp. 228–233.

[40] J. Lin, P. R. Chelliah, M. Hsu, and J. Hou, "Efficient fault-tolerant routing in iot wireless sensor networks based on bipartite-flow graph modeling," *IEEE Access*, vol. 7, pp. 14 022–14 034, 2019.

[41] K. Fan, J. Lu, D. Sun, Y. Jin, R. Shen, and B. Sheng, "Failure resilient routing via iot networks," in *IEEE iThings/GreenCom/CPSCom/SmartData*, 2017, pp. 845–850.

[42] M. Z. Hasan and F. Al-Turjman, "Swarm-based data delivery framework in the ad hoc internet of things," in *GLOBECOM*, 2017, pp. 1–6.

[43] ——, "Optimizing multipath routing with guaranteed fault tolerance in internet of things," *IEEE Sensors Journal*, vol. 17, no. 19, pp. 6463–6473, 2017.

[44] A. Power and G. Kotonya, "Complex patterns of failure: Fault tolerance via complex event processing for iot systems," in *IEEE iThings/GreenCom/CPSCom/SmartData*, 2019, pp. 986–993.

[45] M. S. Ardekani, R. P. Singh, N. Agrawal, D. B. Terry, and R. O. Suminto, "Rivulet: A fault-tolerant platform for smart-home applications," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, ser. Middleware '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 41–54.

[46] P. A. Kodeswaran, R. Kokku, S. Sen, and M. Srivatsa, "Idea: A system for efficient failure management in smart iot environments," in *Proceedings of the MobiSys*. New York, NY, USA: Association for Computing Machinery, 2016, p. 43–56.

[47] J.-w. Xu, K. Ota, M.-x. Dong, A.-f. Liu, and Q. Li, "SIoTFog: Byzantine-resilient IoT fog networking," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1546–1557, Dec. 2018.

[48] T. N. Gia, A. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fault tolerant and scalable iot-based architecture for health monitoring," in *2015 IEEE Sensors Applications Symposium (SAS)*, 2015, pp. 1–6.

[49] S. Sharma, S. Gupta, S. Gupta, and S. B. Kotwal, "Iot based innovative dual level control system with fault tolerance fail safe capability," in *ICICCS*, 2018, pp. 307–312.

[50] A. S. Pillai, G. S. Chandraprasad, A. S. Khwaja, and A. Anpalagan, "A service oriented iot architecture for disaster preparedness and forecasting system," *Internet of Things*, p. 100076, 2019.

[51] G. Furquim, G. P. R. Filho, R. Jalali, G. Pessin, R. W. Pazzi, and J. Ueyama, "How to improve fault tolerance in disaster predictions: A case study about flash floods using iot, ml and real data," *Sensors (Basel, Switzerland)*, vol. 18, no. 3, p. 907, 2018.

[52] T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the use of blockchain for the internet of things," *IEEE Access*, vol. 6, pp. 32 979–33 001, 2018.

[53] E. Leka, B. Selimi, and L. Lamani, "Systematic literature review of blockchain applications: Smart contracts," in *2019 International Conference on Information Technologies (InfoTech)*, 2019, pp. 1–3.

[54] Y. Yu, S. Zhang, C. Chen, and X. Zhong, "Lvchain: A lightweight and vote-based blockchain for access control in the iot," in *IEEE 4th ICCC*, 2018, pp. 870–874.

[55] H. Bai, G. Xia, and S. Fu, "A two-layer-consensus based blockchain architecture for iot," in *IEEE 9th ICEIEC*, 2019, pp. 1–6.

[56] S. Wang, S. Tseng, K. Yan, and Y. Tsai, "Reaching agreement in an integrated fog cloud iot," *IEEE Access*, vol. 6, pp. 64 515–64 524, 2018.

[57] W. Yoon, I. Choi, and D. Kim, "Blockons: Blockchain based object name service," in *IEEE ICBC*, 2019, pp. 219–226.

[58] S.-C. Wang, Y.-J. Lin, and K.-Q. Yan, "Reaching byzantine agreement underlying vanet," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 7, p. 3351, 2019.