UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ
CAMPUS LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO OLIVEIRA DIAS

MONITORAMENTO REATIVO DE PROCOTOLOS EM SISTEMAS IOT SOBRE O MQTT

BANDEIRANTES–PR

2017

GUSTAVO OLIVEIRA DIAS

MONITORAMENTO REATIVO DE PROCOTOLOS EM SISTEMAS IOT SOBRE O MQTT

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Norte do Paraná para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Wellington Aparecido Della Mura

BANDEIRANTES–PR

2017

GUSTAVO OLIVEIRA DIAS

MONITORAMENTO REATIVO DE PROCOTOLOS EM SISTEMAS IOT SOBRE O MQTT

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Norte do Paraná para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. Me. Wellington Aparecido Della Mura
Universidade Estadual do Norte do Paraná
Orientador

Prof. Dr. Bruno Squizato Faiçal
Universidade Estadual do Norte do Paraná

Prof. Me. Carlos Eduardo Ribeiro
Universidade Estadual do Norte do Paraná

Bandeirantes–PR, 24 de Novembro de 2017

DIAS, G. O.. **Monitoramento reativo de procotolos em sistemas IoT sobre o MQTT**. 89 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Norte do Paraná, Bandeirantes–PR, 2017.

RESUMO

Devido ao relevante crescimento da IoT, estima-se uma alta demanda por dispositivos e sistemas que integram esse ambiente, visando otimizar processos industriais, melhorar a qualidade de vida dos usuários, entre outras aplicações. Com tantos dispositivos conectados, a comunicação é um fator crucial para garantia da eficiência dos sistemas em IoT. O MQTT é um protocolo de aplicação que surgiu com o objetivo de solucionar problemas específicos deste setor, como redes com alta latência e taxa de perda de pacotes, além do baixo poder de armazenamento e processamento dos dispositivos. Entre as ferramentas conhecidas que implementam o MQTT, não encontrou-se ao longo do desenvolvimento deste trabalho uma que adote mecanismos de detecção e verificação de falhas e seus responsáveis. Devido a isso, este trabalho tem como objetivo desenvolver um método para monitoramento reativo da comunicação entre dispositivos em sistemas IoT que utilizam o protocolo MQTT. O método é baseado na modelagem do protocolo de comunicação do sistema IoT por meio de um cálculo temporal de processos chamado TPi, de acordo com o comportamento operacional do MQTT com nível de QoS igual a zero. A partir do modelo obtido, é aplicado um método de conversão para representá-lo em forma de contrato multilateral. Em seguida, é executado um algoritmo de monitoramento reativo sobre o contrato que permite a detecção de violações contratuais, assim como a identificação das partes responsáveis pela violação. Neste sentido, este trabalho propõe um estudo de caso real sobre um sistema de detecção e supressão de incêndio. Com a aplicação do método proposto, foi possível aplicar o monitoramento reativo sobre o sistema, assim como realizar a identificação do responsável pela falha contratual que foi simulada. Desta forma, o objetivo do trabalho foi atingido. Enquanto que o modelo em TPi possibilitou uma modelagem simples e abstraída do sistema, o modelo em contrato multilateral garantiu um nível mais elevado de detalhamento das propriedades, essencial para a aplicação do algoritmo de monitoramento reativo. Entretanto, algumas propriedades para representação do modelo em contrato multilateral foram inseridas manualmente, pois não encontrou-se um meio para representá-las por meio do modelo em TPi. Sendo assim, como trabalhos futuros, sugere-se determinar meios para representar as propriedades pendentes, adaptar o método de conversão para casos mais complexos, estender o método proposto para os níveis 1 e 2 de QoS do protocolo MQTT e desenvolver uma implementação do MQTT que aplique o monitoramento reativo entre os dispositivos conectados.

Palavras-chave: Internet das Coisas. Cálculo de Processos. Contratos Multilaterais. Monitoramento Reativo.

DIAS, G. O.. **Title of the Work**. 89 p. Final Project (Bachelor of Science in Computer Science) – State University Northern of Parana , Bandeirantes–PR, 2017.

ABSTRACT

Due to the significant growth of IoT, it is estimated a high demand for devices and systems that integrate this environment, aiming to optimize industrial processes, improve the quality of life of users, among other applications. With so many devices connected, communication is a crucial factor in ensuring the efficiency of IoT systems. MQTT is an application protocol that has emerged with the objective of solving specific problems of this sector, such as networks with high latency and packet loss rate, as well as the low power of storage and processing of the devices. Among the known tools that implement MQTT, it was not found during the development of this work one that adopts mechanisms of detection and verification of failures and their responsible ones. Due to this, this work aims to develop a method for reactive monitoring of communication between devices in IoT systems that use the MQTT protocol. The method is based on the modeling of the communication protocol of the IoT system by means of a temporal calculation of processes called TPi, according to the operational behavior of the MQTT with QoS level equal to zero. From the model obtained, a conversion method is applied to represent it in the form of a multi-party contract. Next, a reactive monitoring algorithm is executed on the contract that allows the detection of contractual violations, as well as the identification of the parties responsible for the violation. In this sense, this work proposes a real case study on a fire detection and suppression system. With the application of the proposed method, it was possible to apply the reactive monitoring on the system, as well as to perform the identification of the person responsible for the contractual fault that was simulated. In this way, the objective of the work was achieved. While the TPi model allowed a simple and abstracted modeling of the system, the multi-party contract model ensured a higher level of detail of the properties, essential for the application of the reactive monitoring algorithm. However, some properties for representation of the model in a multi-party contract were inserted manually, as a way to represent them through the TPi model was not found. Thus, as future work, it is suggested to determine means to represent the pending properties, to adapt the conversion method to more complex cases, to extend the proposed method to the QoS levels 1 and 2 of the MQTT protocol and to develop an implementation of the MQTT that apply reactive monitoring between connected devices.

Keywords: Internet of Things. Process Calculus. Multi-party Contracts. Reactive Monitoring.

LISTA DE ILUSTRAÇÕES

Figura 1 – Escopo do método proposto. 15
Figura 2 – Categorização dos tópicos e tecnologias em IoT [1], adaptado pelo autor. 18
Figura 3 – Três paradigmas de visão para IoT [2], adaptado pelo autor. 19
Figura 4 – Áreas de aplicação e principais contextos [3], adaptado pelo autor. . . 20
Figura 5 – Uso dos protocolos de acordo com a camada *TCP/IP* [4], adaptado pelo autor. 21
Figura 6 – Estratégia requisição/resposta sobre o paradigma cliente/servidor. . . 23
Figura 7 – Exemplo de requisição e resposta HTTP. 24
Figura 8 – Arquitetura CoAP [5], adaptado pelo autor. 25
Figura 9 – Datagrama CoAP [5]. 26
Figura 10 – Exemplo de comunicação MQTT sobre a estratégia *publish/subscribe*. 28
Figura 11 – Estrutura de um pacote MQTT [6]. 29
Figura 12 – Formato do cabeçalho fixo MQTT [6]. 29
Figura 13 – Formato do cabeçalho fixo de um pacote PUBLISH [6]. 30
Figura 14 – Exemplo de diagrama de fluxo para entrega com QoS 0 [6]. 31
Figura 15 – Exemplo de diagrama de fluxo para entrega com QoS 1 [6]. 32
Figura 16 – Exemplo de diagrama de fluxo para entrega com QoS 2 [6]. 34
Figura 17 – Características e propriedades dos protocolos de aplicação em IoT [2], adaptado pelo autor. 35
Figura 18 – Árvore de derivação associada à fórmula $((P \vee Q) \rightarrow (Q \wedge (Q \leftrightarrow P)))$. 41
Figura 19 – Exemplo de um sistema de transição [7]. 42
Figura 20 – Semântica da LTL [7]. 43
Figura 21 – Exemplo de árvore de computação num sistema de transição [7]. . . . 45
Figura 22 – Semântica da CTL [8]. 46
Figura 23 – Exemplo de árvore de computação num sistema de transição. 50
Figura 24 – Passagem de valores em CCS. 51
Figura 25 – Grafo de fluxo do sistema (2.46) [9]. 54
Figura 26 – Semântica Estrutural Operacional do TPi [10]. 56
Figura 27 – Modelo do protocolo MQTTv3.1.1 em TPi baseado nos três níveis de QoS [11], adaptado pelo autor. 58
Figura 28 – Grafo de compromisso $C\_QoS0delivery$. 70
Figura 29 – Procedimento de troca de mensagens do sistema IoT [12], adaptado pelo autor. 75
Figura 30 – Grafo de compromisso do sistema de detecção e supressão de incêndio. 80

Figura 31 – Grafo de compromisso restruturado do sistema de detecção e supressão de incêndio. 83

LISTA DE TABELAS

Tabela 1 – Códigos dos métodos CoAP [5] 26
Tabela 2 – Códigos de resposta CoAP [5] 27
Tabela 3 – Pacotes de controle MQTT [6] 30
Tabela 4 – Leituras dos conectivos proposicionais para a linguagem natural [13]. . 39
Tabela 5 – Tabela verdade dos conectivos proposicionais. 40
Tabela 6 – Tabela verdade associada à fórmula $((P \vee Q) \rightarrow (Q \wedge (Q \leftrightarrow P)))$. . . . 41
Tabela 7 – Semântica Estrutural Operacional do CCS [14]. 50
Tabela 8 – Semântica Operacional Estrutural do Cálculo-π [15]. 53
Tabela 9 – Tabela de comprometimento $C\_QoS0delivery$. 69
Tabela 10 – Protocolo de troca de mensagens para a aplicação IoT sobre o MQTT [12], adaptada pelo autor. 74
Tabela 11 – Tabela de compromisso do sistema de detecção e supressão de incêndio 80
Tabela 12 – Papéis e suas propriedades 82

LISTA DE ABREVIATURAS E SIGLAS

6LoWPAN	*IPv6 over Low power Wireless Personal Area Networks*
AMQP	*Advanced Message Queuing Protocol*
CCS	*Calculus of Communicating Systems*
CoAP	*Constrained Application Protocol*
CoRE	*Constrained RESTful Environments*
CTL	*Computation Tree Logic*
DDS	*Data Distribution Service*
DTLS	*Datagram Transport Layer Security*
EPC	*Electronic Product Code*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IETF	*Internet Engineering Task Force*
IBM	*International Business Machines*
IEEE	*Institute of Electrical and Electronics Engineers*
IoT	*Internet of Things*
IP	*Internet Protocol*
IPv4	*Internet Protocol version 4*
IPv6	*Internet Protocol version 6*
LR-WPANs	*Low-Rate Wireless Personal Area Networks*
LTL	*Linear Temporal Logic*
LTS	*Labelled Transition System*
M2M	*Machine-to-Machine*
MAC	*Media Access Control*
MQTT	*Message Queuing Telemetry Transport*

MQTT-SN	*Message Queuing Telemetry Transport for Sensor Networks*
MTU	*Maximum Transmission Unit*
NFC	*Near Field Communication*
OASIS	*Organization for the Advancement of Structured Information Standards*
PIB	Produto Interno Bruto
QoS	*Quality of Service*
RAM	*Random Access Memory*
REST	*Representational State Transfer*
RFC	*Request for Comments*
RFID	*Radio-Frequency IDentification*
TCP	*Transmission Control Protocol*
TLV	*Type Length Value*
TPi	*Two-Phase Commitment Protocol*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifier*
WLAN	*Wireless Local Area Network*
XMPP	*Extensible Messaging and Presence Protocol*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Motivação 13
1.2 Objetivos gerais e específicos 15
1.3 Organização do Trabalho 15

2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Internet das Coisas 17
2.1.1 Aplicações 19
2.2 Protocolos de comunicação 20
2.2.1 HTTP 22
2.2.2 CoAP 24
2.2.3 MQTT 27
2.2.4 Níveis de *Quality of Service* (QoS) do MQTT 30
2.2.5 Comparação com outros protocolos 34
2.3 Contratos 36
2.3.1 Contratos eletrônicos 36
2.3.2 Verificação de contratos eletrônicos 37
2.4 Representação formal de sistemas 38
2.4.1 Lógica proposicional 38
2.4.2 Lógica temporal 41
2.5 Cálculo de Processos 47
2.5.1 Cálculo de Sistemas de Comunicação 47
2.5.2 Cálculo-π 51
2.5.3 Álgebra temporal de processos TPi 54

3 TRABALHOS RELACIONADOS 57
3.1 Especificação formal de um protocolo de IoT 57
3.2 Um modelo para contratos multilaterais 58
3.3 Teste de violação em contratos multilaterais 61

4 UM MÉTODO PARA MONITORAMENTO REATIVO DO PROTOCOLO MQTT 65
4.1 Análise do modelo e estrutura operacional do QoS 0 66
4.2 Transformando o modelo do MQTT com QoS 0 em um contrato 67
4.3 Algoritmo de conversão de modelo: TPi para contrato multilateral 70

4.4 Algoritmo de monitoramento reativo 72

5 ESTUDO DE CASO: SISTEMA PARA DETECÇÃO E SUPRESSÃO DE INCÊNDIO 74
5.1 Aplicando o algoritmo de conversão 75
5.2 Grafo de compromisso do sistema 79
5.3 Aplicação do algoritmo de monitoramento reativo 81

6 CONCLUSÃO 84

REFERÊNCIAS 86

1 INTRODUÇÃO

O termo Internet das Coisas, do inglês, *Internet of Things* (IoT), foi usado pela primeira vez em 1998 para definir objetos do mundo físico representados virtualmente por meio de dispositivos interconectados [16].

Advinda do avanço das áreas de sistemas embarcados, microeletrônica, comunicação e tecnologia de informações [17], a IoT é considerada uma das mais promissoras tecnologias emergentes [18]. Em estudo realizado pela *Acquity Group*, mais de dois terços dos consumidores planejam comprar tecnologia conectada para suas casas até 2019 [19]. Estima-se que, até 2017, 82% das empresas implementarão a IoT de alguma forma, conforme relatório publicado pela *Business Insider UK* [20].

Por meio da conexão e troca de mensagens entre os dispositivos, é possível realizar o monitoramento de ambientes controlados, dispondo de uma vasta área de aplicações. Entre elas, pode-se destacar: carros, cidades e fazendas inteligentes; transporte e logística; cuidados médicos; interação pessoal e social; e processos industriais [2, 3, 21].

Dentre os diversos fatores responsáveis pelo crescimento do interesse na área, ressalta-se a miniaturização do *hardware*, além da criação do Protocolo de *Internet* IPv6 [19]. Capaz de alocar até 2^{128} endereços IP, o IPv6 possibilita a conexão de uma extensa quantidade de dispositivos [22].

De forma geral, dispositivos utilizados em IoT possuem recursos limitados de memória e processamento, além de operarem em ambientes com baixa largura de banda ou redes instáveis [16, 17, 23]. Portanto, é essencial que a comunicação e a troca de dados entre eles seja executada da forma eficiente, sendo este, um dos principais desafios enfrentados na área [21, 23].

Devido as suas características, as aplicações de IoT necessitam de padronizações específicas. Logo, tratando-se estritamente da comunicação entre os dispositivos, foi desenvolvido em 1999 pela IBM e Eurotech o protocolo MQTT (*Message Queue Telemetry Transport protocol*) [24].

Padronizado em 2014 pela OASIS [6], o MQTT tem como principal objetivo minimizar o uso de largura de banda da rede e recursos dos dispositivos. Para isso, o protocolo foi desenvolvido com base em diversos conceitos que asseguram uma alta taxa de entrega das mensagens [25].

A comunicação é crucial para a execução correta de um sistema em IoT. Mesmo que as especificações do sistema estejam bem definidas, falhas na comunicação estão suscetíveis a influências externas e internas, como interferências e falhas de *hardware*.

Nesse caso, é de interesse do sistema identificar o dispositivo responsável por uma falha, para que assim, o processo seja restabelecido e a falha corrigida.

Além da escolha do protocolo de comunicação adequado, a modelagem da troca de mensagem entre os elementos envolvidos no sistema também deve ser feita, definindo assim, seu protocolo de troca de mensagens. Uma vez modelado, o protocolo encontra-se formalizado, garantindo maior confiabilidade e exatidão em sua interpretação. Desta forma, processos de verificação formal podem ser aplicados.

Um modelo pode ser obtido mediante o uso de um cálculo de processos. O cálculo temporal de processos TPi [10], estende as propriedades do CCS [14] e do Cálculo-π [9], permitindo uma modelagem de alto nível da comunicação entre processos concorrentes com a adição do aspecto temporal. Porém, o TPi não é capaz de formalizar informações importantes como descrição detalhada das ações do sistema, conteúdo das mensagens e ordenação da execução dos sistemas.

Por outro lado, na formalização por meio de conceitos de contratos eletrônicos [26], é possível obter um nível de detalhamento maior, ao mesmo tempo que se mantém um certo nível de abstração. No caso de contratos multilaterais [27], quando há múltiplos agentes envolvidos em relações contratuais complexas, as propriedades da formalização permitem a aplicação de técnicas para verificação formal, como o monitoramento reativo para detecção das partes responsáveis por violações contratuais.

Este trabalho apresenta uma estratégia para monitoramento da comunicação entre dispositivos que utilizam o protocolo MQTT [6]. A abordagem é baseada na modelagem da comunicação por meio do cálculo temporal de processos TPi [10]. A partir do modelo obtido, é aplicado um método de conversão para representá-lo em forma de contrato multilateral [27]. Com base nisso, é executado um algoritmo de monitoramento [27, 28] sobre o contrato, permitindo, desta forma, a detecção de violações contratuais, assim como a identificação das partes responsáveis pela violação.

1.1 Motivação

Como dito anteriormente, a IoT é uma das mais promissoras tecnologias emergentes, estima-se que até 2022 ela irá adicionar cerca de 14,4 trilhões de dólares ao PIB global, e que em 2020 haverá em torno de 50 bilhões de dispositivos interconectados [29]. Assim, aumenta-se a demanda por sensores, dispositivos de rede sem fio, serviços de armazenamento em nuvem, entre outros serviços e tecnologias que compõem este ambiente. Desta forma, desperta-se o interesse dos setores industriais, comerciais e acadêmicos.

A comunicação é um fator crítico num projeto em IoT, pois a troca de dados e informações entre os dispositivos é essencial para seu sucesso. Com a heterogeneidade de dispositivos que compõem o ambiente, padronizações são necessárias para manter a inte-

roperabilidade entre os mesmos, sendo essa uma das funções que os protocolos pretender cumprir.

O protocolo MQTT foi desenvolvido especialmente para aplicações em IoT e M2M (*Machine-to-Machine*). Com um cabeçalho de apenas 2 *bytes*, e implementando um servidor próprio, chamado de *broker*, é capaz de prover a troca de mensagens entre dispositivo por meio da estratégia *publish/subscribe*, minimizando o uso de banda de rede e recursos do dispositivo [6].

Num projeto de IoT, além de escolher o protocolo adequado, também e necessário definir o protocolo de troca de mensagens entre os elementos do sistema, que serão trocadas sobre um protocolo de comunicação, sendo que, cada protocolo possui regras internas distintas para definir seu método de comutação de pacotes.

Uma forma de modelar o protocolo de troca de mensagens da aplicação é por meio de métodos formais. Os cálculos de processos e os contratos eletrônicos apresentam-se como soluções para formalização de regras operacionais que devem ser executadas pelas partes envolvidas em uma relação. No caso dos sistemas em IoT, as partes podem ser constituídas por dispositivos e softwares.

Falhas na comunicação entre os dispositivos podem acarretar sérios problemas à eficiência do sistema. Então, é de interesse do sistema identificar as partes responsáveis pela falha, para que assim, a mesma possa ser corrigida e a comunicação restabelecida.

O cálculo temporal de processos TPi [10] oferece um alto nível de abstração para a modelagem da comunicação entre processos concorrentes, porém, não é capaz de formalizar algumas propriedades. Já um contrato eletrônico multilateral [27] oferece suporte para uma modelagem mais complexa e precisa, de modo que possibilita a aplicação de métodos formais para monitoramento reativo, permitindo a detecção de falhas, assim como a identificação das partes responsáveis.

As implementações do MQTT conhecidas na literatura consistem estritamente em implantar as propriedades, regras e normas estabelecidas na especificação do protocolo [30]. De acordo com a revisão bibliográfica realizada durante o desenvolvimento deste trabalho, não encontrou-se uma implementação que aplique métodos formais para detecção e identificação das falhas e seus responsáveis.

Na Figura 1 é apresentado o escopo deste trabalho em relação aos cálculos e lógicas para especificação de comunicação entre processos e contratos, além das técnicas para formalização do protocolo. As elipses representam as lógicas envolvidas no trabalho enquanto que o retângulo indica a técnica para modelagem e formalização do protocolo. Os retângulos hachurados em cinza indicam a proposta do trabalho com o método de transição dos modelos e o mecanismo de detecção e identificação de violações contratuais e das partes responsáveis.

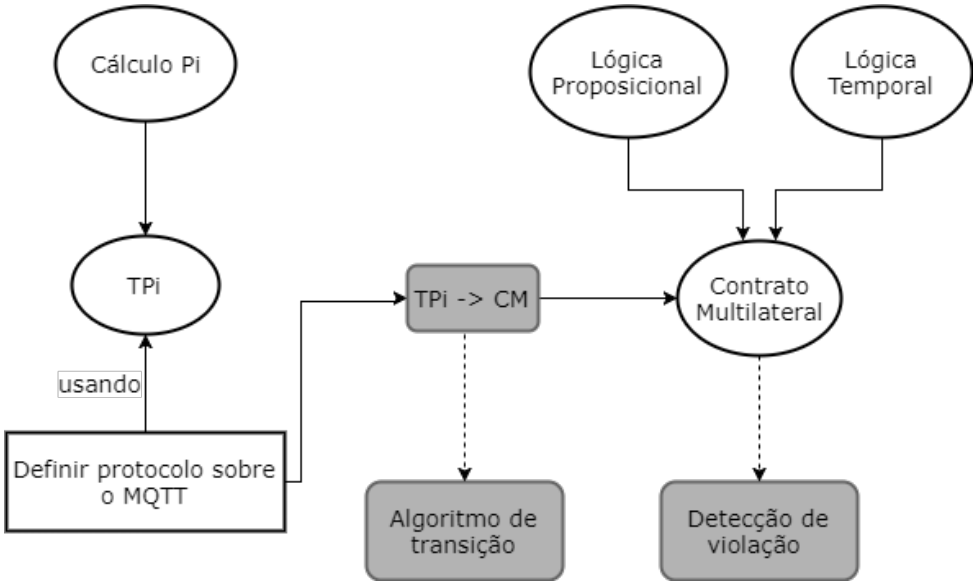

Figura 1 – Escopo do método proposto.

Fonte: Próprio autor.

1.2 Objetivos gerais e específicos

O objetivo geral deste trabalho é propor um mecanismo para formalização e monitoramento reativo de sistemas de IoT que utilizam o protocolo MQTT. Existem várias aplicações do protocolo MQTT que implementam seus requisitos normativos. No entanto, a partir da revisão bibliográfica realizada durante o desenvolvimento deste trabalho, não encontrou-se uma abordagem para detecção de violações ou falhas, assim como a identificação das partes responsáveis.

Os objetivos específicos para se atingir o propósito deste trabalho são:

- Determinar o comportamento operacional do MQTT de acordo com o modelo em TPi;
- Definir um método de conversão do modelo em TPi para um contrato multilateral;
- Aplicar o método proposto a um estudo de caso real.

1.3 Organização do Trabalho

A organização do trabalho obedece a estrutura a seguir. No Capítulo 2 são apresentados aspectos gerais sobre Internet das Coisas e protocolos de comunicação específicos, referencial teórico sobre contratos e cálculo de processos, além de formalismos para a representação formal de sistemas, contratos eletrônicos e processos. O Capítulo 3 apresenta os trabalhos mais intimamente relacionados com a pesquisa apresentada neste documento.

O Capítulo 4 expõe o método proposto para monitoramento reativo, descrevendo o modelo utilizado como base para a análise do protocolo MQTT, o algoritmo de conversão do modelo em TPi para contrato multilateral e o mecanismo de monitoramento. O Capítulo 5 propõe um estudo de caso real, assim como a aplicação do método proposto sobre o caso. Finalmente, o Capítulo 6 descreve as considerações finais do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta alguns conceitos envolvendo IoT, como suas características, tecnologias envolvidas, poder de mercado e aplicações. A comunicação em IoT é discutida mais profundamente e os principais protocolos de comunicação são descritos e discutidos, em especial o MQTT, foco de pesquisa deste trabalho. Algumas representações para contratos eletrônicos, sistemas e processos concorrentes também são descritas, fornecendo um conjunto de conceitos necessários para a solução dos problemas propostos neste trabalho.

2.1 Internet das Coisas

Com a popularização das tecnologias de comunicação (*Wi-Fi, Bluetooth, 3G/4G, etc*), dos dispositivos conectados a *internet*, juntamente com a difusão do IPv6, uma série de aplicações e tecnologias passam a ser exploradas.

Neste contexto, surge o conceito de Internet das Coisas, caracterizado por dispositivos inseridos na rede mundial de computadores para adquirir e trocar dados que devem ser processados, gerando informação e conhecimento que auxilia na tomada de decisões.

Pode-se destacar a heterogeneidade como uma das principais características da IoT, pois promove a integração de diversos tipos de dispositivos e tecnologias, como sensores, dispositivos de rede, *smartphones*, bancos de dados, protocolos, entre outros.

Em seu trabalho, Mayer[1] classifica a IoT em 8 tópicos:

- **Comunicação:** permite a troca de informações entre dispositivos;
- **Sensores:** captura e representa o mundo físico no mundo digital;
- **Atuadores:** realiza ações no mundo físico desencadeadas no mundo digital;
- **Armazenamento:** coleta de dados a partir de sensores, sistemas de identificação e rastreamento;
- **Dispositivos:** provê interação com humanos no mundo físico;
- **Processamento:** fornece mineração de dados e serviços;
- **Localização e rastreamento:** determinação e rastreamento de localização do mundo físico;
- **Identificação:** concede uma única identificação física de um objeto no mundo digital.

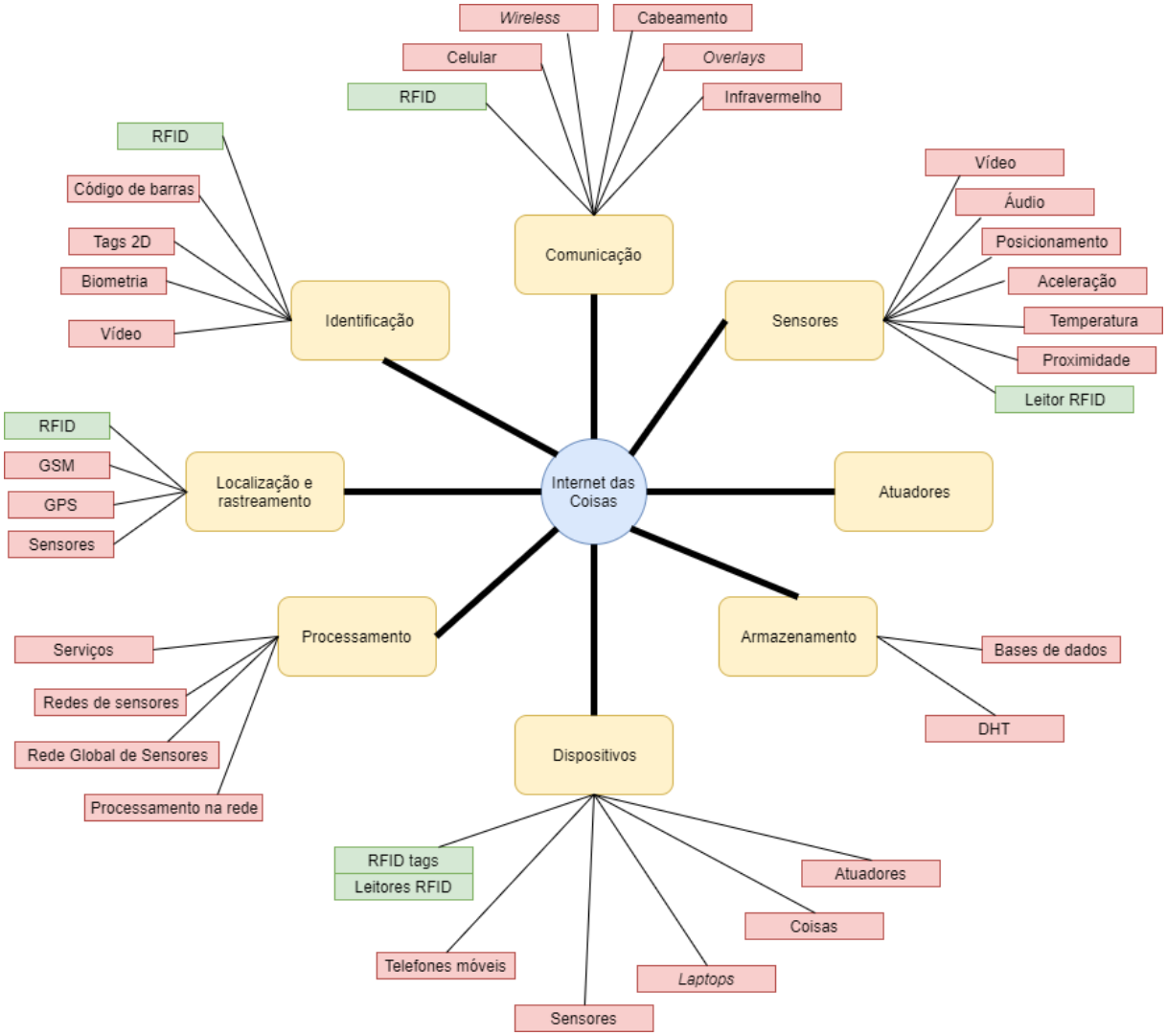

Figura 2 – Categorização dos tópicos e tecnologias em IoT [1], adaptado pelo autor.

A Internet das Coisas é um paradigma que desempenha e continuará desempenhando um papel vital num futuro próximo [2]. Sankar e Srinivasan[2] classificam a IoT sob a perspectiva de visão, destacando três categorias: visão orientada a coisas; visão orientada a *internet*; e visão orientada a semântica. Os paradigmas de visão são expostos na Figura 3.

Visão Orientada a Coisas: realiza o rastreamento ou monitoramento de objetos usando sensores e tecnologias ubíquas. O Código de Produto Eletrônico, do inglês *Electronic Product Code* (EPC), costumava ser usado para identificar objetos, e tal técnicas foi estendidas para os sensores.

Visão Orientada a Internet: objetos físicos são convertidos em objetos inteligentes. Um protocolo de *internet* é atribuído a um objeto, que é acessado de forma exclusiva. O objeto habilitado pelo sensor torna-se inteligente, o qual é identificado exclusivamente e monitorado de forma contínua.

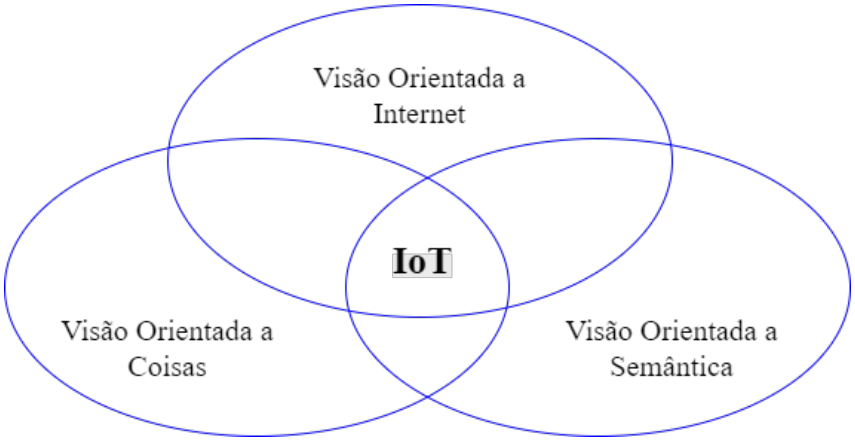

Figura 3 – Três paradigmas de visão para IoT [2], adaptado pelo autor.

Visão Orientada a Semântica: tem-se uma grande quantidade de objetos habilitados por sensores conectados a *internet*. Os sensores monitoram e geram continuamente os dados, que podem ser redundantes, além de serem heterogêneos ou homogêneos. A visão semântica ajuda a processar os dados de forma significativa e tomar as decisões necessárias adequadamente.

2.1.1 Aplicações

As potencialidades oferecidas pela IoT possibilitam uma vasta área de aplicações, das quais poucas são realmente aplicadas e comercializadas. Tais aplicações proverão maior eficiência e conforto em atividades como: fazer uma viagem de carro, cuidar da saúde, cultivar vegetais, trabalhar, praticar exercícios físicos, entre muitas outras [3, 21].

Estes ambientes podem ser equipados com objetos que comunicam-se para produzir informações de acordo com a percepção do ambiente ao redor. Isso implica numa gama de aplicações que podem ser desenvolvidas. Algumas destas são agrupadas nos seguintes tópicos:

- **Transporte e logística:** Carros, aviões, ônibus, navios, bicicletas, estradas, portos e trilhos tornam-se mais monitoráveis com o uso de sensores, atuadores e processamento de dados. Assim, os meios para transporte e os bens transportados enviam informações para sistemas de controle de tráfego e de veículos para melhor direcionamento do trânsito, ajudando no controle e status das entregas e na exibição de informações aos motoristas acerca do tráfego [3].

- **Saúde conectada:** Considerada uma revolução na área da medicina, envolve a coleta de dados por meio de sensores fixados ao corpo dos pacientes. Os dados são enviados para sistemas que realizam o monitoramento remoto da saúde. Deste modo,

informações sobre a saúde dos pacientes podem ser vistas remotamente pelos seus familiares, pelos médicos e pelo próprio paciente [2].

- **Domínio pessoal e social:** Aplicações com objetivo de motivar o usuário a manter e construir novas relações pessoais. Mensagens podem ser enviadas para que as pessoas saibam o que os amigos estão fazendo, quais seus gostos em comum e demais informações. O domínio pessoal pode envolver, por exemplo, o rastreamento de objetos pessoais em caso de perda ou roubo [3].
- **Ambientes inteligentes:** Carros conectados que emitem alertas de segurança ao motorista. Fazendas com sistemas de agropecuária e agricultura de precisão, visando aumentar a produção utilizando menos tempo e recursos. Processos industriais monitorados para maior eficácia na realização dos procedimentos. Casas conectadas, onde o morador pode controlar o funcionamento de eletrodomésticos, luzes, eletroeletrônicos e demais aparelhos, provendo conforto e qualidade de vida [2, 3, 21]. Todos os casos acima são exemplos de como a IoT pode auxiliar no aperfeiçoamento de processos envolvendo diversos ambientes.

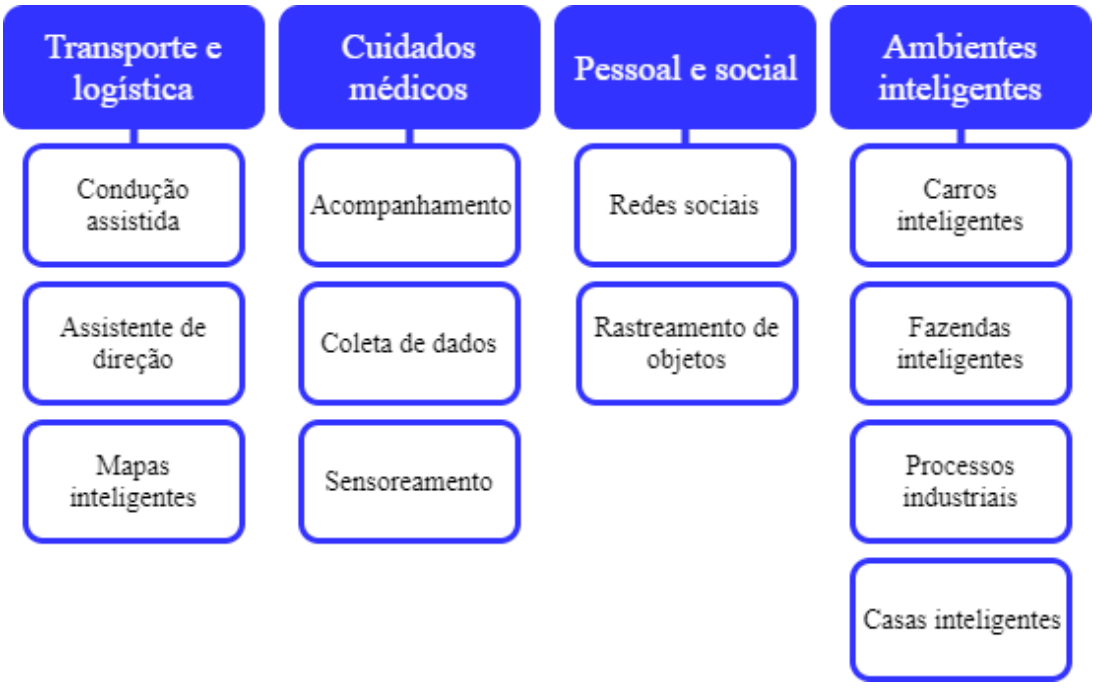

Figura 4 – Áreas de aplicação e principais contextos [3], adaptado pelo autor.

2.2 Protocolos de comunicação

Perante a perspectiva da comunicação, a Internet das Coisas pode ser vista como um conjunto de diferentes redes, incluindo redes móveis (3G, 4G, etc.), *WLANs*, redes de sensores e redes *Adhoc* móveis [4].

Sendo assim, nota-se a existência de uma série de protocolos de comunicação disponíveis, considerando os diferentes tipos de redes que podem ser usadas. Logo, a escolha de um protocolo pode afetar diretamente fatores como a velocidade, confiabilidade e durabilidade de uma conexão.

Na Figura 5, alguns protocolos são exibidos de acordo com a camada no qual é aplicado de acordo com o modelo *TCP/IP*.

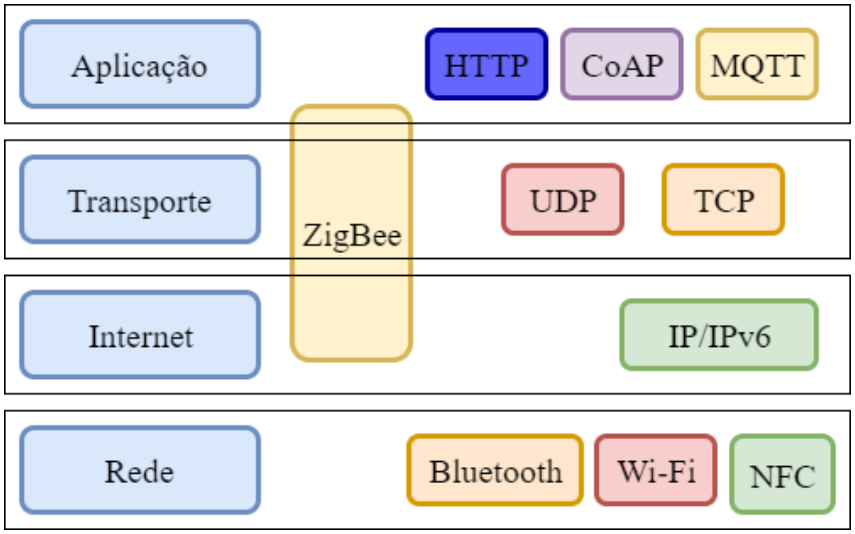

Figura 5 – Uso dos protocolos de acordo com a camada *TCP/IP* [4], adaptado pelo autor.

No entanto, alguns protocolos que são amplamente usados em redes predominantemente formada por Computadores Pessoais, como o *Internet Protocol version 6 (IPv6)* e o *Hypertext Transfer Protocol (HTTP)*, não possuem a mesma usabilidade para a IoT, devido ao poder computacional restrito dos dispositivos [17].

O protocolo *IPv6* representa um passo importante na evolução da IoT, visto que utiliza 128 bits para endereçamento de uma imensa quantidade de objetos, ante aos 32 bits utilizados pelo *IPv4*. Entretanto, o *IPv6* não se adequa as limitações de grande parte dos dispositivos aplicados na IoT. Por exemplo, enquanto que o *IPv6* necessita de 1280 bytes de *MTU* (*Maximum Transmission Unit*), dispositivos que seguem o padrão IEEE 802.15.4 (*LR-WPANs*) limitam-se a pacotes de 128 bytes [17].

Objetivando resolver esses problemas, o *IETF* desenvolveu o *6LoWPAN*, uma camada de adaptação desenvolvida primordialmente para o padrão IEEE 802.15.4. Sua principal finalidade consiste na compressão de pacotes *IPv6* utilizando informações de protocolos presentes em outras camadas, como o endereço MAC, por exemplo. Desta forma, permite o uso do *IPv6* por dispositivos com baixo poder computacional [31].

Outro protocolo extensamente utilizado, mas que mostra-se inadequado para aplicações em IoT é o HTTP [17]. Porém, os protocolos CoAP (*Constrained Application*

Protocol) e MQTT (*Message Queue Telemetry Transport*), ideais para aplicações em IoT, foram desenvolvidos para lidar com a comunicação entre dispositivos com limitações de processamento inseridos em ambientes com redes instáveis e baixa largura de banda.

As características gerais do HTTP serão discutidas na Seção 2.2.1 a seguir. Adiante, nas Subseções 2.2.2 e 2.2.3, serão abordadas características gerais do protocolo CoAP e propriedades gerais e específicas do MQTT, respectivamente. Por ser um dos principais focos de estudo deste trabalho, o MQTT será discutido com maior profundidade em relação ao HTTP e ao CoAP.

Por fim, a Seção 2.2.5 exibe um breve comparativo entre as características gerais dos três protocolos em relação a usabilidade para IoT.

2.2.1 HTTP

O protocolo HTTP é utilizado na camada de aplicação para receber e enviar informações na rede mundial de computadores usando a estratégia requisição/resposta sobre o paradigma cliente/servidor [32], como pode ser observado na figura 6.

Utilizado em sistemas de informação de hipermídia, distribuídos e colaborativos, o HTTP prove comunicação por meio da troca ou transferência de hipertexto, que é um texto estruturado que utiliza ligações lógicas (*hiperlinks*) entre nós contendo texto [32].

A comunicação entre o cliente e o servidor é feita mediante a mensagens, que são codificadas por meio da linguagem de marcação HTML [33]. O cliente envia uma mensagem de requisição de um determinado recurso ao servidor, que recebe e requisição e envia uma resposta, como ilustrado na Figura 6. As mensagens possuem cabeçalho e corpo, além de um método, no caso das mensagens de requisição [32]. Os principais métodos são listados abaixo:

- **GET:** Solicita a exibição de um recurso;
- **POST:** Envia informações do corpo da requisição que serão utilizadas para criar um novo recurso;
- **DELETE:** Remove um recurso;
- **PUT:** Atualiza um recurso na URI (*Uniform Resource Identifier*) especificada. Se o recurso não existir, o mesmo pode ser criado;
- **HEAD:** Retorna informações sobre um recurso.

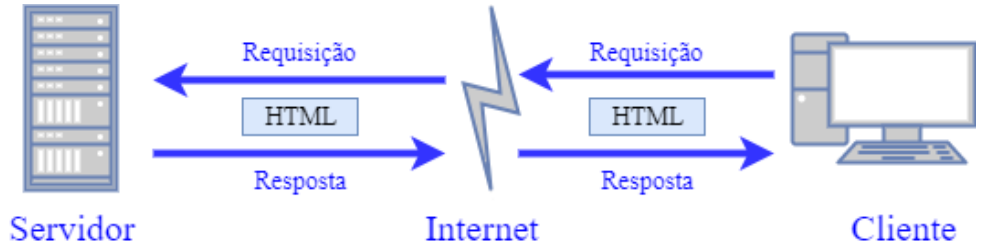

Figura 6 – Estratégia requisição/resposta sobre o paradigma cliente/servidor.

Fonte: Próprio autor.

O cabeçalho da mensagem HTTP é usado para transmitir mensagens adicionais entre o cliente e o servidor. Ele é especificado imediatamente após a linha inicial da transação que contêm o método, tanto para a requisição do cliente quanto para a resposta do servidor. Existem quatro tipos de cabeçalhos que podem ser incluídos nas mensagens [32], os quais são:

- ***General-header***: Utilizado para enviar informações adicionais sobre a mensagem transmitida;
- ***Request-header***: Presente em mensagens de requisição, permite ao cliente transmitir informações adicionais relacionadas à requisição, ao próprio cliente ou ao servidor;
- ***Entity-header***: Pode estar presente tanto em uma requisição quanto numa resposta. Usado para definir meta-dados sobre o corpo da mensagem, como tamanho, tipo e linguagem. Quando não há corpo na mensagem, os meta-dados referem-se aos recursos identificados pela requisição.
- ***Response-header***: Utilizado para transmitir informações adicionais da resposta. As informações podem estar relacionadas com o servidor ou sobre acesso adicional ao recurso identificado pelo *Request*-URI.

Uma mensagem HTTP pode conter um corpo de dados que são inseridos abaixo das linhas de cabeçalho. Em uma mensagem de resposta, o corpo da mensagem é o recurso que foi requisitado pelo cliente, ou ainda uma mensagem de erro, caso este recurso não seja possível. Já em uma mensagem de requisição, o corpo pode conter dados que serão enviados diretamente pelo usuário ou um arquivo que será enviado para o servidor [32].

Na Figura 7 é apresentado um exemplo de requisição e resposta HTTP. À esquerda, é apresentado uma requisição, em que a primeira linha contêm o método da requisição (POST), a URI ("*/servlet/default.jsp*") e a versão do protocolo ("*HTTP/1.1*"). A linha de requisição é seguida por uma série de informações adicionais do cabeçalho, e por fim, na última linha, a informação "*LastName=Magalhaes&FirstName=Guilherme*" pertence

ao corpo da mensagem. No exemplo de resposta HTTP, localizado à direita, a primeira linha armazena o status do protocolo [32], com informações acerca da versão do protocolo ("HTTP/1.1"), o status do código ("*200*" = bem sucedido) e uma *Reason-Phrase* [32] ("*OK*") que indica uma breve descrição textual sobre o status do código. Posterior a linha de status, tem-se as informações do cabeçalho, que é semelhante as da requisição. Afinal, o corpo da resposta é o conteúdo HTML da própria resposta.

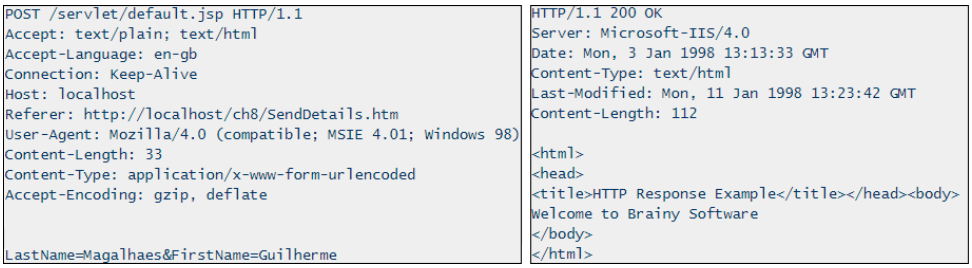

Figura 7 – Exemplo de requisição e resposta HTTP.

Fonte: <http://protocoloti.blogspot.com.br/2012/11/http-requisicao-e-resposta.html>.

2.2.2 CoAP

Definido em 2010 pela IETF, o CoAP tounou-se um *Request for Comments* (*RFC*) em 2014 [5]. Inicialmente, um grupo de trabalho criado pela IETF chamado *Constrained RESTful Environments* (*CoRE*) iniciou suas atividades em 2010 com o objetivo de desenvolver um *framework* para aplicações que manipulam recursos simples localizados em dispositivos interconectados por meio de redes limitadas [5].

Tais aplicações vão desde monitoramento de temperatura e medidores de energia por meio sensores, até o controle de atuadores como interruptores ou trancas eletrônicas, além do gerenciamento de dispositivos que compõem a rede. Assim sendo, o CoAP trata de uma parte deste *framework* [5].

Segundo o RFC 7252 [5], o CoAP é um protocolo de comunicação voltado para objetos limitados (com pouca memória RAM, por exemplo) e redes restritas onde há frequente perda de pacotes. Projetado para aplicações *Machine-to-Machine* (M2M), o CoAP define quatro tipos de mensagens:

- ***Confirmable (CON)*:** Mensagens que precisam ser confirmadas no destino. Se não houver perda de pacotes, cada mensagem deste tipo resulta em uma mensagem do tipo *Acknowledgment* ou *Reset*;
- ***Non-confirmable (NON)*:** Mensagem que não necessita de confirmação de recebimento.
- ***Acknowledgment (ACK)*:** Confirmação de uma mensagem *Confirmable*.

- ***Reset (RST)*:** Indica que outra mensagem CON ou NON foi recebida, mas não pôde ser processada devido a falta de algum contexto.

O CoAP oferece uma estratégia de interação requisição/resposta entre os dispositivos das aplicações e inclui recursos *Web* como URIs e tipos de mídias usadas na *Internet*, além de interagir com o HTTP para integração com a *Web*. Outras características descritas no RFC 7252 [5] são citadas abaixo:

- Troca de mensagens assíncrona;
- Capacidades simples de *proxy* e *caching*;
- Mapeamento HTTP que permite que *proxies* possam prover acesso aos recursos do CoAP via HTTP;
- Interligação segura para *Datagram Transport Layer Security* (DTLS);
- *Binding* em *Datagram Transport Layer Security* (UDP) com confiabilidade opcional suportando requisições tanto *unicast* quanto *multicast*;
- Suporte aos métodos GET, POST, PUT e DELETE. de maneira uniforme

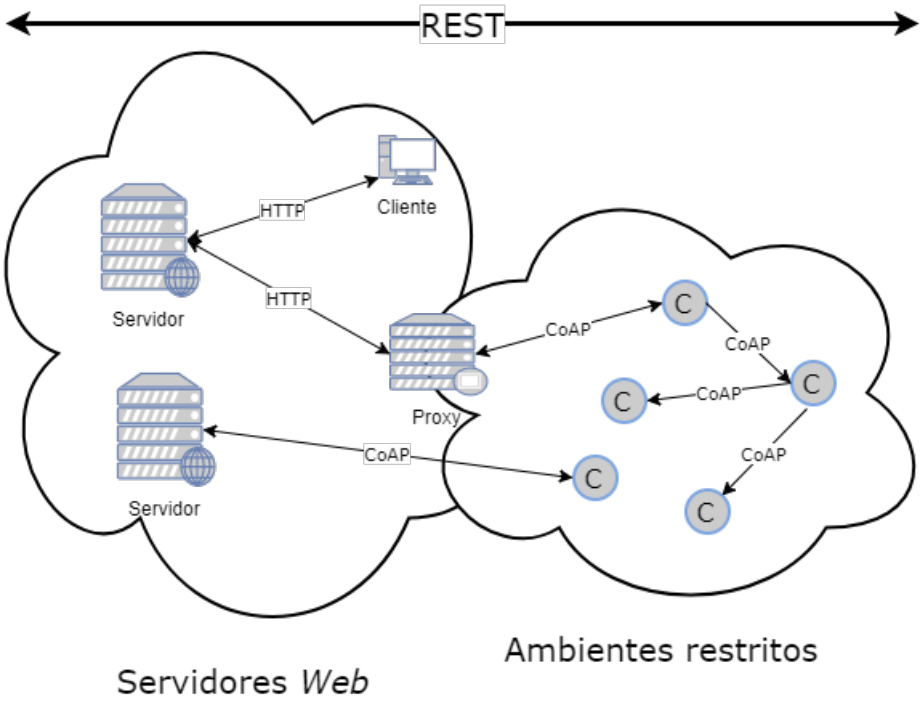

Figura 8 – Arquitetura CoAP [5], adaptado pelo autor.

Na Figura 8 é apresentada uma abstração da arquitetura CoAP. Um dos objetivos do CoRE embasa-se em adequar a arquitetura *Representational State Transfer* (REST)

para ambientes restritos aos nós e rede, e, baseado nisso, foi desenvolvido o CoAP. Sendo assim, o CoAP pode ser considerado um protocolo *RESTful* [5].

A comunicação com o CoAP consiste na troca de mensagens compactas transportadas sobre o UDP, sendo que, há uma camada DTLS (*Datagram Transport Layer Security*) entre as mensagens e o UDP, usada para prover segurança aos dados [5].

As mensagens são codificadas em formato binário com um cabeçalho fixo de 4 *bytes*, seguido de um *token* com largura variável (de 0 a 8 *bytes*). Após o *token* pode haver ou não uma série de opções do CoAP no formato *Type-Length-Value* (TLV). As opções TLV podem ser seguidas por um *payload*, preenchendo o resto o do datagrama [5]. O formato da mensagem é ilustrado na Figura 9.

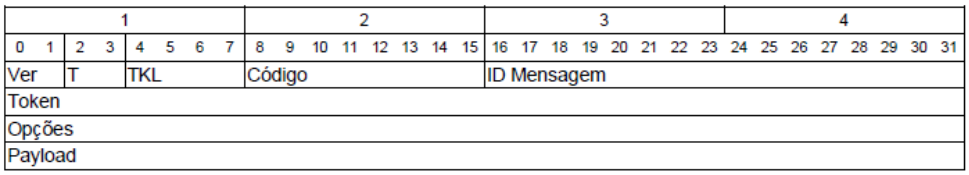

Figura 9 – Datagrama CoAP [5].

Os campos que constituem o cabeçalho do datagrama são definidos pelo RFC 7252 [5] como:

- **Versão (Ver)**: Inteiro não assinado de dois *bits* que indica o número da correspondente versão do CoAP;
- **Tipo (T)**: Inteiro não assinado de dois *bits* que indica se a mensagem é do tipo *Confirmable* (0), *Non-confirmable* (1), *Acknowledgement* (2) ou *Reset* (3);
- **Tamanho do *token* (TKL)**: Inteiro não assinado de quatro *bits* que indica o tamanho variável do *token* para até 8 *bytes*. Tamanhos de 9 a 15 *bytes* são reservados e não devem ser enviados;
- **Código**: Inteiro não assinado de 8 *bits* que caracteriza o tipo da mensagem, que pode ser um método de requisição (Tabela 1) ou um código de resposta (Tabela 2)

Tabela 1 – Códigos dos métodos CoAP [5]

Código	Nome
0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE

O cabeçalho é seguido de um *token* utilizado para relacionar requisições e respostas. Ele é gerado pelo cliente e transportado junto da requisição. O *token* deve então ser transmitido na resposta correspondente pelo servidor.

Tabela 2 – Códigos de resposta CoAP [5]

Código	**Nome**
2.01	Criado
2.02	Deletado
2.03	Válido
2.04	Alterado
2.05	Conteúdo
4.00	*Bad Request*
4.01	Não autorizado
4.02	*Bad Option*
4.03	Proibido
4.04	Não encontrado
4.05	Método não permitido
4.06	Inaceitável
4.12	Precondição falhou
4.13	Entidade de requisição muito grande
4.15	*Context-format* não suportado
5.00	Erro interno no servidor
5.01	Não implementado
5.02	*Bad Gateway*
5.03	Serviço indisponível
5.04	*Gateway timeout*
5.05	*Proxing* não suportado

2.2.3 MQTT

Criado em meados de 1999 por Andy Stanford-Clark (IBM) e Arlen Nipper (*Eurotech*), o MQTT foi padronizado em 2014 pela *Organization for the Advancement of Structured Information Standards* (OASIS) e encontra-se na versão 3.1.1 [6].

Com um cabeçalho fixo de apenas 2 *bytes* e utilizando a estratégia *publish/subscribe* (publicação/assinatura) para troca de mensagens, o MQTT é ideal para ambientes em IoT e demais contextos que exigem comunicação *Machine-to-Machine* (M2M), onde dispositivos com baixa capacidade computacional são inseridos em redes com latência, baixa largura de banda e alta taxa de perda de pacotes [6].

O MQTT foi projetado para ser fácil de implementar, sendo que, deve ser usado juntamente com os protocolos TCP/IP nas camadas de transporte e rede, respectivamente, ou sobre outros protocolos que forneçam ordenação, comunicação sem perdas e conexões bidirecionais [6]. Estas características incluem:

- Uso do padrão *publish/subscribe*, provendo distribuição de mensagem um-para-muitos e dissociação das aplicações (Figura 10);
- Transporte de mensagens não influenciado pelo conteúdo do *payload*.
- Entrega de mensagem com três níveis de *Quality of Service* (QoS): "*At most once*" (No máximo uma vez); "*At least once*" (Ao menos uma vez); e "*Exactly once*" (Exatamente uma vez) (Seção 2.2.4).
- Baixa sobrecarga de transporte e trocas de protocolo minimizadas, visando reduzir o tráfego na rede.

- Notificação às partes interessadas em caso de desconexões inesperadas.

Para realizar a comunicação sobre a estratégia *publish/subscribe*, o MQTT deve implementar um servidor próprio, chamado de *broker*, que serve como intermediador para troca de mensagens entre os dispositivos.

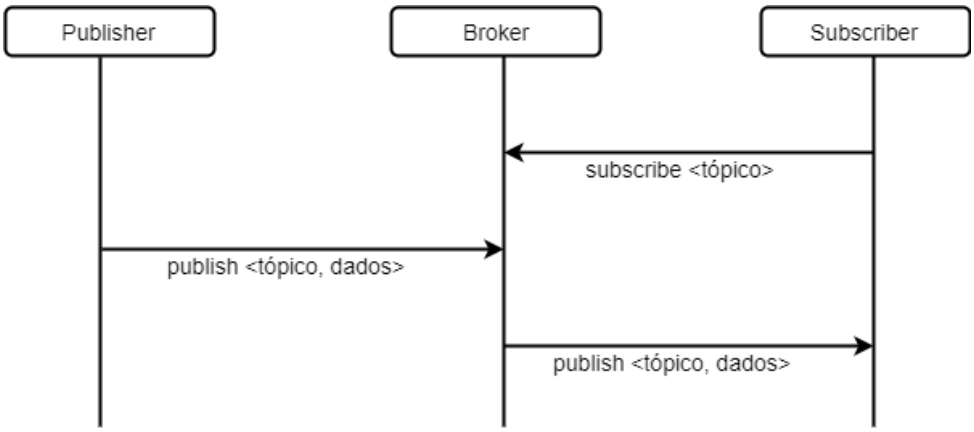

Figura 10 – Exemplo de comunicação MQTT sobre a estratégia *publish/subscribe*.

Fonte: Próprio autor.

Quando um dispositivo que atua como cliente deseja enviar uma mensagem para um ou mais clientes, o mesmo deve publicar um pacote de controle PUBLISH contendo os dados e um tópico. O servidor recebe o pacote PUBLISH, que é direcionado para os clientes previamente assinados no tópico correspondente. Os clientes que desejam receber as mensagens de um determinado tópico devem enviar um pacote SUBSCRIBE para o servidor com o tópico pretendido. Confirmada a assinatura, ele está apto a receber as mensagens publicadas no respectivo tópico, como ilustrado na Figura 10.

Os pacotes de controle dos tipos PUBLISH e SUBSCRIBE são apenas 2 dos 14 que podem ser usados durante a comunicação com o MQTT. Alguns podem ser usados em qualquer troca de mensagens, enquanto que outros dependem do nível de QoS das publicações ou de operações específicas para serem utilizados ou não.

A estrutura dos pacotes de controle, seus valores e tipos são descritas a seguir, e posteriormente, o comportamento operacional do protocolo é abordado em relação ao nível de QoS das publicações.

Pacotes de Controle

Um pacote de controle é um conjunto estruturado de *bits* que contêm alguma informação que é emitida por meio de uma conexão de rede. Sendo que, a conexão de rede deve ser providenciada por algum protocolo da camada de transporte.

Os pacotes MQTT podem ter até 3 elementos em sua estrutura, como exibido na Figura 11. O primeiro destes é o cabeçalho fixo, presente em todos os pacotes, o qual pode

ser seguido de um cabeçalho variável e um *Payload*, ambos presentes em alguns tipos de pacotes, mas não em todos.

Cabeçalho fixo
Cabeçalho variável
Payload

Figura 11 – Estrutura de um pacote MQTT [6].

O cabeçalho fixo possui 2 *bytes*. Os *bits* 7, 6, 5 e 4 do *byte* 1, são usados para definir qual tipo de pacote será transmitido, enquanto que os *bits* 3, 2, 1 e 0 são marcadores específicos para cada tipo de pacote, como ilustrado na Figura 12. O *byte* 2 corresponde ao campo de Largura Restante, que indica o número de *bytes* restantes no pacote, incluindo dados do cabeçalho variável e do *payload*.

Bit	7	6	5	4	3	2	1	0
Byte 1	Tipo de pacote MQTT				Marcadores específicos para cada tipo de pacote MQTT			
Byte 2	Largura restante							

Figura 12 – Formato do cabeçalho fixo MQTT [6].

Com exceção do pacote PUBLISH, o qual pode ter seus marcadores definidos de variadas formas, todos os outros tipos de pacotes têm seus marcadores fixos, definidos de forma reservada. Nos pacotes PUBREL, SUBSCRIBE e UNSUBSCRIBE, os *bits* 3, 2, 1 e 0 são fixados como 0, 0, 1 e 0, respectivamente, enquanto que nos outros (exceto PUBLISH), os 4 *bits* devem ter valor 0.

O pacote PUBLISH conta com 3 marcadores específicos que abrangem os *bits* 3, 2, 1 e 0 do cabeçalho fixo. Na Figura 13 são apresentados detalhes de seu formato, no qual o *bit* 3 representa o marcador DUP, seguido de um indicador de nível de QoS envolvendo os *bits* 2 e 1, e do sinalizador RETAIN, determinado pelo bit 0. A definição dos marcadores pode ser conferida nos tópicos abaixo:

- **DUP**: Quando tem valor 0, o marcador DUP indica que essa é a primeira vez que o Cliente ou Servidor tentam enviar o pacote PUBLISH referido. Se o valor for 1, então aponta que o pacote teve que ser reenviado após uma tentativa prévia.
- **QoS**: Define o nível de QoS para o envio do pacote, os *bits* podem assumir os seguintes valores: 0 e 0 (*At most once*); 0 e 1 (*At least once*); 1 e 0 (*Exactly once*); e 1 e 1, que é reservado e não deve ser usado.
- **RETAIN**: quando um cliente envia uma mensagem PUBLISH ao servidor com este marcador ativado (RETAIN = 1), ela deve ser retida no servidor mesmo depois de ser entregue aos assinantes. No evento de uma nova subscrição a um tópico, a última

mensagem retida para este tópico deve ser enviada para o novo assinante caso este marcador esteja ativado.

Bit	7	6	5	4	3	2	1	0
Byte 1	Tipo de pacote MQTT				DUP	Nível QoS		RETAIN
Byte 2	Largura restante							

Figura 13 – Formato do cabeçalho fixo de um pacote PUBLISH [6].

Após o cabeçalho fixo, o pacote MQTT pode conter um cabeçalho variável, cujo conteúdo varia dependendo do tipo de pacote. O conteúdo mais comum é o Identificador de Pacote, que consiste na alocação de 2 *bytes* para designar um valor único de identificação do pacote enquanto seu fluxo de entrega está em andamento.

O terceiro e último elemento de um pacote MQTT é o *payload*. Presente nos pacotes dos tipos CONNECT, PUBLISH (opcional), SUBSCRIBE, SUBACK e UNSUBSCRIBE, o conteúdo do *payload* varia entre cada tipo de pacote. Em um pacote PUBLISH, por exemplo, o *payload* contém a mensagem enviada pela aplicação.

Na Tabela 3 são citados todos os 14 tipos de pacotes de controle, seus valores decimais, a direção de seu fluxo entre o Cliente (C) e o Servidor (S), seguidos de uma breve descrição.

Tabela 3 – Pacotes de controle MQTT [6]

Nome	**Valor**	**Direção de fluxo**	**Descrição**
Reservado	0	Proibido	Reservado
CONNECT	1	C → S	Requisição do cliente para conexão com o Servidor
CONNACK	2	S → C	Confirmação de conexão. Resposta a um CONNECT
PUBLISH	3	C → S ou S → C	Publicar mensagem
PUBACK	4	C → S ou S → C	Confirmação de publicação. Resposta a um PUBLISH (QoS 1)
PUBREC	5	C → S ou S → C	Publicação recebida (entrega assegurada 1). Resposta a um PUBLISH (QoS 2)
PUBREL	6	C → S ou S → C	Publicação liberada (entrega assegurada 2). Resposta a um PUBREC (QoS 2)
PUBCOMP	7	C → S ou S → C	Publicação completa (entrega assegurada 3). Resposta a um PUBREL (QoS 2)
SUBSCRIBE	8	C → S	Requisição de assinatura de um Cliente
SUBACK	9	S → C	Confirmação da assinatura. Resposta a um SUBSCRIBE
UNSUBSCRIBE	10	C → S	Requisição para cancelar assinatura
UNSUBACK	11	S → C	Confirmação de cancelamento de assinatura. Resposta a um UNSUBSCRIBE
PINGREQ	12	C → S	Requisição de *PING*
PINGRESP	13	S → C	Resposta *PING*
DISCONNECT	14	C → S	Cliente está desconectando
Reservado	15	Proibido	Reservado

2.2.4 Níveis de *Quality of Service* (QoS) do MQTT

O MQTT realiza a entrega das mensagens de acordo com os níveis de QoS definidos. O protocolo de entrega é simétrico, nas descrições e exemplos mostrados nesta seção, o Cliente e o Servidor podem assumir os papéis tanto de expedidor quanto de receptor.

Além disso, a entrega de cada mensagem é concebida por um único expedidor e um único recebedor, pois quando o Servidor entrega uma mensagem para mais de um Cliente, cada Cliente é tratado independentemente.

Ademais, os diagramas de fluxo exibidos nesta seção são considerados exemplos de possíveis implementações.

QoS 0: *At most once*

Com a regra "*At most once*" (No máximo uma vez), a entrega da mensagem depende das capacidades da rede subjacente. Não há resposta por parte do recebedor e nem reenvio por parte do expedidor, logo, a mensagem é recebida uma ou nenhuma vez.

A declaração normativa MQTT-4.3.1-1 [6] define apenas uma ação que deve ser obrigatoriamente cumprida:

- O expedidor deve enviar um pacote PUBLISH com QoS=0 e DUP=0.

Em caso de entrega concluída, o recebedor aceita a posse da mensagem (Figura 14).

Ação do Cliente	Fluxo de pacotes	Ação do Servidor
Enviar pacote PUBLISH com QoS=0 e DUP=0		
	PUBLISH ----------→	
		Direcionar mensagem para o(s) destinatários(s)

Figura 14 – Exemplo de diagrama de fluxo para entrega com QoS 0 [6].

QoS 1: *At least once*

Este nível de QoS garante que a mensagem será entregue para o recebedor ao menos uma vez. Um pacote PUBLISH com QoS=1 possui um Identificador de Pacote em seu cabeçalho variável e sua entrega é confirmada com um pacote PUBACK.

Segundo a declaração normativa MQTT-4.3.2-1 [6], na entrega com QoS 1, o expedidor deve:

- Atribuir um Identificador de Pacote não usado sempre que haver uma nova mensagem a ser publicada;
- Enviar um pacote PUBLISH com QoS=1 e DUP=0 contendo este Identificador de Pacote;

- Tratar o pacote PUBLISH como “não confirmado” até que tenha recebido o correspondente pacote PUBACK do recebedor.

O Identificador de Pacote torna-se disponível para uso novamente logo após o expedidor receber o pacote PUBACK.

Enquanto isso, a afirmação MQTT-4.3.2-2 [6] define que o recebedor deve:

- Responder com um pacote PUBLISH contendo o Identificador de Pacote do pacote PUBLISH correspondente, aceitando assim a posse da mensagem;
- Assim que enviar um pacote PUBACK, qualquer pacote PUBLISH recebido que possua o mesmo Identificador de Pacote deve ser tratado como uma nova publicação, independentemente do valor do marcador DUP.

Ação do Cliente	Fluxo de pacotes	Ação do Servidor
Armazenar mensagem		
Enviar pacote PUBLISH com QoS=1, DUP=0 e <Identificador do Pacote>	PUBLISH ----------→	
		Iniciar direcionamento da entrega para o(s) destinatário(s)
		Responder com um pacote PUBACK <Identificador do Pacote>
	PUBACK ←----------	
Descartar mensagem		

Figura 15 – Exemplo de diagrama de fluxo para entrega com QoS 1 [6].

QoS 2: *Exactly once*

Este é o último nível de QoS, ideal para ser usado quando nenhuma perda ou duplicação de mensagens deve ser aceita. Assim como no QoS 1, suas mensagens também devem possuir um Identificador de Pacote atrelado ao seu cabeçalho variável.

A especificação do protocolo define um processo de duas etapas para confirmação de recepção das mensagens PUBLISH. Desta forma, o MQTT-4.3.3-1 [6] determina que o expedidor deve:

- Atribuir um Identificador de Pacote não usado quando houver uma nova mensagem a ser publicada;
- Enviar um pacote PUBLISH com QoS=2 e DUP=0 contendo este Identificador de Pacote;
- Tratar o pacote PUBLISH como “não confirmado” até que tenha recebido o respectivo pacote PUBREL do recebedor;

- Enviar um pacote PUBREL assim que receber o pacote PUBREC do recebedor. Sendo que, este pacote PUBREL deve conter o mesmo Identificador de Pacote do pacote PUBLISH original;
- Tratar o pacote PUBREL como "não confirmado" até que receba o pacote PUBCOMP correspondente do recebedor;
- Não deve reenviar o pacote PUBLISH, uma vez que seu respectivo pacote PUBREL tenha sido enviado.

O Identificador de Pacote torna-se disponível para reuso logo após o expedidor receber o pacote PUBCOMP.

Ademais, o MQTT-4.3.3-2 [6] define que o recebedor deve:

- Responder com um pacote PUBREC contendo o Identificador de Pacote do pacote PUBLISH recebido, aceitando assim a posse da mensagem;
- Enquanto não receber o pacote PUBREL correspondente, o recebedor deve confirmar qualquer pacote PUBLISH subsequente que tenha o mesmo Identificador de Pacote por meio do envio de um PUBREC. Nesse caso, o método não pode causar duplicação de mensagens;
- Responder um pacote PUBREL por meio do envio de um pacote PUBCOMP contendo o mesmo Identificador de Pacote;
- Depois que enviar um PUBCOMP, o recebedor deve tratar qualquer pacote PUBLISH recebido que tenha o mesmo Identificador de Pacote como uma nova publicação.

Ação do Cliente	Fluxo de pacotes	Ação do Servidor
Armazenar mensagem		
Enviar pacote PUBLISH com QoS=2, DUP=0 e <Identificador do Pacote>		
	PUBLISH ----------→	
		Método A: Armazenar mensagem ou Método B: Armazenar o <Identificador do Pacote> e iniciar o direcionamento da mensagem
		Responder com um pacote PUBREC <Identificador do Pacote>
	PUBREC ←----------	
Discartar mensagem original e armazenar o pacote PUBREC recebido com o <Identificador do Pacote>		
Enviar um pacote PUBREL <Identificador do Pacote>		
	PUBREL ----------→	
		Método A: Direcionar a mensagem para o(s) destinatário(s) e descartá-la em seguida ou Método B: Descartar <Identificador do Pacote>
		Enviar pacote PUBCOMP <Identificador do Pacote>
	PUBCOMP ←----------	
Descartar estado armazenado		
Observação: O servidor não é necessário para realizar a entrega da mensagem enquanto não enviar o pacote PUBREC ou PUBCOMP. Quando o expedidor original da mensagem recebe o pacote PUBREC, a posse da mensagem é transferida para o Servidor.		

Figura 16 – Exemplo de diagrama de fluxo para entrega com QoS 2 [6].

No caso ilustrado na Figura 16, a escolha dos métodos A ou B por parte do recebedor fica a critério da implementação. Desde que seja escolhido apenas um dos métodos, as garantias presentes no QoS 2 não serão afetadas.

2.2.5 Comparação com outros protocolos

Diversos protocolo de aplicação são propostos para soluções em IoT, cada um com seus aspectos específicos para prover comunicação.

A escolha do protocolo deve levar em conta fatores como: capacidade de processamento e armazenamento dos dispositivos; limitações do ambiente; e tipo de comunicação (*Machine-to-Machine*, *Machine-to-Server* ou *Server-to-Server*) [34].

Além dos protocolos HTTP, CoAP e MQTT apresentados anteriormente, na Fi-

gura 17 são citados também outros protocolos de destaque no cenário de IoT e suas principais características.

Protocolo de aplicação	RESTful	Transporte	Arquitetura publicação/ Assinatura	Requisição/ Resposta	Segurança	QoS	Tamanho do cabeçalho fixo em *bytes*	Padronizado por	Aplicação
HTTP	Sim	TCP	Não	Sim	SSL	Não	-	IETF e W3C	Sistemas de informação Cliente-Servidor
CoAP	Sim	UDP	Não	Sim	DTLS	Sim	4	IETF	Redes sociais e redes móveis
MQTT	Não	TCP	Sim	Não	SSL	Sim	2	OASIS	*Facebook messenger*, cuidados médicos, monitoramento, aferição de energia
MQTT-SN	Não	TCP	Sim	Não	SSL	Sim	2	Clark, A. S e Truong, H. L.	Monitoramento remoto, redes de sensores
XMPP	Não	TCP	Sim	Sim	SSL	Não	-	Padrão aberto	*Chats* multilaterais, chamadas de voz e vídeo
AMQP	Não	TCP	Sim	Não	SSL	Sim	8	OASIS	Ambientes orientados a mensagens
DDS	Não	TCP UDP	Sim	Não	SSL DTLS	Sim	-	OMG	*Multcasting*

Figura 17 – Características e propriedades dos protocolos de aplicação em IoT [2], adaptado pelo autor.

O protocolo MQTT-SN é uma adaptação do MQTT voltada para redes de sensores, pois é mais leve e exige ainda menos recursos em relação ao MQTT [35].

Utilizando uma arquitetura baseada em um *broker*, o DDS (*Data Distribution Service*) [36] é semelhante ao MQTT. Porém, o DDS mostra-se mais adequado para serviços. Uma das principais diferenças está no suporte para *Qualities os Service*, enquanto o MQTT detêm 3 níveis de QoS, o DDS opera com 23, além do suporte a implementação com UDP e DTLS.

Portando aplicabilidades distintas, o AMQP [37] e o XMPP [38], mostram-se ideais para ambientes orientados a mensagens e *chats* multilaterais de mensagem, voz e vídeo, respectivamente.

Ao analisar as particularidades de cada protocolo, percebe-se o caráter mais abrangente do CoAP e do MQTT, enquanto que os demais são projetados para ambientes mais específicos, o que não impede que uma aplicação complexa possa empregar diferentes protocolos em diferentes contextos.

2.3 Contratos

Desde os princípios da sociedade moderna, os compromissos, normas, regras e padrões regem o relacionamento entre as pessoas [8]. A origem etimológica do vocábulo contrato conduz ao vínculo jurídico das vontades com vistas a um objeto específico, sendo um acordo de vontades criador de direitos e obrigações entre as partes envolvidas [39].

Um contrato reflete a vontade entre dois ou mais indivíduos no intuito de adquirir, resguardar, modificar, transferir ou extinguir direitos, garantindo a vontade dos mesmos sobre um determinado assunto [8, 39].

Segundo GOMES[40], contrato é o acordo jurídico bilateral, ou plurilateral, que atribui as partes a responsabilidade de observância de conduta idônea à satisfação dos interesses que regularam.

Os contratos podem ser classificados conforme a atribuição de responsabilidades. Contratos unilaterais são aqueles no qual um indivíduo assume responsabilidades, enquanto que nos contratos bilaterais, dois indivíduos assumem responsabilidades. Já em contratos multilaterais, três ou mais indivíduos assumem responsabilidades [39].

A negociação de contratos entre os indivíduos pode resultar na combinação de cláusulas de interesse de cada um. Essa combinação pode acarretar em diversos problemas, tal como os conflitos normativos [41], em que ocorrem uma ou mais contradições entre as normas [42]. Um conflito faz com que o contrato torne-se inválido, visto que não há como satisfazê-lo sem violar ou desconsiderar alguma das normas estabelecidas [41].

A quebra de um contrato ocorre mediante a violação de alguma cláusula por parte de um ou mais indivíduos. Posto isto, pode ser interessante para as partes envolvidas no contrato, ter conhecimento de quais indivíduos estão relacionados direta ou indiretamente com a violação. Portanto, no processo de monitoramento da execução de um contrato as partes envolvidas em cada ação a ser tomada devem ser levadas em conta, possibilitando a identificação dos envolvidos numa quebra contratual.

2.3.1 Contratos eletrônicos

Um contrato eletrônico ou *"e-Contract"* é um meio para representação digital dos contratos convencionais, utilizados para firmar acordos entre pessoas ou entidades. Desta forma, a tecnologia é utilizada como aliada na solução de problemas encontrados neste contexto. Assim como em contratos convencionais, um contrato eletrônico engloba as obrigações de cada participante, atividades exercidas e responsabilidades definidas para cada um dos indivíduos para satisfazer os termos e condições do contrato estabelecido [43].

São diversas as áreas de aplicação dos contratos eletrônicos. Em meios legais, contratos físicos são convertidos ou substituídos para o meio eletrônico. Técnicas de contra-

tação eletrônica em sistemas, lógicas de negócio ou integração de serviços também podem ser utilizadas. A contratação eletrônica legal visa a criação de um documento contratual que busca expressar a intenção das partes envolvidas [43]. As ferramentas para contratação eletrônica buscam informar as partes acerca dos efeitos das disposições contratuais, auxiliar na verificação e execução do contrato, assim como monitorar sua execução [8].

A lei considera contratos como um conjunto de obrigações e o mesmo se aplica ao meio eletrônico. O contrato eletrônico tem como aliado o apoio computacional e automatizado, que permite verificar, controlar e monitorar de forma mais eficiente [8].

Topologia dos contratos eletrônicos

A topologia de um contrato eletrônico pode ser definida de acordo com o número de participantes envolvidos, bem como a complexidade das relações de direitos e deveres entre os mesmos [44].

Um contrato bilateral tem como característica principal o acordo entre no máximo dois indivíduos, que estabelecem um contrato com obrigações de ambos. O contrato em cadeia inclui a participação de vários indivíduos, porém, em forma de sub-contratos, dos quais vários acordos bilaterais são estabelecidos. Já no contrato multilateral, os vários indivíduos envolvidos podem interagir entre si, de forma a causar dependências mais complexas do que no modelo em cadeia, pois um indivíduo pode ter cláusulas e dependências que envolvem responsabilidades por parte de outros indivíduos [8].

Contratos complexos podem ser divididos em sub-contratos bilaterais para facilitar sua execução. Porém, essa divisão não pode ser realizada sem a perda de informações importantes em contratos multilaterais [27]. Portanto, um conjunto de contratos bilaterais não é capaz de expressar a complexidade do relacionamento entre os indivíduos de um contrato multilateral, visto que a perda de informações e a diminuição da expressividade das normas do contrato é um fator iminente [8].

2.3.2 Verificação de contratos eletrônicos

Variadas pesquisas já foram realizadas no âmbito dos contratos eletrônicos, especialmente na verificação dos mesmos. Além de acelerar sua escrita, a automatização de um contrato permite que suas propriedades sejam formalmente verificadas [26]. Algumas das técnicas e abordagens mais comuns em contratos eletrônicos são:

- **Negociação:** Cenário em que pelo menos dois indivíduos propõem soluções e contrapostas de seu interesse, visando alcançar um acordo aceitável por ambas as partes envolvidas. Este processo é realizado em contratos bilaterais e multilaterais e pode envolver ou não o uso de mediadores [45].

- **Detecção de conflitos:** A detecção de conflitos [43] visa a detecção e eliminação de conflitos normativos de um contrato, pois esta situação invalida um contrato, induzindo a uma violação [42].

- **Assinatura:** De forma geral, a assinatura de um contrato digital gera mais complicações do que no modo convencional. Isso deve-se ao fato de que nenhum indivíduo quer ser o primeiro a assinar o contrato, pois um indivíduo pode recusar-se a assiná-lo após obter a assinatura e o contrato do primeiro indivíduo, o que pode causar uma vantagem indesejada para um dos envolvidos [46, 47].

- **Monitoramento:** Durante a execução do contrato, o monitoramento tem a função de verificar se as cláusulas são respeitadas pelos participantes. Um contrato é violado quando alguma cláusula é ignorada por um dos participantes. Tornando as transações e relacionamentos entre os indivíduos mais confiáveis, flexíveis, eficientes e aceitáveis, é possível manter os benefícios entre todos os indivíduos na presença de violações [43]. Sendo assim, um sistema de monitoramento de contratos precisa saber quais ações são aceitáveis ou esperadas de um indivíduo, assim como identificar o responsável por uma violação, caso ocorra [43].

2.4 Representação formal de sistemas

Com a utilização de formalismos para expressar contratos alguns problemas normalmente encontrados em contratos convencionais são evitados, como inconsistências e ambiguidades. Portanto, a definição de um formalismo adequado faz parte do processo de verificação de contratos eletrônicos [26]. A seguir são descritas as lógicas proposicional e temporal, normalmente utilizadas para representar sistemas e propriedades.

2.4.1 Lógica proposicional

Geralmente, qualquer cálculo matemático é criado com a intenção de representar um certo domínio de objetos formais, normalmente com o objetivo de facilitar as computações e inferências que precisam ser realizadas sobre esta representação [48]. Em lógica e matemática, uma lógica proposicional é um sistema formal em que fórmulas representam proposições formadas por proposições atômicas usando conectivos lógicos e um sistema de regras de derivação, desta forma, as fórmulas podem ser consideradas como teoremas do sistema formal [48].

A lógica proposicional tem como objetivo modelar o raciocínio humano, partindo de proposições, como frases declarativas [48]. Por exemplo, considerando-se a frase “1 mais 1 é igual a 10”, ou simbolicamente, “1 + 1 = 10”. Esta frase pode ser considerada uma asserção declarativa, visto que afirma ou nega um fato, portanto representa uma proposição com valor verdadeiro ou falso. Neste caso, a proposição é verdadeira, num

sistema numérico de base 2, ou falsa, caso o sistema seja decimal. Portanto, a lógica proposicional estuda formas de raciocínio sobre afirmações que podem ser verdadeiras ou falsas, além de definir meios para construção de demonstrações que provem que uma determinada conclusão é verdadeira num certo contexto, levando em conta um conjunto de hipóteses [48].

A sintaxe do cálculo proposicional é definida por um alfabeto constituído por [49]:

- **Símbolos de pontuação:** (,);
- **Símbolos de verdade:** $true$ ($\top$), $false$ ($\bot$);
- **Símbolos proposicionais:** $P,\ Q,\ R,\ S,\ P_1,\ Q_1,\ R_1,\ S_1,\ ...$;
- **Conectivos proposicionais:** $\neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow$.

Para um melhor entendimento, na Tabela 4 são apresentas algumas formas de leitura para linguagem natural atribuídas aos conectivos proposicionais.

Tabela 4 – Leituras dos conectivos proposicionais para a linguagem natural [13].

Conectivo	**Exemplo**	**Leitura**
Negação $\neg$	$\neg A$	Não A; Não se dá que A; Não é verdade que A.
Conjunção $\wedge$	$A \wedge B$	A e B; A, mas B; A, embora B; A, assim como B.
Disjunção $\vee$	$A \vee B$	A ou B ou ambos.
Implicação $\rightarrow$	$A \rightarrow B$	se A, então B; se A, isto significa que B; tendo-se A, então B; A implica B; B é implicada por A.
Bi-implicação $\leftrightarrow$	$A \leftrightarrow B$	A se e somente se B; A quando e somente quando B; A equivale a B.

Um cálculo proposicional é um sistema formal $\mathcal{L} = (\mathcal{A},\ \Omega,\ \mathcal{Z},\ \mathcal{I})$, em que $\mathcal{A}$ é um conjunto finito de símbolos proposicionais. Conhecidos também como fórmulas atômicas ou elementos terminais, os elementos de $\mathcal{A}$ são os mais básicos da linguagem formal $\mathcal{L}$, sintaticamente falando [48]. No decorrer desta sessão, os elementos de $\mathcal{A}$ serão denotados como $P,\ Q,\ R$, em diante.

O conjunto Ω compreende a coleção de símbolos de verdade e dos conectivos proposicionais, ou conectivos lógicos, e é dividido entre conjuntos distintos, de forma que $\Omega = \Omega_0 \cup \Omega_1 \cup ... \cup \Omega_j \cup ... \cup \Omega_m$. Nesta divisão, Ω_j é o conjunto dos símbolos de aridade j [48].

A linguagem, ou conjunto de fórmulas $\mathcal{L}$, é definida recursiva ou indutivamente pelas seguintes regras [13, 48]:

- Qualquer elemento do conjunto $\mathcal{A}$ é uma fórmula de $\mathcal{L}$;
- Se P é uma fórmula, então $\neg P$, a negação de P, é uma fórmula;
- Se P e Q são fórmulas, então a disjunção dada por $P \vee Q$, é uma fórmula;
- Se P e Q são fórmulas, então a conjunção dada por $P \wedge Q$, é uma fórmula;
- Se P e Q são fórmulas, então a implicação dada por $P \rightarrow Q$, é uma fórmula, tal que P é o antecedente, e Q, o consequente.
- Se P e Q são fórmulas, então a bi-implicação dada por $P \leftrightarrow Q$, é uma fórmula.

O conjunto $\mathcal{Z}$ é a coleção finita de regras de inferência [48]. Já o conjunto $\mathcal{I}$ é a coleção finita de pontos iniciais, também chamados de axiomas [48].

As propriedades semânticas da Lógica Proposicional podem ser determinadas por meio uma Tabela-verdade, em que cada linha, exceto a primeira, representa uma valoração para cada sub-fórmula de uma dada fórmula em relação aos possíveis valores de suas proposições. Seguindo o princípio da bivalência, os valores para cada átomo da tabela possui favor verdadeiro (V) ou falso (F) [48].

Na Tabela 5 são demonstradas as propriedades dos conectivos de negação, conjunção, disjunção, implicação e bi-implicação, respectivamente. O operador de negação pode ser adicionado indefinidamente, mesmo que não haja necessidade, visto que $\neg\neg P \equiv P$ e $\neg\neg\neg P \equiv \neg P$. Nota-se que a conjunção entre duas fórmulas só é verdadeira quando ambas são verdadeiras, enquanto que, a disjunção entre duas fórmulas só é verdadeira quando ao menos uma delas é verdadeira. A implicação entre duas fórmulas só é falsa se a da esquerda (antecedente) for verdadeira e a da direita (consequente) for falsa. Sendo assim, das propriedades descritas, a implicação é a única não comutativa, ou seja, a ordem das proposições envolvidas alteram seu resultado. Por fim, a bi-implicação, ou equivalência, entre duas fórmulas é verdadeira quando ambas são verdadeiras ou ambas são falsas.

Tabela 5 – Tabela verdade dos conectivos proposicionais.

Proposições		**Conectivos proposicionais**										
		Negação $\neg$			**Conjunção** $\wedge$		**Disjunção** $\vee$		**Implicação** $\rightarrow$		**Bi-implicação** $\leftrightarrow$	
P	**Q**	$\neg P$	$\neg\neg P$	$\neg\neg\neg P$	$P \wedge Q$	$Q \wedge P$	$P \vee Q$	$Q \vee P$	$P \rightarrow Q$	$Q \rightarrow P$	$P \leftrightarrow Q$	$Q \leftrightarrow P$
V	V	F	V	F	V	V	V	V	V	V	V	V
V	F	F	V	F	F	F	V	V	F	V	F	F
F	V	V	F	V	F	F	V	V	V	F	F	F
F	F	V	F	V	F	F	F	F	V	V	V	V

A construção de uma Tabela-verdade consiste em dois passos: primeiro é definida uma linha em que estão contidas as proposições presentes numa dada fórmula, assim como as sub-fórmulas, seguidas da própria fórmula; no segundo passo são preenchidas as linhas l, em que estão todos os possíveis valores que as proposições atômicas podem receber e os valores recebidos pelas fórmula a partir dos valores das proposições. O número de linhas

é $l = n^t$, sendo n o número de valores que o sistema permite (verdadeiro ou falso) e t, o número de proposições atômicas que a fórmula possui.

Dada a fórmula $((P \vee Q) \rightarrow (Q \wedge (Q \leftrightarrow P)))$, tem-se o conjunto de sub-fórmulas $(Q \wedge (Q \leftrightarrow P))$, $(Q \leftrightarrow P)$, $(P \vee Q)$, P e Q. Como há apenas duas proposições atômicas, P e Q, então $t = 2$, logo, $l = 2^2 = 4$. Baseado nisso, obtêm-se a valoração completa, exibida na Tabela 6.

Outra forma de se observar as propriedades semânticas das fórmulas proposicionais é por meio da construção de uma árvore de derivação, na qual a fórmula é derivada em suas sub-fórmulas repetidamente até que as "folhas" da árvores contenham apenas as proposições atômicas, como ilustrado na Figura 18.

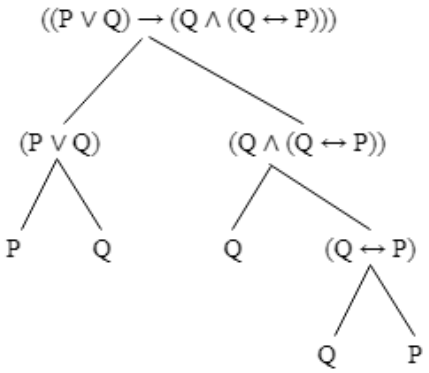

Figura 18 – Árvore de derivação associada à fórmula $((P \vee Q) \rightarrow (Q \wedge (Q \leftrightarrow P)))$.

Tabela 6 – Tabela verdade associada à fórmula $((P \vee Q) \rightarrow (Q \wedge (Q \leftrightarrow P)))$.

P	**Q**	$(P \vee Q)$	$(Q \leftrightarrow P)$	$(Q \wedge (Q \leftrightarrow P))$	$((P \vee B) \rightarrow (Q \wedge (Q \leftrightarrow P)))$
V	V	V	V	V	V
V	F	V	F	F	F
F	V	V	F	F	F
F	F	F	V	F	V

A lógica proposicional é considerada um dos tipos mais simples de cálculo lógico, podendo ser estendida de diversas formas. Com a adição de novas regras e definições, outras lógicas podem ser desenvolvidas, como, por exemplo, a lógica de primeira ordem [50], as lógicas modais [51], e a lógica temporal [52].

2.4.2 Lógica temporal

A lógica temporal é um formalismo usado para descrever sequências de transações entre estados em sistemas reativos [53]. Portanto, uma fórmula lógica temporal geralmente é avaliada sobre um sistema de transição que modela uma especificação [53], isto é, um modelo de *Kripke* [54].

Na lógica temporal, o modelo é dado pela tupla $\mathcal{M} = (S, \rightarrow, L)$, em que S é o conjunto de estados, $\rightarrow$ (ou R) $\in S \times S$ é a relação de transição e $L\colon S \rightarrow 2^{AP}$ denota a

função de rotulação dos estados com as proposições atômicas AP verdadeiras no estado em que se encontram.

Um exemplo de sistema de transição é apresentado na Figura 19, definido por:

- $AP = \{p, q, r\}$
- $S = \{s_0, s_1, s_2\}$
- $\rightarrow = \{(s_0, s_1), (s_1, s_0), (s_0, s_2), (s_1, s_2), (s_2, s_2)\}$
- $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$

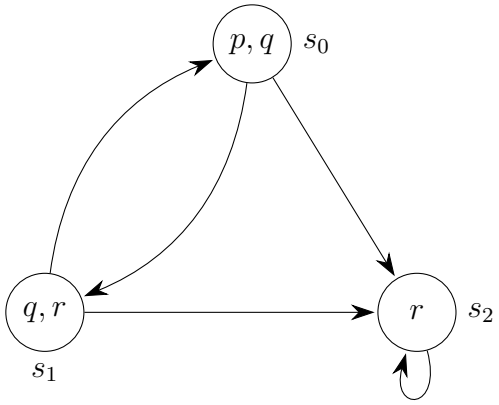

Figura 19 – Exemplo de um sistema de transição [7].

Uma fórmula lógica temporal pode ser verdadeira ou falsa dependendo do modelo no qual é interpretada. O conjunto de estados representam a evolução do sistema ao longo do tempo e as fórmulas são avaliadas em cada estado do sistema. Portanto, a noção estática de verdade é substituída pela noção dinâmica de evolução dos estados do sistema ao longo do tempo [53].

A lógica temporal pode ser dos tipos linear ou ramificada. Na abordagem linear, o tempo é tratado como se cada momento tivesse apenas um único futuro possível, logo, as fórmulas são interpretadas como sequências lineares, ou caminhos. Já na abordagem ramificada, cada estado pode evoluir para uma ramificação de vários futuros possíveis, de acordo com a evolução do sistema, com uma representação em árvore [8].

Lógica temporal linear

Como dito acima, uma lógica temporal compreende a evolução de um sistema de transição de estados ao longo do tempo. Diante disso, a lógica temporal linear (ou LTL) possui conectivos que se referem ao futuro, e suas fórmulas são interpretadas sobre uma sequência de estados chamada de *path* (caminho). Um *path* é uma sequência finita não vazia denotada por $\pi = \langle \pi_0, \pi_1, \ldots, \pi_{n-1} \rangle$, em que os estados $\pi_0, \pi_1, \ldots, \pi_{n-1} \in S$ e

$(\pi_i, \pi_{i-1}) \in R$ para todo $0 \leq i < n-1$. O tamanho de um *path* é denotado por $|\pi| = \infty$. Um *path* infinito é denotado por $\pi = \langle \pi_0, \pi_1, \pi_3, \ldots \rangle$ com tamanho $|\pi| = \infty$. Todo *path* que não pode ser estendido é considerado maximal, logo, qualquer *path* infinito é maximal [55].

A sintaxe de uma fórmula LTL é composta de proposições atômicas e gerada conforme segue:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi U \varphi' \tag{2.1}$$

Assim como na lógica proposicional, os valores lógicos avaliados como verdadeiro e falso são representados pelos símbolos $\top$ e $\bot$, respectivamente, além dos operadores lógicos $\neg$, $\wedge$, $\vee$ e $\rightarrow$, que também têm o mesmo significado. Já o símbolo p representa um identificador das proposições, e os operadores X, F, G e U, são usados para especificar situações temporais de uma fórmula. A expressão $X\varphi$ (após φ) indica que a proposição φ é verdadeira no próximo estado. Já a expressão $F\varphi$ (afinal, φ) configura que em algum estado futuro da computação, φ é verdadeira, enquanto que $G\varphi$ (sempre φ) indica que φ é verdadeira em todos os estados do caminho de computação. Enfim, a expressão $\varphi U \varphi'$ (φ até que φ') significa que a proposição φ deve ser verdadeira em todos os estados até que φ seja verdadeira [8, 55].

Uma fórmula α em LTL é satisfeita se existe um modelo M e um *path* π tal que $\mathcal{M}, \pi \models \varphi$, ou seja, um modelo no qual o *path* π satisfaz a fórmula φ. Um *path* é formado por uma sequência de estados s e suas posições são indexadas partindo de π_1 até π_n, em que n determina a quantidade de posições de π. Sendo π um dado *path* de $\mathcal{M}$, as relações de satisfação $\models$ e $\not\models$ indicam se uma fórmula LTL é satisfeita ou não, respectivamente, por π [7], como exposto a seguir na Figura 20.

$$\pi \models \top \tag{2.2}$$
$$\pi \not\models \bot \tag{2.3}$$
$$\pi \models p \iff p \in L(\pi_1) \tag{2.4}$$
$$\pi \models \neg\varphi \iff \pi \not\models \varphi \tag{2.5}$$
$$\pi \models \varphi \wedge \varphi' \iff \pi \models \varphi \text{ e } \pi \models \varphi' \tag{2.6}$$
$$\pi \models \varphi \vee \varphi' \iff \pi \models \varphi \text{ ou } \pi \models \varphi' \tag{2.7}$$
$$\pi \models \varphi \rightarrow \varphi' \iff \pi \models \varphi \text{ sempre que } \pi \models \varphi' \tag{2.8}$$
$$\pi \models X\varphi \iff \pi_2 \models \varphi \tag{2.9}$$
$$\pi \models G\varphi \iff \forall i \geq 1 \mid \pi_i \models \varphi \tag{2.10}$$
$$\pi \models F\varphi \iff \exists i \geq 1 \mid \pi_i \models \varphi \tag{2.11}$$
$$\pi \models \varphi U \varphi' \iff \exists i \geq 1 \mid \pi_i \models \varphi' \text{ e } \forall j = 1 \wedge j < i \mid \pi_j \models \varphi \tag{2.12}$$

Figura 20 – Semântica da LTL [7].

Nas definições 2.2 e 2.3 da Figura 20 são indicadas as proposições verdadeiras e falsas, respectivamente. Na definição 2.4 é dito que a proposição p só é válida se estiver no primeiro estado do *path*. Nas proposições 2.5 a 2.8 são definidos os conectivos proposicionais da lógica proposicional. Em 2.9, o operador X denota que a fórmula φ é satisfeita no próximo estado, no caso, a segunda posição do *path*. As definições 2.10 e 2.11 representam os operadores G e F que apontam, respectivamente, que a fórmula φ é verdadeira em todas ou em pelo menos uma posição de π. Por fim, a definição 2.12 representa o operador U, expressando que, para toda posição antecessora à π_i, a fórmula φ é verdadeira, caso contrário, a proposição φ' é verdadeira.

Diversas propriedades relevantes podem ser verificadas com fórmulas LTL, tais como, de segurança, no qual o sistema é livre de *deadlocks*, e de progresso, em que toda requisição realizada é atendida [7, 8]. Huth e Ryan[7] descrevem algumas propriedades que podem ser representadas em sistemas reais, considerando um sistema especificado com as proposições *ocupado*, *requisitado*, *iniciado*, *pronto* e *habilitado*, como nos exemplos a seguir:

- o sistema não pode ter *iniciado* e ainda não estar *pronto* para atender requisições:

$$G\neg(iniciado \wedge \neg\, pronto) \tag{2.13}$$

- em todo sistema, se é *requisitado* algum recurso, o requisitante deve ser *respondido*:

$$G(requisitado \rightarrow F\ respondido) \tag{2.14}$$

- um processo é *habilitado* infinitas vezes em todo o caminho de computação:

$$GF\ habilitado \tag{2.15}$$

- um elevador subindo para o segundo andar não muda sua direção quando houver passageiros que desejam ir para o quinto andar:

$$G(andar2 \wedge subindo \wedge pressionadoBotao5 \rightarrow (subindoUandar5)) \tag{2.16}$$

Os exemplos acima demonstram que a LTL é capaz de representar propriedades essenciais em sistemas de transição de estados. Porém, há propriedades que não podem ser expressadas em LTL, como as citadas abaixo:

- Partindo de algum estado, é possível chegar em um estado dentre todos os caminhos possíveis que satisfaça a reinicialização do sistema?
- A partir do estado que indica que o elevador está no terceiro andar parado e com as portas fechadas, é possível chegar num estado onde o elevador continue parado?

A LTL não é capaz de expressar os exemplos acima, pois não é possível afirmar a existência de outros caminhos possíveis, ou seja, de outros *paths* [7].

Lógica temporal ramificada

A lógica temporal ramificada, ou lógica de computação em árvore (CTL), é capaz de representar sistemas de transições de estados em que o futuro não é determinado, ou seja, quando há uma variedade de possíveis caminhos que podem ser tomados a partir de um determinado estado. Sendo assim, a evolução do sistema ao longo do tempo pode ser representada por uma árvore, que representa todas as execuções possíveis [7].

Na Figura 21, à esquerda, é representado um modelo, e à direita, a árvore de computação e seus nós, representando o estado corrente e o nível de profundidade da execução, indicando as mudanças de estados.

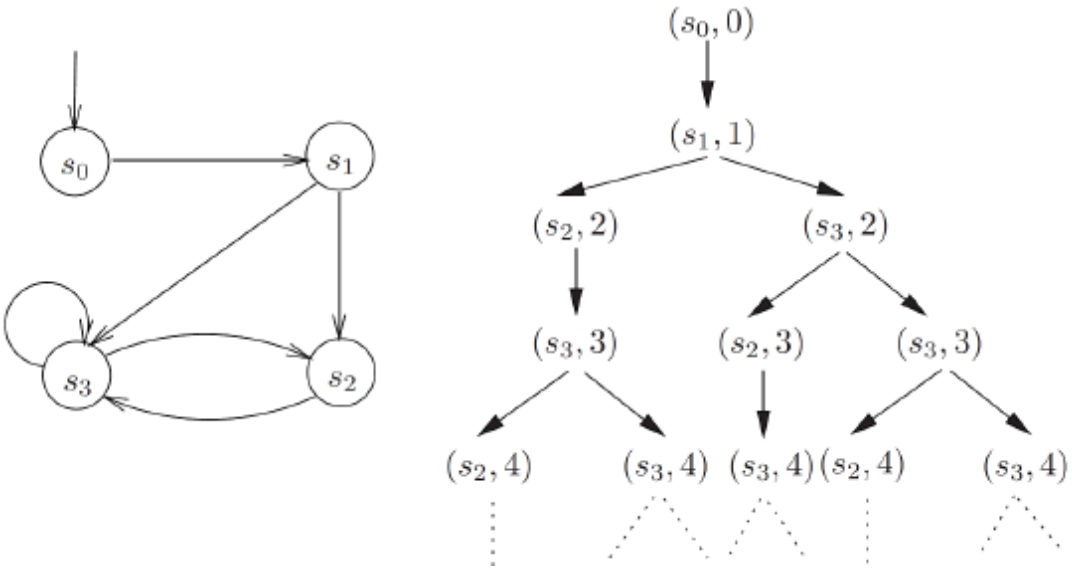

Figura 21 – Exemplo de árvore de computação num sistema de transição [7].

A sintaxe de construção de fórmulas CTL estende a LTL e adiciona os operadores existencial (E) e universal (A), e consiste em:

$$\begin{aligned} &\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \\ &AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U \varphi] \mid E[\varphi U \varphi] \end{aligned} \tag{2.17}$$

Os conectivos temporais da CTL são usados em pares. O primeiro conectivo da fórmula é um quantificador, denotado por A (inevitavelmente) ou E (provavelmente), enquanto que o segundo é usado da mesma forma que na LTL. Os operadores temporais E e A expressam propriedades para algum ou todos os caminhos de computação que se iniciam em determinado estado [56]. Alguns exemplos de fórmulas CTL são apresentados a seguir:

- é possível chegar a um estado em que o sistema está *iniciado* mas não *pronto*?

$$EF(iniciado \wedge \neg pronto) \tag{2.18}$$

- em qualquer caminho e em todo este caminho, se algum recurso é *requisitado*, ele eventualmente é *reconhecido*?

$$AG(requisitado \rightarrow AF\ reconhecido) \tag{2.19}$$

- um determinado processo está *habilitado* infinitamente em cada caminho de computação.

$$AG(AF\ habilitado) \tag{2.20}$$

Assim como na LTL, as fórmulas em CTL são interpretadas sobre os caminhos a partir de estados e um sistema de transição [56]. Cada *path* é entendido como um caminho de estados possível no modelo, em que a raiz da árvore representa o estado inicial. Dado o modelo de transição $\mathcal{M} = (S, \rightarrow, L)$, a relação de satisfação $\mathcal{M}, s \models\equiv$ pode ser dada indutivamente conforme é exibido na Figura 22, em que $s \in \mathcal{M}$ e φ é uma fórmula em CTL [8].

$$\mathcal{M}, s \models \top \tag{2.21}$$
$$\mathcal{M}, s \not\models \bot \tag{2.22}$$
$$\mathcal{M}, s \models p \iff p \in L(s) \tag{2.23}$$
$$\mathcal{M}, s \models \neg\varphi \iff \pi \not\models \varphi \tag{2.24}$$
$$\mathcal{M}, s \models \varphi \wedge \varphi' \iff \mathcal{M}, s \models \varphi \text{ e } \mathcal{M}, s \models \varphi' \tag{2.25}$$
$$\mathcal{M}, s \models \varphi \vee \varphi' \iff \mathcal{M}, s \models \varphi \text{ ou } \mathcal{M}, s \models \varphi' \tag{2.26}$$
$$\mathcal{M}, s \models \varphi \rightarrow \varphi' \iff \mathcal{M}, s \not\models \varphi \text{ ou } \mathcal{M}, s \models \varphi' \tag{2.27}$$
$$\mathcal{M}, s \models AX\varphi \iff \forall s_1 \mid s \rightarrow s_1 \text{ com } \mathcal{M}, s_1 \models \varphi \tag{2.28}$$
$$\mathcal{M}, s \models EX\varphi \iff \exists s_1 \mid s \rightarrow s_1 \text{ com } \mathcal{M}, s_1 \models \varphi \tag{2.29}$$
$$\mathcal{M}, s \models AG\varphi \iff \forall \pi = s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots \mid s_1 = s \text{ e } \forall s_i \in \pi \text{ com } \mathcal{M}, s_i \models \varphi \tag{2.30}$$
$$\mathcal{M}, s \models EG\varphi \iff \exists \pi = s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots \mid s_1 = s \text{ e } \forall s_i \in \pi \text{ com } \mathcal{M}, s_i \models \varphi \tag{2.31}$$
$$\mathcal{M}, s \models AF\varphi \iff \forall \pi = s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots \mid s_1 = s \text{ e } \exists s_i \in \pi \mid \mathcal{M}, s_i \models \varphi \tag{2.32}$$
$$\mathcal{M}, s \models EF\varphi \iff \exists \pi = s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots \mid s_1 = s \text{ e } \exists s_i \in \pi \mid \mathcal{M}, s_i \models \varphi \tag{2.33}$$
$$\mathcal{M}, s \models A[\varphi U \varphi'] \iff \forall \pi = s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots \mid s_1 = s \exists s_i \in \pi \mid \tag{2.34}$$
$$\mathcal{M}, s_i \models \varphi' \text{ e } \forall j < i \text{ com } \mathcal{M}, s_j \models \varphi \tag{2.35}$$
$$\mathcal{M}, s \models E[\varphi U \varphi'] \iff \exists \pi = s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots \mid s_1 = s \exists s_i \in \pi \mid \tag{2.36}$$
$$\mathcal{M}, s_i \models \varphi' \text{ e } \forall j < i \text{ com } \mathcal{M}, s_j \models \varphi \tag{2.37}$$

Figura 22 – Semântica da CTL [8].

Na CTL, os quantificadores existencial e universal pode ser aninhados para expressar propriedades mais complexas. Por exemplo, uma propriedade em que para todo

caminho de computação sempre é possível retornar para o estado inicial, pode ser representada por $AG\ (EF\ (inicio))$. Assim, interpreta-se que em todo estado do sistema (G), para todo caminho de computação (A), existe a possibilidade (E) de eventualmente retornar ao início $(F\ inicio)$ [57].

Outra expressão que não pode ser descrita em LTL é apresentada como segue:

- Um elevador pode permanecer parado no 3º andar com as portas fechadas?

$$AG(andar3 \wedge parado \wedge portaFechada \rightarrow EG(andar3 \wedge parado \wedge portaFechada)) \tag{2.38}$$

2.5 Cálculo de Processos

O cálculo de processos envolve um conjunto de abordagens para modelagem formal de sistemas concorrentes que utilizam fórmulas algébricas. Por meio de métodos descritivos de alto nível, possibilita a representação de interações, comunicações e sincronizações entre uma coleção de agentes ou processos independentes. Desta forma, descrições de processos podem ser manipuladas e analisadas, permitindo a formalização do raciocínio sobre equivalências entre processos [58].

Mediante ao uso de operadores, são definidas leis algébricas que representam interações entre processos independentes, de forma que as expressões possam ser manipuladas usando raciocínio equacional [58].

A seguir são descritos alguns cálculos utilizados para modelagem de processos concorrentes. O Cálculo de Sistemas de Comunicação (CCS) [14] serve como base para o desenvolvimento do Cálculo-π [9, 15], que por sua vez, possui uma extensão com aspecto temporal denominada TPi [10].

2.5.1 Cálculo de Sistemas de Comunicação

O CCS é um cálculo de processos desenvolvido por Milner[14] para modelagem de sistemas concorrentes sob duas ideias centrais: comunicação e sincronização [14]. Para isso, o formalismo utiliza dois componentes básicos:

- **Agente**: Também chamado de processo, é qualquer sistema concorrente o qual seu comportamento é uma ação discreta;
- **Ação**: É a interação (comunicação) entre dois agentes ou uma interação independente consigo mesmo. Para que a comunicação ocorra, os agentes devem estar sincronizados.

Com um alto nível de articulação e flexibilidade na manipulação, o CCS é aplicável não somente aos processos de *softwares*, mas também à estruturas de dados, e, sob um certo nível de abstração, para sistemas de *hardware* [14].

O cálculo permite que os processos modelados sejam observáveis em função de sua estrutura e funcionamento. Ou seja, seu comportamento pode ser observado enquanto é executado.

Algumas definições são essenciais para a compreensão da estrutura do CCS, são elas:

- $\mathcal{K}$ é o conjunto dos nomes dos processos (constantes);
- $\mathcal{A}$ define o conjunto dos canais;
- $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ representa o conjunto dos rótulos, onde $\overline{\mathcal{A}} = \overline{a} | a \in \mathcal{A}$. Elementos de $\mathcal{A}$ são chamados de canais e os de $\overline{\mathcal{A}}$ são os co-canais;
- $Act = \mathcal{L} \cup \{\tau\}$ denota o conjunto das ações, onde τ representa uma ação nula;
- O conjunto dos termos que definem um processo (expressão do processo), é denotado por $\mathcal{P}$.

Com isso, a sintaxe da CCS é apresentada conforme segue:

$$P := K \mid \alpha.P \mid \Sigma_{i \in I}\ P_i \mid P_1 | P_2 \mid P \setminus L \mid P[f] \mid \underline{if}\ true\ \underline{then}\ P\ \underline{else}\ F \tag{2.39}$$

O CCS é baseado nos elementos: constante de processo $K \in \mathcal{K}$; ação prefixada $\alpha.P$ (lê-se: α, então P), onde $\alpha \in Act$ e $P \in \mathcal{P}$; somatório $\Sigma_{i \in I} P_i \mid P_1$, no qual $P_1 + P_2 = \Sigma_{i \in \{1,2\}}\ P_i$, que representa uma escolha não determinística entre P_1 ou P_2 ($P_1, P_2 \in \mathcal{P}$); composição paralela $P_1 | P_2$, representa que os processos P_1 e P_2 estão executando paralelamente e encontram-se prontos para se comunicar; restrição $P \setminus L$ (lê-se: P restringido por L), sendo que $L \subseteq \mathcal{A}$; renomeação $P[f]$ (lê-se: P renomeado por f), dado que $P \in \mathcal{P}$ e f é uma função de renomeação denotada por $f : Act \longrightarrow Act$, em que $f(\tau) = \tau$ e $f(\overline{a}) = \overline{f(a)}$; condição $\underline{if}\ true\ \underline{then}\ P\ \underline{else}\ F$, no qual *true* indica que uma dada condição pode ser verdadeira ou não.

A fim de evitar o uso excessivo de parênteses e organizar a interpretação sobre as expressões dos processos, é dada a seguinte regra de procedência de operadores:

$$\{Restrição, Renomeação\} > Ação > Composição > Somatório$$

Portanto, por exemplo:

$$P \mid \tau.Q \setminus \alpha \ + \ R[f] \ = \ (P \mid (\tau.(Q \setminus \alpha))) \ + \ (R[f]) \tag{2.40}$$

Um sistema em CCS consiste num conjunto de definições de equações na forma $K \stackrel{def}{=} P$, sendo que $K \in \mathcal{K}$ é uma constante de processo e $P \in \mathcal{P}$ é a expressão que define um processo. O formalismo permite que cada constante de processo tenha apenas uma expressão, no qual a recursão é permitida, por exemplo: $P \stackrel{def}{=} \overline{a}.P \mid A$.

A semântica do CCS é determinada por um Sistema de Transições Rotuladas (LTS) composto relações binárias do tipo $\xrightarrow{\alpha}$, em que $\alpha \in Act \cup \{\tau\}$.

A partir de uma coleção de equações de definição, tem-se um LTS na forma $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$, no qual seus elementos são descritos respectivamente como:

- $Proc = \mathcal{P}$: conjunto de todas as expressões de processo;
- $Act = \mathcal{L} \cup \{\tau\}$: conjunto de todas as ações, inclusive a ação nula τ;
- $\{\xrightarrow{a} \mid a \in Act\}$: Relação de transição dada por regras de uma Semântica Operacional Estrutural na forma:

$$regra \ \frac{premissa}{conclusão} \ condições \tag{2.41}$$

As transições denotadas na Tabela 8 podem gerar derivações capazes de serem computadas por meio de uma árvore de derivação, como ilustrado na Figura 23, considerando os seguintes processos e suas definições:

$$\begin{aligned} A &\stackrel{def}{=} a.A', \\ A' &\stackrel{def}{=} \overline{c}.A, \\ B &\stackrel{def}{=} c.B', \\ B' &\stackrel{def}{=} \overline{b}, B \end{aligned} \tag{2.42}$$

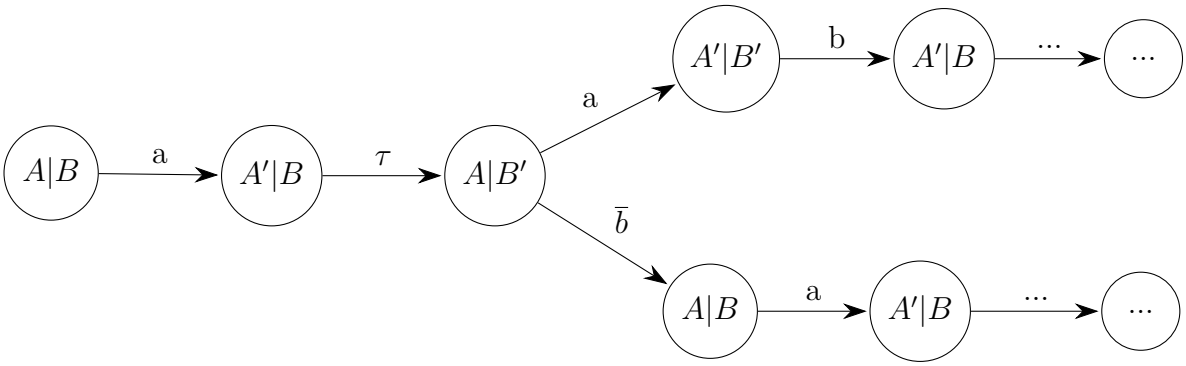

Figura 23 – Exemplo de árvore de computação num sistema de transição.

Fonte: Próprio autor.

Tabela 7 – Semântica Estrutural Operacional do CCS [14].

Regra	$\frac{\textbf{premissa}}{\textbf{conclusão}}$	**Condição**
Prefixo	$\frac{}{\alpha.P \xrightarrow{\alpha} P}$	-
Somatório$_j$	$\frac{P_j \xrightarrow{\alpha} P'_j}{\Sigma_{i\in I}\ P_i \xrightarrow{\alpha} P'_j}$	$j \in I$
Composição 1	$\frac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'\|Q}$	-
Composição 2	$\frac{Q \xrightarrow{\alpha} Q'}{P\|Q \xrightarrow{\alpha} P\|Q'}$	-
Composição 3	$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$	-
Restrição	$\frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$	$\alpha, \overline{\alpha} \notin L$
Renomeação	$\frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$	-
Condicional if_1	$\frac{P \xrightarrow{\alpha} P'}{\underline{if}\ true\ \underline{then}\ P\ \underline{else}\ Q \xrightarrow{\alpha} P'}$	$if\ true$
Condicional if_2	$\frac{Q \xrightarrow{\alpha} Q'}{\underline{if}\ true\ \underline{then}\ P\ \underline{else}\ Q \xrightarrow{\alpha} Q'}$	$if \neg true$
Definição	$\frac{(P\{y_i/x_i\}) \xrightarrow{\alpha} P'}{K(y_i) \xrightarrow{f(\alpha)} P'}$	$fornece\ K(x_i) \stackrel{def}{=} P$

Uma característica presente no CCS é a passagem de valores, que permite que valores específicos possam ser enviados ou recebidos pelos processos (e.g., inteiros e *Strings*). Para isso, ações que representam canais e co-canais são utilizadas na notação prefixada com a forma $a(x).P$ ou $\overline{a}(x).P$, a qual x simboliza um valor arbitrário a ser transmitido ou rebebido. Os canais $\mathcal{A}$ e os co-canais $\overline{\mathcal{A}} = \{\overline{a} | a \in \mathcal{A}\}$ caracterizam os canais de recepção e saída dos valores, respectivamente.

Na Figura 24 são apresentados exemplos de passagem de valor, em que é aplicada uma troca de valor simples e outra infinita, de acordo com os processos modelados abaixo.

$$P \stackrel{def}{=} in(x).P'(x)$$
$$P'(x) \stackrel{def}{=} \overline{out}(x).P \tag{2.43}$$

$$P \stackrel{def}{=} \sum_{i \in \mathbb{N}} in(i).C'_i$$
$$C'_i \stackrel{def}{=} \overline{out}(i).C \tag{2.44}$$

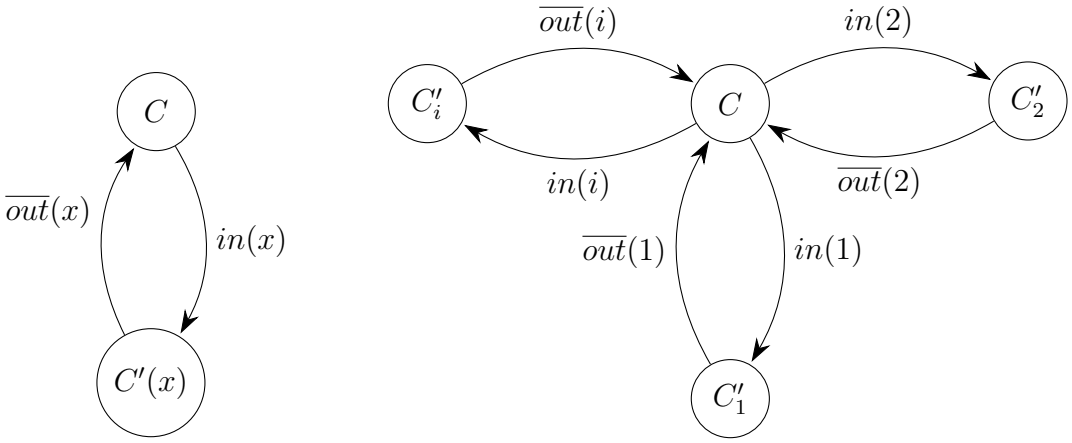

Figura 24 – Passagem de valores em CCS.

Fonte: <http://www.ru.is/faculty/luca/IMTCOURSE/SLIDES/l2.pdf>

2.5.2 Cálculo-π

O Cálculo-π, desenvolvido por Milner, Parrow e Walker[9], é uma extensão do CCS, no qual aspectos de mobilidade são acrescentados e sua complexidade é reduzida, reforçando sua teoria básica, enquanto preserva suas propriedades algébricas [9].

Com este formalismo, os nomes dos canais (ou *links*) aparecem como parâmetros na comunicação, portanto, vai além do CCS. Apesar da adição de variáveis aos canais, bem como sobre valores de dados comuns, o cálculo não tornou-se mais complexo. De fato, todas as distinções entre nomes de canais, variáveis e demais dados foram removidas, e, portanto, todos são chamados de *names*. Logo, tem-se duas classes de entidades: *names* e agentes [9].

Para quaisquer definições adiante, pressupõe-se que x, y, z, w, v e u são meta-variáveis representadas por nomes, pertencentes ao conjunto $\mathcal{N}$ de nomes. Presume-se também que há um conjunto de identificadores de agente, em que cada identificador A possui aridade maior ou igual a zero. P, Q e R representam o conjunto do comportamento de um agente.

A sintaxe dos agentes pode ser resumida como segue:

$$P ::= \ 0 \mid \overline{y}x.P \mid y(x).P \mid \tau.P \mid (x)P \mid [x = y]P \mid P_1|P_2 \mid P_1 + P_2 \mid A(y_1, ..., y_n) \quad (2.45)$$

Considera-se que 0 é um operador nulo. Os operadores $\overline{x}y.$, $x(y).$, $\tau.$, (x), e $[x = y]$ são unários, enquanto que $|$ e $+$ são binários, e n indica a aridade de A [15].

Para cada agente nas formas $\overline{y}x.P$ e $y(x).P$, a ocorrência de y entre parênteses é chamada de ocorrência de ligação. Uma ocorrência de y de um agente é considerada "livre" se a mesma não estiver no âmbito de uma ocorrência de ligação. Portanto, o conjunto de nomes livres (*free names*) em P é denotado por $fn(P)$.

O operador de correspondência $[x = y]P$ indica que o agente irá se comportar como P caso os nomes x e y sejam idênticos. Caso contrário, agirá como 0.

Uma equação de definição de um identificador de agente A é dada por

$$A(x_1, ..., x_n) \stackrel{def}{=} P,$$

em que os elementos do tipo x_i são distintos e $fn(P) \subseteq \{x_1, ..., x_n\}$.

Se uma ocorrência de um nome em um agente não é livre, então é chamado de nome "vinculado" (*bound names*). Assume-se $bn(P)$ como o conjunto de nomes vinculados de P. Se $A(\tilde{x}) \stackrel{def}{=} Q$, então $bn(A(\tilde{x})) = bn(Q)$, no qual $\tilde{x} = x_1, ..., x_n$. Portanto, $n(P)$ representa o conjunto $fn(P) \cup bn(P)$ dos nomes contidos em P.

A substituição é uma função σ de $\mathcal{N}$ pra $\mathcal{N}$ em que, quando $x_i\sigma = y_i$, no qual $1 \leq i \leq n$ (e $x\sigma = x$ para os demais nomes x), pode-se escrever $\{y_1/x_1, ..., y_n/x_n\}$ ou $\{\tilde{y}/\tilde{x}\}$ para σ. Ou seja, para todo x_i haverá um y_i que substituirá seu valor.

As ações do Cálculo-π podem ser observadas por meio de um LTS, assim como no CCS. este formalismo define 4 quatro tipos de ações α:

- **Ação nula** τ: $P \xrightarrow{\tau} Q$ expressa que P pode envolver Q de forma que isso exija alguma interação com o ambiente. As ações nulas podem surgir na forma $\tau.P$, e também em comunicações entre agentes;
- **Saída livre** $\overline{x}y$: $P \xrightarrow{\overline{x}y} Q$ implica que P emite o nome livre y pela porta de saída $\overline{x}$ e é representada pela forma prefixada $\overline{x}y.P$;
- **Entrada** $x(y)$: $P \xrightarrow{x(y)} Q$ define que P recebe qualquer nome w pela porta x, que envolve $Q\{w/y\}$. Enquanto que em CCS, a ação de entrada contêm o valor atual a ser recebido, no Cálculo-π y representa uma referência ao local onde o nome recebido deve ir. Tal ação pode mostrar-se na forma $x(y).P$;

- **Saída vinculada** $\overline{x}(y)$: $P \xrightarrow{\overline{x}(y)} Q$ denota que P emite um nome particular (i.e., um nome vinculado de P) pela porta de saída $\overline{x}$, em que (y) é a referência de onde o nome particular ocorre. Ações de saída vinculadas estão presentes em ações de saída livre que carregam nomes fora de seu escopo, por exemplo, em $(y)\overline{x}y.P$.

A semântica do Cálculo-π é apresentada na Tabela 8, e, assim como no CCS, é declarada por meio de uma Semântica Operacional Estrutural que leva em conta as relações de transição de seus operadores.

Tabela 8 – Semântica Operacional Estrutural do Cálculo-π [15].

Regra	$\frac{\textbf{premissa}}{\textbf{conclusão}}$	**Condição**
Ação vazia	$\frac{}{\tau.P \xrightarrow{\tau} P}$	-
Ação de saída livre	$\frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$	-
Ação de entrada	$\frac{}{x(z).P \xrightarrow{x(w)} P\{w/z\}}$	$w \notin fn((z)P)$
Somatório	$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$	-
Correspondência	$\frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}$	-
Identificador	$\frac{P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'}$	$A(\tilde{x}) \stackrel{def}{=} P$
Composição paralela	$\frac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'\|Q}$	$bn(\alpha) \cap fn(Q) = \varnothing$
Comunicação	$\frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'\{y/z\}}$	-
Fechamento de escopo	$\frac{P \xrightarrow{\overline{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P\|Q \xrightarrow{\tau} (w)(P'\|Q')}$	-
Abertura de escopo	$\frac{P \xrightarrow{\overline{x}y}}{(y)P \xrightarrow{\overline{x}(w)} P'\{w/y\}}$	$y \neq x, w \notin fn((y)P')$
Restrição	$\frac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'}$	$y \notin n(\alpha)$

Um sistema em Cálculo-π pode ser representado por meio de um grafo de fluxo. Supõe-se que um agente P deseja transferir para um novo agente Q a tarefa de transmitir o valor 5 para R. Para isso, presume-se que P está conectado ao Q, inicialmente, por meio de um canal b.

Portanto, tem-se $P \equiv \overline{b}a.\overline{b}5.P'$, no qual tanto o canal a quanto o valor 5 são transmitidos ao longo de a. Sendo assim, o processo $Q \equiv b(y).b(z).\overline{y}z.\mathbf{0}$, o qual recebe, em ordem, um canal e um valor através de b, e então transmite o valor pelo canal, e, por fim, é encerrado. O sistema completo é expressado a seguir:

$$(\overline{b}a.\overline{b}5.P' \mid b(y).b(z).\overline{y}z.\mathbf{0} \mid a(x).R') \setminus a \setminus b \xrightarrow{\tau} (P' \mid \overline{a}5.\mathbf{0} \mid a(x).R') \setminus a \setminus b \qquad (2.46)$$

A Figura 25 a seguir apresenta o grafo de fluxo do sistema:

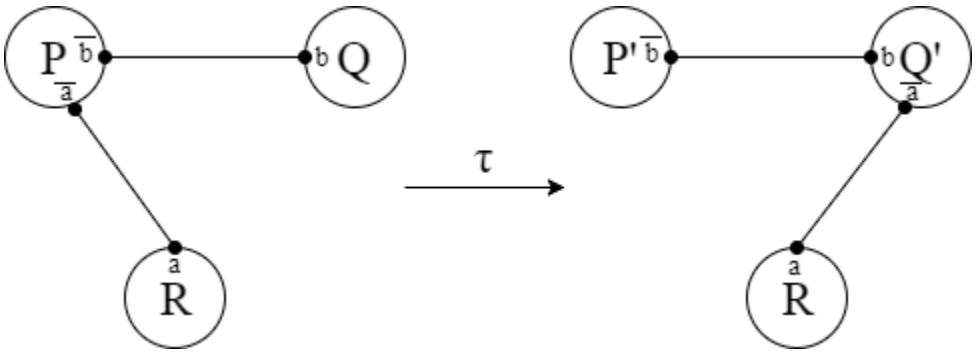

Figura 25 – Grafo de fluxo do sistema (2.46) [9].

2.5.3 Álgebra temporal de processos TPi

A TPi é uma álgebra temporal de processos inspirada no Cálculo-π [9, 15] e desenvolvida por Berger e Honda[10]. O formalismo apresenta um cálculo para transmissão síncrona de mensagens capaz de expressar entradas cronometradas [11].

A sintaxe da linguagem define processos $P, Q \in \mathcal{P}$, baseado nos nomes $x, y \in \mathcal{N}$ como segue:

$$P, Q ::= \overline{x}\langle y\rangle.P \mid timer^t(x(y).P, Q) \mid !P \mid (vx)P \mid (P|Q) \mid (P+Q) \mid \mathbf{0} \mid A(x) \quad (2.47)$$

A sintaxe corresponde aos operadores do Cálculo-π, exceto pelas ações de entrada representadas com o acréscimo de um cronômetro, de forma $timer^t(x(y).P, Q)$, em que $t \in \mathbb{N}$ simboliza um tempo limite, o qual é decrementado de acordo com o ambiente do processo, podendo t assumir qualquer unidade de tempo. Desta forma, enquanto $t > 0$, a ação de entrada $x(y).P$ pode atuar paralelamente com ações de saída. Em contrapartida, quando $t = 0$, o processo age como Q [11].

Chamadas de definição de processos são expressas na forma $A(x)$, as quais invocam uma definição de processo $A(y) \stackrel{def}{=} P$, enquanto que o valor x é passado e substitui y. Como a definição A não aceita quaisquer parâmetros de entrada, o parâmetro y é omitido. Assim, a definição é escrita como $A() \stackrel{def}{=} P$ [11].

A Semântica Operacional Estrutural da TPi é dada em termos de sua congruência estrutural $\equiv$, e a relação de suas transições rotuladas $\xrightarrow{\alpha}$ é apresentada na Figura 26, em que $fn(P)$ denota o conjunto de nomes livres de P.

Definições de $\equiv$ são padronizadas [15], exceto pelas regras 2.53, 2.54, 2.55 e 2.56, as quais lidam com cronômetros zerados e infinitos, e chamadas de definição de processos parametrizadas e não parametrizadas, respectivamente.

Os rótulos $\alpha \in \{\xrightarrow{\overline{x}\langle y\rangle}, \xrightarrow{\overline{x}(y)}, \xrightarrow{x(z)}, \xrightarrow{\tau}\}$, expressam saídas livres e vinculadas, entradas e ações nulas.

As regras de transição $\xrightarrow{\alpha}$ são descritas detalhadamente por Berger e Honda[10], exceto pela regra 2.66, que define uma função de passagem de tempo $\eth : \mathcal{P} \longrightarrow \mathcal{P}$, que expressa a marcação de tempo dos cronômetros ativos, no qual uma entrada temporizada está aguardando a aceitação de uma mensagem, em que:

$$\eth(P) = \begin{cases} timer^t(x(y).Q, Q'), & se\ P = timer^{t+1}(x(y).Q, Q'), \\ & e\ 0 < t+1 < \infty \\ \eth(Q)|\eth(R), & se\ P = Q|R \\ \eth(Q) + \eth(R), & se\ P = Q + R \\ (vx)\eth(Q), & se\ P = (vx)Q \\ \eth(Q[x/y]), & se\ P = A(x)\ e\ A(y) \stackrel{def}{=} Q \\ \eth(Q), & se\ P = A()\ e\ A() \stackrel{def}{=} Q \\ P, & \text{outra forma} \end{cases}$$

Regras da relação $\equiv$:

$$\mathcal{P}/\equiv, |, \mathbf{0}) \text{ é um monóide comutativo} \tag{2.48}$$
$$(vx)\mathbf{0} \equiv \mathbf{0} \tag{2.49}$$
$$(vx)(vy)P \equiv (vy)(vx)P \tag{2.50}$$
$$!P \equiv P|!P \tag{2.51}$$
$$(vx)(P|Q) \equiv (P|(vx)Q) sex \notin fn(Q) \tag{2.52}$$
$$timer^0(x(z).P, Q) \equiv Q \tag{2.53}$$
$$timer^\infty(x(z).P, Q) \equiv x(z).P \tag{2.54}$$
$$A(x) \equiv P[x/y], \text{ em que } A(y) \stackrel{def}{=} P \tag{2.55}$$
$$A() \equiv P, \text{ em que } A() \stackrel{def}{=} P \tag{2.56}$$

Regras da relação $\stackrel{\alpha}{\longrightarrow}$:

$$\overline{x}\langle y\rangle.P \stackrel{\overline{x}\langle y\rangle}{\longrightarrow} P \tag{2.57}$$
$$timer^{t+1}(x(z).P, Q) \stackrel{x(z)}{\longrightarrow} P \tag{2.58}$$
$$P \stackrel{\overline{x}\langle y\rangle}{\longrightarrow} Q \quad \Rightarrow \quad (vy)P \stackrel{\overline{x}(y)}{\longrightarrow} se\ x \neq y \tag{2.59}$$
$$P \stackrel{\overline{x}\langle y\rangle}{\longrightarrow} P', Q \stackrel{x(z)}{\longrightarrow} Q' \quad \Rightarrow \quad P|Q \stackrel{\tau}{\longrightarrow} P'|Q'[y/z] \tag{2.60}$$
$$P \stackrel{\overline{x}(y)}{\longrightarrow} P', Q \stackrel{x(z)}{\longrightarrow} Q' \quad \Rightarrow \quad P|Q \stackrel{\tau}{\longrightarrow} (vy)(P'|Q'[y/z]) \tag{2.61}$$
$$P \stackrel{\alpha}{\longrightarrow} Q \quad \Rightarrow \quad (vx)P \stackrel{\alpha}{\longrightarrow} (vx)Q\ se\ x \neq fn(\alpha) \tag{2.62}$$
$$P \stackrel{\alpha}{\longrightarrow} P' \quad \Rightarrow \quad P|Q \stackrel{\alpha}{\longrightarrow} P'|Q \tag{2.63}$$
$$P \stackrel{\alpha}{\longrightarrow} P' \quad \Rightarrow \quad P + Q \stackrel{\alpha}{\longrightarrow} P' \tag{2.64}$$
$$P \stackrel{\alpha}{\longrightarrow} P' \quad \Rightarrow \quad Q + P \stackrel{\alpha}{\longrightarrow} P' \tag{2.65}$$
$$P \stackrel{\tau}{\longrightarrow} \eth(P) \tag{2.66}$$

Figura 26 – Semântica Estrutural Operacional do TPi [10].

3 TRABALHOS RELACIONADOS

Uma abordagem para modelagem do protocolo MQTT escrita em TPi [10], foi proposta por Aziz[11]. Se por um lado, a TPi oferece um alto nível de abstração da comunicação concorrente entre múltiplos agentes, por outro, a mesma não é capaz de formalizar informações importantes como descrição detalhada das ações, conteúdo das mensagens, ordenação da execução dos sistemas e definição dos agentes e suas propriedades.

Um contrato multilateral consiste num conjunto de compromissos envolvendo o acordo entre múltiplos agentes para garantir o cumprimento de cláusulas, a fim de satisfazer as vontades das partes envolvidas, formalizando assim, uma relação contratual. O modelo de contrato multilateral proposto por Xu[27], oferece um alto nível de detalhamento das relações existentes entre os agentes e suas ações, permitindo a formalização completa do contrato, além de proporcionar a possibilidade da aplicação de um método de monitoramento reativo para detecção das partes responsáveis por violações contratuais [27, 28].

A seguir são apresentados os trabalhos que estão mais intimamente relacionados com esta pesquisa. Inicialmente é apresentada a técnica de modelagem do protocolo MQTT e posteriormente técnicas para formalização, monitoramento reativo e detecção de violação de contratos multilaterais.

3.1 Especificação formal de um protocolo de IoT

Devido ao crescimento e popularização da IoT, estima-se que bilhões de novos dispositivos estejam interconectados num futuro próximo, gerando bilhões em receita e promovendo maior conforto e bem estar na vida dos usuários, além de otimizar processos industriais e outras aplicações. Tais fatores potencializam a proliferação e demanda por equipamentos com baixo poder de processamento e armazenamento, como sensores e leitores RFID. Com base nisso, são impulsionados os esforços pela padronização da comunicação M2M. Assim, protocolos de comunicação como o MQTT [6], XMPP [38], CoAP [5] e outros, surgiram como solução para lidar com a comunicação entre dispositivos inseridos no ambiente de IoT [11].

Aplicações em IoT podem carecer de um alto nível de confiança em termos de corretude das especificações dos sistemas e assegurar que propriedades não funcionais sejam cumpridas, como segurança e privacidade. Sendo assim, é de suma importância a adoção de métodos formais de análise que garantam que as especificações tenham o mínimo de ambiguidade possível, oferecendo maior confiança e robustez para aplicações que adotam tais especificações [11]. Diante desta questão, foi proposto por Aziz[11] uma método

para modelagem e análise do protocolo MQTT, que auxilia na compreensão do comportamento operacional do protocolo, bem como na detecção de ambiguidades presentes na sua especificação.

Modelo

O comportamento operacional do protocolo foi modelado com base nas especificações descritas em sua documentação [6] por meio de um cálculo temporal de processos chamado TPi [10], descrito na Seção 2.5.3. O modelo é baseado na entrega de pacotes PUBLISH, de acordo com os níveis de QoS oferecidos no MQTT, cujas características são discutidas na Seção 2.2.4. A modelagem completa é exposta na Figura 27.

QoS nível 0:

$$
\begin{aligned}
&Cliente(Publish) \mid Servidor(),\ em\ que: \\
&Cliente(z) \stackrel{def}{=} \overline{c}\langle z\rangle \\
&Servidor() \stackrel{def}{=} c(x).\overline{pub}\langle x\rangle
\end{aligned} \tag{3.1}
$$

QoS nível 1:

$$
\begin{aligned}
&Cliente(Publish) \mid Servidor(),\ em\ que: \\
&Cliente(z) \stackrel{def}{=} \overline{c}\langle z\rangle.timer^{t}(c'(y), Cliente(Publish_{DUP}) \\
&Servidor() \stackrel{def}{=} !c(x).\overline{pub}\langle x\rangle.\overline{c'}\langle Puback\rangle
\end{aligned} \tag{3.2}
$$

QoS nível 2:

$$
\begin{aligned}
&Cliente(Publish) \mid Servidor(),\ em\ que: \\
&Cliente(z) \stackrel{def}{=} \overline{c}\langle z\rangle.timer^{t}(c(y).ClienteCont(y), Cliente(Publish_{DUP})) \\
&ClienteCont(u) \stackrel{def}{=} \overline{c'}\langle Pubrel_u\rangle.timer^{t'}(c'(w), ClienteCont(u)) \\
&Servidor() \stackrel{def}{=} !c(l).(ServidorLate(l) + ServidorEarly(l)) \\
&ServidorLate(x) \stackrel{def}{=} (\overline{c}\langle Pubrec_x\rangle.c'(v).\overline{pub}\langle x\rangle.\overline{c'}\langle Pubcomp_v\rangle.!(c'(v').\overline{c'}\langle Pubcomp_{v'}\rangle)) \\
&ServidorEarly(x) \stackrel{def}{=} (\overline{pub}\langle x\rangle.c'(v).\overline{pub}\langle x\rangle.\overline{c}\langle Pubrec_x\rangle.c'(q).\overline{c'}\langle Pubcomp_q\rangle. \\
&\qquad\qquad !(c'(q').\overline{c'}\langle Pubcomp_{q'}\rangle))
\end{aligned} \tag{3.3}
$$

Figura 27 – Modelo do protocolo MQTTv3.1.1 em TPi baseado nos três níveis de QoS [11], adaptado pelo autor.

3.2 Um modelo para contratos multilaterais

A execução de contratos entre múltiplas partes de um negócio representam um ofício relevante dentro da economia global, em que atividades ao longo da cadeia de valor

são executadas de forma independente, porém cooperativamente entre as companhias [43]. Sendo assim, é de suma importância que as partes envolvidas em uma relação contratual cumpram com as cláusulas das quais são responsáveis, e, em caso de não cumprimento, ou seja, quando há uma violação do contrato, o responsável pela violação deve ser identificado.

Uma forma de identificar o responsável por uma violação baseia-se em "quebrar" o contrato num conjunto de contratos bilaterais, e então apontar o responsável, já que essa é uma tarefa relativamente mais simples de ser executada em contratos bilaterais. Porém, pesquisas como a de Haugen e Fletcher[59] demonstram que, apesar da ideia parecer viável, em um relacionamento multilateral complexo, onde há uma série de dependências envolvendo múltiplos agentes, essa conversão pode resultar na perda ou ocultação de informação.

Diante disso, foi proposto por Xu[27] um modelo de contrato multilateral capaz de representar relações contratuais complexas entre múltiplos agentes, assim como realizar o monitoramento do contrato e identificar os responsáveis por violações.

Modelo de contrato multilateral

De acordo com Xu[27], um contrato é o acordo entre duas ou mais partes baseado em compromissos mútuos. Assim, o modelo proposto consiste em três partes:

- **Ação**: Descreve o que cada parte deverá fazer;
- **Compromisso**: É a garantia entre as partes envolvidas de que um conjunto de ações deverá ser completamente executado, e todas as partes envolvidas irão cumprir sua parte;
- **Grafo de compromisso**: É uma forma de visualizar os compromissos entre as partes, demonstrando assim os relacionamentos de compromissos entre elas.

Ações

As ações representam um átomo do modelo contratual e constituem as arestas do grafo de comprometimento. Um agente envolvido num contrato pode assumir diferentes papéis. A coleção dos papéis de um agente é especificada como $\mathcal{R}_x$, no qual x é a identificação do agente, e o conjunto de todos os papéis de um contrato é denotado por $\mathbb{R}$. As ações e todas as outras definições do contrato estão reunidas em um conjunto denominado ID.

Posto isto, uma ação é um quadrupla $a = name, sender, receiver, deadline$, no qual:

- $name \in ID$ é o identificador da ação;

- $sender, receiver \in \mathbb{R}$ são os papéis;
- $deadline \in \mathbb{T}$ é uma representação simplificada do tempo (dias, horas, segundos, etc).

O identificador (*name*) deve ser único para cada ação, permitindo seu rastreamento. A coleção de todas as ações é denotada pelo conjunto $\mathbb{A}$, tal que

$$\mathbb{A} = \bigcup_{\forall x \in \mathcal{P}} \{action\}.$$

Compromisso

Um compromisso é uma garantia de uma das partes que a outra parte irá cumprir com o combinado no contrato. Considerando que um contrato multilateral é formado por um ou mais compromissos, um compromisso inclui ações que deverão ser executadas por um ou mais agentes. Tais ações podem acionar ($trigger$), implicar ($involve$) ou terminar ($finish$) o compromisso. Estes atributos pertencem ao conjunto $\mathcal{U} = \{tr, in, fi\}$.

Considerando o conjunto de identificadores ID, o conjunto das partes de um contrato como $\mathcal{P}$, o numerador $N = \{1, 2, 3, ...\}$ e o conjunto de ações denotado por $\mathbb{N}$, um compromisso pode ser denotado pela quíntupla

$$commitment = (name, sender, receiver, n, \{(a_1, u_1), (a_2, u_2), ..., (a_n, u_n) \colon a_i \in \mathbb{A}, u_i \in \mathbb{U}\}).$$

Em que:

- *name* é o identificador único do compromisso;
- *sender* é a parte do contrato que atua como expedidor do compromisso;
- *receiver* é a parte do contrato que atua como receptor do compromisso;
- n representa o número total de ações envolvidas, tal que $n \in \mathbb{N}$;
- $(a_1, a_2, ..., a_n)$ *e* $(u_1, u_2, ..., u_n)$ representam o par ordenado das ações e seus atributos.

A conjunto de todos os compromissos de um contrato é denotado por $\mathbb{M}$ e representado como

$$\mathbb{M} = \bigcup_{\forall x \in \mathcal{P}} \{commitment\}.$$

Considerando que $a_i \in \mathbb{A}$, $m \in \mathbb{M}$ e a função de sequência $f_{position} \colon \mathbb{A} \times \mathbb{M} \to N$, então $f_{position}(a_1, m)$ define a posição de uma ação a_i em um compromisso m.

Grafo de compromisso

Os compromissos tem um aspecto dinâmico, e o grafo de compromissos mostra os relacionamentos complexos entre os compromissos do contrato.

O grafo de compromisso é um grafo direcionado que consiste em um conjunto de nós que corresponde a todos os papéis contidos em $\mathbb{R}$, e em um conjunto de arestas correspondendo às suas ações, seus rótulos e às ordens dos compromissos.

Sendo $\mathbb{A}$ o conjunto de ações, em que $a \in \mathbb{A}$, $\mathbb{M}$ é um conjunto de compromissos, tal que $m \in \mathbb{M}$ e $X = \{1, 2, 3, 4, ...\}$, a função de sequência $f_{position}(a, m)$, as arestas são representadas pela relação $\mathbb{A} \times \mathbb{M} \times X$, como segue:

$$edge = \bigcup_{\forall x \in \mathcal{P}} \{(a, m, f_{position}(a, m)) \colon a \in \mathbb{A}, m \in \mathbb{M}, f_{position}(a, m) \in X$$

O conjunto de todas as arestas é denotado por

$$\mathbb{E} = \bigcup_{\forall x \in \mathcal{P}} \{edge\}.$$

A ordem dos compromissos indica como eles irão ocorrer. Se m_2 é ativado após o término de m_1, então $m_1.m_2$. A relação $M \times M$ expressa a ordem de ocorrência de um compromisso, conforme segue:

$$ordercommitment = \{(m_1.m_2) \colon m_1, m_2 \in \mathbb{M}, m_1 \neq m_2\}$$

Tendo $\mathcal{P}$ como o conjunto de todos os agentes do contrato, o conjunto de todas as ordens de compromisso de todos os relacionamentos nos quais os compromissos ocorrem ordenadamente pode ser especificado como

$$\mathbb{O} = \bigcup_{\forall x \in \mathcal{P}} \{m_1.m_2\}.$$

Por fim, o grafo de compromisso pode ser definido como

$$G = (\mathbb{R}, \mathbb{E}, \mathbb{O}).$$

Desta forma, um contrato multilateral pode ser representado formalmente, sendo $\mathbb{A}$ o conjunto de ações, $\mathbb{M}$ o conjunto de todos os compromissos e G o grafo de compromisso, da seguinte forma:

$$Contract = \{\mathbb{A}, \mathbb{M}, G\}$$

3.3 Teste de violação em contratos multilaterais

No estágio de execução do contrato, identificar o responsável por uma violação é a tarefa mais importante num processo de monitoramento [28]. A abordagem usada

neste trabalho foi proposta por Xu, Jeusfeld e Grefen[28], e consiste no uso do grafo de compromisso e propriedades dos papéis desempenhados pelos agentes para detectar as partes responsáveis pela violação contratual.

Durante a execução de um contrato, um monitor do contrato constata que há um conjunto de ações que, quando executadas, impedem a realização de outra determinada ação. Esta ação é denotada como a_{miss}, e a última ação realizada até então, é definida como a_{done}. A partir das ações a_{miss}, a_{done} e suas posições no grafo de compromisso, o grafo pode ser restruturado [28]. O processo de restruturação é descrito pelo Algoritmo 1.

Primeiramente, todas as ações que seriam executadas após a_{miss} são localizadas. A princípio, os compromissos no qual a_{miss} está envolvido devem ser evidenciados, os mesmos são denotados por $M_{current}$. A partir do compromisso $m \in M_{current}$, as posições das ações $(\exists a', f_{position}(a', m))$ estão adiante da posição de a_{miss}. As ações não executadas são contidas no conjunto E_{not_occur}, e caso estejam envolvidas em outros compromissos que contenham ações posteriores à a_{miss}, estas também devem ser incluídas em E_{not_occour}. Compromissos posicionados antes de $M_{current}$ são adicionados em $M_{not\_trigger}$, e suas ações são inseridas em E_{not_occur}.

Em seguida, ações anteriores à a_{done} são executadas. Cada ocorrência de a_{done} em cada compromisso de $M_{current}$ é identificada. Com base no compromisso $m \in M_{current}$, as ações posteriores à a_{done} posicionadas em $(\exists a', f_{position}(a', m))$ são inseridas em E_{done}. Caso estas ações estejam inclusas em compromissos que possuam ações anteriores à a_{miss}, então estas também devem ser alocadas em E_{done}.

Por fim, o conjunto $E_{not\_occur \wedge E_{done}}$ representa as ações que devem ser simplificadas. A função $RestructureCG(E)$ realiza a simplificação, resultando no conjunto reduzido de arestas E. Vértices que não possuírem nenhuma conexão também devem ser eliminados.

Com base no grafo de compromisso restruturado, é checado se o papel expedidor de a_{miss} teve suas entradas recebidas. O processo de detecção das partes responsáveis por violações contratuais segue os seguintes passos:

- A parte do contrato que tem um papel em execução verifica a violação baseada nas entradas das propriedades do papel;
- Se a violação estiver presente na entrada do papel, então as saídas de outros papéis correspondentes à entrada violada devem ser localizadas;
- A ação que produz a saída causadora da violação é identificada;
- Se a ação ainda não ocorreu e a condição para sua ocorrência é satisfeita, então o expedidor da ação é o responsável;
- Os passos acima são repetidos até que o responsável seja identificado.

Algoritmo 1: Simplificação do grafo de compromisso

Entrada : $Contract = \{\mathbb{A}, \mathbb{M}, G\}$,
Ação perdida: a_{miss}
Ação feita: a_{done}

Inicialização: $M_{not\_triggered} = \emptyset$
$E_{not\_occur} = \emptyset$
$M_{finished} = \emptyset$
$E_{done} = \emptyset$
$E_{cut} = \emptyset$

begin

▷ /*Primeiro, encontrar ações que ainda não ocorreram*/

$$M_{current} = \bigcup_{\forall m, m \in \mathbb{M}} \{m, \exists edge, edge(a_{miss}, m, f)\}$$

$$M_{not\_triggered} = \bigcup_{\forall m, m \in M_{current}} \{m' \colon m.m' \in \mathbb{O}\}$$

for $m \in M_{current}$ **do**
  for $\exists a', f_{position}(a', m) > f$ **do**
    $E_{not\_occur=\{a'\} \cup E_{not\_occur}}$;
    for $\exists m', \forall a'' \colon edge(a', m', f') \vee f_{position}(a'', m') > f \vee a'' \notin E_{not\_occur}$ **do**
      $E_{not\_occur} = \{a''\} \cup E_{not\_occur}$;
    end
  end
end
for $m \in M_{not\_triggered}$ **do**
  for $\exists a, m.acao = a \vee a \notin E_{not\_occur}$ **do**
    $E_{not\_occur=\{a\} \cup E_{not\_occur}}$;
  end
end
▷ /* Segundo, obter ações que já executaram */

$$M_{current} = \bigcup_{\forall m, m \in \mathbb{M}} \{m, \exists edge, edge(a_{done}, m, f)\}$$

for $m \in M_{current}$ **do**
  for $\exists a', f_{position}(a', m) < f \vee a' \notin E_{done}$ **do**
    $E_{done} = \{a'\} \cup E_{done}$;
    for $\exists m', \forall a'' \colon edge(a', m', f') \vee f_{position}(a'', m') < f \vee a'' \notin E_{done}$ **do**
      $E_{done} = \{a''\} \cup E_{done}$;
    end
  end
end
▷/* Terceiro, restruturar o grafo de compromisso */
$E_{cut} = E_{not\_occur} \cup E_{done}$;
RestructureCG(E_{cut});

end

O processo de detecção das partes responsáveis pela violação é descrito no Algoritmo 2.

Algoritmo 2: Detectar violação

E é o conjunto de arestas ($edges$) resultante do grafo de compromisso restruturado;
Chamar função check(a_{miss})
begin
  check(a: uma ação) **repeat**
    $E = E - \{a\}$;
    if $\exists input, property(a.sender, input, \_, \_) \wedge$ $recebida(input) = True$ **then**
      return($a.sender$)
    else if $\exists output, rule$: $property(a'.sender, \_, output, rule) \vee output = input \vee \exists a', a' \in rule, condicao \vee output \in rule.resultado$ **then**
      Chamar função check(a');
    end
  until $E = \emptyset$;
end

Com essa estrutura é possível detectar os responsáveis pela violação do contrato, realizando o monitoramento reativo, isto é, aquele que ocorre após a violação do contrato.

4 UM MÉTODO PARA MONITORAMENTO REATIVO DO PROTOCOLO MQTT

A tarefa de identificar os responsáveis por violações contratuais é extremamente necessária na área de contratação eletrônica. Da mesma forma, em um ambiente de IoT, pode ser de interesse das partes envolvidas identificar os responsáveis por falhas de comunicação, como um sensor de incêndio que deixa de enviar um alerta, um sistema de cuidados médicos que deixa de emitir um aviso quando uma situação crítica é detectada, ou até mesmo um atuador que não é ativado, não por falha interna, mas devido a mensagem de ativação que deveria ter recebido ser perdida por conta de fatores externos, como interferências.

Conforme discutido na Seção 2.2.5, um dos principais desafios da IoT envolve a comunicação eficiente entre dispositivos limitados. O MQTT [6] é um dos protocolos que surgiram para lidar com este problema, e seu uso tende a crescer, visto que a área da IoT encontra-se em pleno crescimento. Sendo assim, haverão cada vez mais dispositivos conectados operando sobre o MQTT.

Assim como numa relação contratual, em um ambiente de IoT, os dispositivos estão operando de acordo com regras pré estabelecidas no âmbito de se atingir os objetivos do sistema. Portanto, cada objeto conectado desempenha uma função e executa ações com base em compromissos que o sistema se propõe a cumprir. Desta forma, pode-se considerar esse conjunto de compromissos e funções pré configuradas como sendo o protocolo do sistema, que atua sobre um ou mais protocolos de comunicação. Com base nisso, este Capítulo propõe um método para monitoramento reativo de protocolos de IoT que atuam sobre o MQTT.

O método proposto consiste em modelar a comunicação entre os dispositivos do ambiente de IoT de acordo com as características operacionais do MQTT. O modelo é obtido utilizando um cálculo temporal de processos concorrentes chamado TPi [10], que apresenta alto nível de abstração para a representação da comunicação entre processos, que também podem ser chamados de agentes, ou partes envolvidas. Para maior detalhamento das ações e suas ordens de execução, foi desenvolvido um algoritmo para conversão do protocolo modelado em TPi para um modelo de contrato multilateral [27]. Com o contrato obtido, é possível aplicar o algoritmo de monitoramento reativo proposto e aplicado por Xu[27] [28, 43].

A modelagem dos protocolos é feita com base no comportamento operacional do MQTT com nível de QoS igual a zero, quando uma mensagem é enviada no máximo uma vez e não há confirmação de recebimento [6]. A pesquisa para viabilidade da aplicação do

método para os demais níveis de QoS é sugerida como trabalho futuro. Detalhes sobre os três níveis de QoS são discutidos na Seção 2.2.4.

Com o intuito de manter o método proposto autocontido neste Capítulo, algumas propriedades relacionadas ao protocolo MQTT, e a estrutura sintática do TPi e de contratos multilaterais serão apresentadas novamente quando necessárias.

4.1 Análise do modelo e estrutura operacional do QoS 0

De acordo com a documentação do protocolo MQTT [6], o QoS 0 é definido pela regra "*At most once*" (No máximo uma vez), em que o pacote PUBLISH é enviado apenas uma vez, e recebido uma ou nenhuma vez, visto que não há nenhum tipo de confirmação de recebimento documentada.

A declaração normativa MQTT-4.3.1-1 [6] define apenas uma ação que deve ser obrigatoriamente cumprida:

- O expedidor deve enviar um pacote PUBLISH com QoS=0 e DUP=0.

Assim sendo, o modelo construído por Aziz[11] escrito em TPi é apresentado com algumas alterações como segue:

QoS nível 0:

$$
\begin{aligned}
&Cliente(Publish) \mid Servidor() \mid ClienteSub(),\ em\ que: \\
&Cliente(z) \stackrel{def}{=} \overline{c}\langle z\rangle \\
&Servidor() \stackrel{def}{=} c(x).\overline{pub}\langle x\rangle \\
&ClienteSub \stackrel{def}{=} pub(y)
\end{aligned} \tag{4.1}
$$

O modelo determina que os processos *Cliente*, *Servidor* e *ClienteSub*, tal que $Cliente, Servidor, ClienteSub \in \mathcal{P}$, estão executando em paralelo por meio do uso do operador de composição paralela $|$. O pacote PUBLISH contido na porta de entrada de $Cliente()$, indica que o sistema tem como ponto de partida a entrada do pacote, que substituirá o nome z, de forma que $Cliente\{Publish/z\}$.

O processo *Cliente* define que a mensagem contida no nome z deverá ser enviada pela porta de saída $\overline{c}$ do canal de comunicação c. O *Servidor*, por sua vez, é encarregado de receber a mensagem pela porta de entrada do canal c, e em seguida, disponibilizá-la no canal de saída $\overline{pub}$ para que assim, seja recebida pelos clientes previamente assinados no mesmo tópico no qual o pacote PUBLISH é destinado.

Percebe-se que no modelo 4.1, diferente do que foi apresentado na Seção 3.1, há a adição do processo *ClienteSub* (cliente *subscriber*), que representa o cliente assinado no tópico em que o pacote PUBLISH é direcionado. Portanto, o *ClienteSub*

é definido pela porta de entrada *pub*, por onde o pacote será recebido, de forma que $ClienteSub\{Publish/y\}$. Vale ressaltar que não há um limite de assinantes para o mesmo tópico, logo, vários processos podem estar vinculados ao mesmo tópico, sendo que, o processo *ClienteSub* foi adicionado no intuito de auxiliar a compreensão do processo como um todo, uma vez que não está previsto na especificação do protocolo.

Percebe-se que, durante o processo de comunicação, as mensagens representadas pelos nomes $z, x, y \in \mathcal{N}$ sempre serão as mesmas.

O modelo considera um ambiente onde o processo *Cliente* encontra-se apto para se comunicar com o servidor. Por conta disso, é ocultada a operação obrigatória de conexão entre o *Cliente* e o *Servidor*, em que o *Cliente* envia um pacote CONNECT requisitando a conexão, e a mesma é estabelecida assim que obtêm um pacote CONNACK como resposta do *Servidor* [6]. Esse processo pode ser modelado como:

$$\begin{aligned} &Cliente(Connect) \mid Servidor(),\ em\ que: \\ &Cliente(z) \stackrel{def}{=} \overline{c}\langle z\rangle.c'(y) \\ &Servidor() \stackrel{def}{=} c(x).\overline{c'}\langle Connack\rangle \end{aligned} \tag{4.2}$$

Outra operação que deve ficar subentendida é a assinatura, quando um cliente se inscreve num determinado tópico com a intensão de receber as mensagens publicadas posteriormente neste tópico. De acordo com a especificação do MQTT [6], o cliente deve enviar ao servidor um pacote SUBSCRIBE, e o servidor, por sua vez, recebe a mensagem e responde o cliente com um pacote SUBACK. Por fim, o cliente recebe a resposta e está apto a receber mensagens que correspondam ao tópico no qual está registrado. O modelo em TPi desta operação é apresentado abaixo.

$$\begin{aligned} &ClienteSub(Subscribe) \mid Servidor(),\ em\ que: \\ &Cliente(z) \stackrel{def}{=} \overline{c}\langle z\rangle.c'(y) \\ &Servidor() \stackrel{def}{=} c(x).\overline{c'}\langle Suback\rangle \end{aligned} \tag{4.3}$$

4.2 Transformando o modelo do MQTT com QoS 0 em um contrato

Com base nas observações feitas na Seção anterior, nota-se que os processos *Cliente*, *Servidor* e *ClienteSub*, atuam como agentes definidos pelas operações $\overline{c}$, $c(x).\overline{pub}\langle x\rangle$ e $pub(y)$, respectivamente, as quais apontam ações de entrada ou saída de mensagens. Diante disso, as definições dos processos podem ser consideradas um conjunto de ações a serem executadas para que o modelo do sistema seja satisfeito. Portanto, um modelo em TPi pode representar um compromisso definido por ações a serem realizadas pelos processos, no caso do modelo 4.1, expressa o compromisso de entregar um pacote PUBLISH com

QoS= 0 aos assinantes. Assim sendo, de acordo com o modelo de contrato multilateral definido por Xu[27], podemos nomear o compromisso como

$$C\_QoS0delivery,$$

em que "$C\_$" é um prefixo usado para identificação de compromissos.

O conjunto dos agentes é definido por $\mathcal{P} = \{C, S, CS\}$, no qual C é o cliente, S, o servidor, e CS, o cliente *subscriber*.

Uma ação é um quadrupla $a = name, sender, receiver, deadline$, no qual:

- $name \in ID$ é o identificador da ação;
- $sender, receiver \in \mathbb{R}$ são os papéis;
- $deadline \in \mathbb{T}$ é uma representação simplificada do tempo (dias, horas, segundos, etc).

Posto isto, como as mensagens representadas pelos nomes $z, x, y \in \mathcal{N}$ serão sempre as mesmas, a partir das definições $\{(\overline{c}\langle z\rangle), (c(x).\overline{pub}(x)), (pub(y))\}$, são apontadas as seguintes ações:

- $\overline{c}\langle z\rangle$: O cliente C envia um pacote PUBLISH ao servidor S;

$$(A\_QoS0enviarPublish, C, S, ?)$$

- $c(x)$: O servidor S recebe o pacote PUBLISH;

$$(A\_QoS0receberPublish, S, S, ?)$$

- $\overline{pub}(x)$: O servidor S disponibiliza o pacote PUBLISH para os assinantes interessados;

$$(A\_QoS0publicarPacote, S, CS^n, ?)$$

- $pub(y)$: O cliente assinante CS recebe o pacote PUBLISH.

$$(A\_QoS0receberPacote, CS, CS, ?)$$

Desta forma, o conjunto das ações é descrito como

$$\{(A\_QoS0enviarPublish, C, S, ?), (A\_QoS0receberPublish, S, S, ?),$$

$$(A\_QoS0publicarPacote, S, CS^n, ?), (A\_QoS0receberPacote, CS, CS, ?)\}.$$

Como não há nenhum critério temporal envolvido no envio de pacotes com QoS= 0, o campo *deadline* das ações são atribuídos com o símbolo "?". Assim, subentende-se que ficará a critério da aplicação de IoT definir a representação temporal adequada.

Visto que são detectadas apenas ações relacionadas com envio ou recebimento de mensagens, em que um envio é finalizado com um recebimento, então pode-se definir que ações de envio são do tipo *trigger* e as ações de recebimento são do tipo $finish$, não havendo nada envolvido entre as mesmas.

Um compromisso pode ser denotado pela quíntupla

$$commitment = (name, sender, receiver, n, \{(a_1, u_1), (a_2, u_2), ..., (a_n, u_n) : a_i \in \mathbb{A}, u_i \in \mathbb{U}\}).$$

Em que:

- *name* é o identificador único do compromisso;
- *sender* é a parte do contrato que atua como expedidor do compromisso;
- *receiver* é a parte do contrato que atua como receptor do compromisso;
- n representa o número total de ações envolvidas, tal que $n \in \mathbb{N}$;
- $(a_1, a_2, ..., a_n)$ *e* $(u_1, u_2, ..., u_n)$ representam o par ordenado das ações e seus atributos.

Sendo assim, o compromisso $C\_QoS0delivery$ é constituído como segue:

$$\begin{aligned} c = \{C\_QoS0delivery, C, CS, 4, \{ \quad & (A\_QoS0enviarPublish, tr) \\ & (A\_QoS0receberPublish, fi) \\ & (A\_QoS0publicarPacote, tr) \\ & (A\_QoS0receberPacote, fi)\}\} \end{aligned} \tag{4.4}$$

Uma vez que o compromisso está devidamente formalizado, a tabela de compromissos pode ser obtida, como apresentado na Tabela 9, sendo que, a coluna referente as ações do tipo *involve* é ocultada, já que não é utilizada.

Tabela 9 – Tabela de comprometimento $C\_QoS0delivery$.

Compromissos	**Classificação das ações e compromissos**		**Rótulos**
	tr	fi	
$C\_QoS0delivery$ ($Q0$)	$A\_QoS0enviarPublish$		$Q0, 1$
		$A\_QoS0receberPublish$	$Q0, 2$
	$A\_QoS0publicarPacote$		$Q0, 3$
		$A\_QoS0receberPacote$	$Q0, 4$

A partir da Tabela 9, o grafo de comprometimento pode ser obtido, conforme ilustrado na Figura 28.

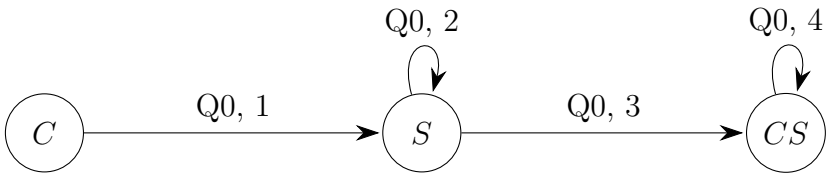

Figura 28 – Grafo de compromisso $C\_QoS0delivery$.

Fonte: Próprio autor.

4.3 Algoritmo de conversão de modelo: TPi para contrato multilateral

Com base no que foi feito na Seção anterior, a conversão do modelo em TPi para um contrato multilateral é realizada, primeiramente, seguindo um conjunto de 6 passos, são eles:

1. Nomear o compromisso. Por exemplo, $c = C\_QoS0delivery$.

2. Identificar os agentes envolvidos. Para cada processo em $P_1 \mid P_2 \mid ... \mid P_n$, compõe o conjunto de agentes $\mathcal{P}$, tal que $P_1, P_2, ..., P_n \in \mathcal{P}$;

3. Identificar a ação que dá início ao sistema, esse passo consiste em duas tarefas:

   - Localizar o primeiro processo do conjunto $\mathcal{P}$, P_1 por exemplo. O processo é denotado por Pi;
   - Identificar a mensagem que dá início ao sistema, situada entre parênteses após o ID do processo inicial descrito em TPi. A mensagem inicial é denotada por Mi. Dado $P_1(msg)$, então $Mi = msg$.

4. Agrupar em duplas os processos e suas respectivas definições, formando o conjunto $\mathcal{D}$, tal que
$$\mathcal{D} = \{(P_1, (def_1), (P_2, (def_2), \ldots, (P_n, (def_n))\}.$$

5. Trilhar a sequência de troca de mensagens. Toda ação de entrada e saída em TPi, como $\overline{y}\langle x\rangle.P$ e $y(x).P$, por exemplo, pode ser seguida ou não de um ponto ".", que indica uma nova ação a ser tomada após a execução da ação antecedente. Portanto, é possível traçar o caminho ordenado da troca de mensagens, da primeira à última ação, que são armazenadas no conjunto $\mathcal{A}$ seguidas de seu processo correspondente, formando uma dupla (a_n, P_m). Para isso, o algoritmo percorre as ações partindo da primeira ação do processo inicial P_1, e então:

   - Para cada ação de saída $\overline{y}\langle x\rangle$, a mesma é alocada em $\mathcal{A}$. Em seguida, deve ser localizada uma ação de entrada $y(n)$, em que y, x e n representam nomes abstratos tal que $y, x, n \in \mathcal{N}$, e o nome y deve ser igual em ambas as ações. É

importante frisar que, uma porta de entrada de uma ação pode estar presente em outras ações, assim, todas irão receber a mesma mensagem;

- Para cada ação de entrada localizada seguida de ".", a ação é armazenada em $\mathcal{A}$ e o algoritmo segue para a ação posterior ao ponto.

6. O algoritmo encerra quando todas as ações são armazenadas em $\mathcal{A}$.

Após a aplicação do algoritmo, o sistema é representado pela quíntupla

$$s = (name, P_{ini}, \mathcal{P}, \mathcal{D} = \{(P_1, \{a_1, ..., a_n\}), ..., (P_m, \{a_1, ..., a_n\})\}, \mathcal{A} = \{(a_1, P_1), (a_2, P_1)..., (a_n, P_m)\}).$$

Sendo que:

- *name*: É o identificador único do compromisso;
- P_{ini}: Representa o processo inicial e a mensagem carregada pelo mesmo, denotado como (Pi, Mi), tal que $Pi \in \mathcal{P}$;
- $\mathcal{P}$: Conjunto dos processos $\{P_1, P_2, ..., P_n\}$;
- $\mathcal{D}$: Conjunto dos processos e das ações que formam suas definições, tal que $P_1, ..., P_m \in \mathcal{P}$ e $a_1, ..., a_n \in \mathbb{A}$;
- $\mathcal{A}$: Conjunto das ações dispostas de acordo com sua ordem de execução no sistema, seguidas de seus respectivos processos.

Vale ressaltar que, para que o algoritmo funcione corretamente, o primeiro processo descrito no modelo em TPi deve ser sempre aquele que dá início ao sistema.

Baseado nas propriedades do processo presentes na quíntupla s, o contrato pode ser, por fim, formalizado. Para definir as ações, basta procurá-las percorrendo as definições dos processos contidos no conjunto $\mathcal{D}$. Para cada ação de saída, o agente (ou processo) expedidor da ação recebe o papel de *sender*, e as ações que contenham a porta de entrada correspondente assumem o papel de *receiver*. A ação de entrada tem como *sender* e *receiver* o próprio processo no qual é definida. Quando uma ação de saída não possui uma ação de entrada correspondente no sistema, então o *receiver* é considerado como um cliente abstrato C_{sub}. Se uma ação de saída possui duas ou mais ações de entrada correspondentes, então estas são transformadas em n ações com o mesmo *sender* e diferentes $receivers$, em que n é o número de ações de entrada. Desta forma, cada *receiver* é definido de acordo com o processo no qual a ação de entrada correspondente está contida.

Definir o *name* do compromisso é o primeiro passo para sua formalização. O *name* do compromisso deve ser idêntico ao *name* do sistema s. Os campos *sender* e o *receiver* são definidos pelo processo inicial contido em P_{ini} e pelo processo contido no último

elemento de $\mathcal{A}$, respectivamente. No caso de uma ação de saída estar presente no último elemento de $\mathcal{A}$, então o *receiver* é denotado como um cliente abstrato C_{sub}. As ações dos pares ordenados de ações e atributos do compromisso são atribuídos na mesma ordem que as ações presentes nos pares do conjunto $\mathcal{A}$. Já os atributos são declarados de acordo com o tipo de ação. As ações de saída e entrada recebem os atributos $trigger$ e $finish$, respectivamente, exceto quando uma ação de saída é o último elemento de $\mathcal{A}$, a qual deve ser atribuída como $finish$.

Uma vez que os compromissos estão formalizados, o grafo de compromisso está pronto para ser gerado. Contudo, resta definir os papéis dos agentes e as ordens de execução dos compromissos, sendo que, estas propriedades devem ser definidas manualmente em razão de não se ter encontrado um meio para representá-las no modelo em TPi. Assim que os papéis e as ordens de execução são definidas, o contrato possui todos os elementos necessários para sua formalização.

4.4 Algoritmo de monitoramento reativo

No estágio de execução do contrato, identificar o responsável por uma violação é a tarefa mais importante num processo de monitoramento [28]. A abordagem usada neste trabalho foi proposta por Xu, Jeusfeld e Grefen[28], e consiste no uso do grafo de compromisso e propriedades dos papéis desempenhados pelos agentes para detectar as partes responsáveis pela violação contratual.

Durante a execução de um contrato, um monitor do contrato constata que há um conjunto de ações que quando executadas, impedem a realização de outra determinada ação. Esta ação é denotada como a_{miss}, e a última ação realizada até então, é definida como a_{done}. A partir das ações a_{miss}, a_{done} e suas posições no grafo de compromisso, o grafo pode ser restruturado [28]. O processo de restruturação é descrito pelo Algoritmo 1.

Primeiramente, todas as ações que seriam executadas após a_{miss} são localizadas. A princípio, os compromissos no qual a_{miss} está envolvido devem ser evidenciados, os mesmos são denotados por $M_{current}$. A partir do compromisso $m \in M_{current}$, as posições das ações $(\exists a', f_{position}(a', m))$ estão adiante da posição de a_{miss}. As ações não executadas são contidas no conjunto E_{not_occur}, e caso estejam envolvidas em outros compromissos que contenham ações posteriores à a_{miss}, estas também devem ser incluídas em E_{not_occour}. Compromissos posicionados antes de $M_{current}$ são adicionados em $M_{not\_trigger}$, e suas ações são inseridas em E_{not_occur}.

Em seguida, ações anteriores à a_{done} são executadas. Cada ocorrência de a_{done} em cada compromisso de $M_{current}$ é identificada. Com base no compromisso $m \in M_{current}$, as ações posteriores à a_{done} posicionadas em $(\exists a', f_{position}(a', m))$ são inseridas em E_{done}. Caso estas ações estejam inclusas em compromissos que possuem ações anteriores à a_{miss},

então estas também devem ser alocadas em E_{done}.

Por fim, o conjunto $E_{not\_occur \wedge E_{done}}$ representa as ações que devem ser simplificadas. A função $RestructureCG(E)$ realiza a simplificação, resultando no conjunto reduzido de arestas E_{cut}. Vértices que não possuírem nenhuma conexão também devem ser eliminados.

Com base no grafo de compromisso restruturado, é checado se o papel expedidor de a_{miss} teve suas entradas recebidas. O processo de detecção das partes responsáveis por violações contratuais segue os seguintes passos:

- A parte do contrato que tem um papel em execução verifica a violação baseada nas entradas das propriedades do papel;
- Se a violação estiver presente na entrada do papel, então as saídas de outros papéis correspondentes à entrada violada devem ser localizadas;
- A ação que produz a saída que gera a violação é identificada;
- Se a ação ainda não ocorreu e a condição para sua ocorrência é satisfeita, então o expedidor da ação é o responsável;
- Os passos acima são repetidos até que o responsável seja identificado.

Assim que as propriedades dos compromissos estão devidamente formalizadas, o algoritmo de monitoramento reativo proposto por Xu[27] pode ser aplicado, possibilitando desta forma, a detecção de responsáveis por violações contratuais. A descrição do mecanismo de monitoramento é reproduzido pelo Algoritmo 2, localizado na Seção 3.2.

5 ESTUDO DE CASO: SISTEMA PARA DETECÇÃO E SUPRESSÃO DE INCÊNDIO

O estudo de caso proposto é baseado no trabalho desenvolvido por Kang et al.[12], no qual um servidor MQTT para uma aplicação de IoT foi construído utilizando recursos do *Amazon Web Services* (AWS). A aplicação consiste na implementação de um sistema que simula o cenário de um ambiente inteligente. O ambiente conta com um sistema para controle de temperatura e outro para detecção de supressão de incêndio. Para o presente estudo, será considerado apenas o módulo de detecção e supressão de incêndio [12].

O sistema conta com quatro elementos principais: um servidor MQTT, chamado de *Broker*; um aplicativo móvel; um detector de incêndio; e um chuveiro de combate à incêndios, também chamado de *sprinkler*. No cenário descrito por Kang et al.[12], o usuário pode checar o status do *sprinkler* (ligado/desligado) por meio do aplicativo, o qual emite uma mensagem de requisição de status para o *sprinkler*, que responde logo em seguida. Quando as chamas são detectadas pelo sensor de incêndio, uma mensagem de alerta é enviada para o aplicativo e para o *sprinker*, que é imediatamente acionado. Se, por alguma razão, o *sprinkler* não for ativado, o usuário é notificado, e pode acioná-lo pelo aplicativo.

Na Tabela 10 é apresentado o conjunto de mensagens trocadas durante a execução do sistema, de acordo com a estratégia *publish/subscribe* do protocolo MQTT. Para cada mensagem, um tópico de assinatura é fornecido, assim como o *publisher* e o *subscriber* envolvidos, isto é, aquele que envia e aquele que recebe a mensagem. Para cada mensagem entregue, subentende-se que a mesma está contida num pacote PUBLISH. Por fim, após cada mensagem recebida, uma ação é efetuada pelo *subscriber*.

Tabela 10 – Protocolo de troca de mensagens para a aplicação IoT sobre o MQTT [12], adaptada pelo autor.

Mensagem	**Tópico MQTT**	***Publisher***	***Subscriber***	**Ação do *subscriber***
Fire Detection	*Fire/Detected*	Sensor de incêndio	*App*	Exibir notificação
Sprinkler Request	*Sprinkler/StReq*	*App*	*Sprinkler*	Ler status
Sprinkler Reply	*Sprinkler/StRep*	*Sprinkler*	*App*	Enviar mensagem
Sprinkler Start	*Sprinkler/Start*	*App*	*Sprinkler*	Ativar *sprinkler*
Sprinkler Start	*Sprinkler/Start*	Sensor de incêndio	*Sprinkler*	Ativar *sprinkler*

É importante ressalvar que, para receber uma mensagem, os agentes envolvidos devem estar previamente registrados no tópico correspondente. Esta característica do protocolo MQTT fica evidente na Figura 29, na qual é apresentado um diagrama de sequência da troca de mensagens que representa o comportamento operacional do sistema.

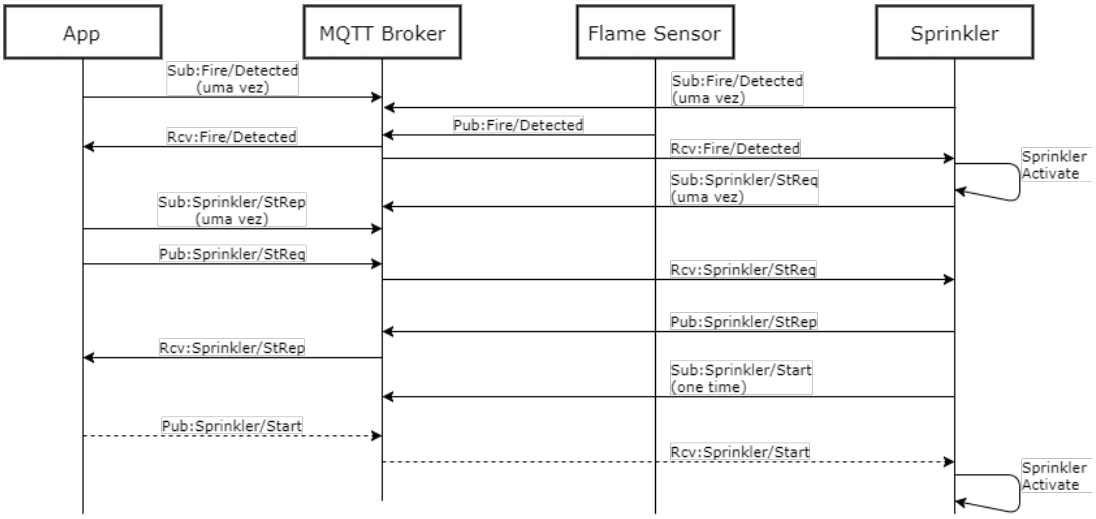

Figura 29 – Procedimento de troca de mensagens do sistema IoT [12], adaptado pelo autor.

Alguns elementos foram acrescentados na Figura 29 e na Tabela 10, com a finalidade de deixá-las de acordo com as especificações do projeto. Ao descrever o cenário do aplicação, Kang et al.[12] relatam que: "*When the flame sensor detects the fire, the IoT device automatically alarms the fire event, activates the sprinkler to suppress the fire, and sends fire alarm message to smartphone app via IoT platform*".

Observa-se que, os autores não deixam claro que o disparo do evento de ativação do *sprinkler* acontece mediante o recebimento de uma mensagem, que no caso, deve ser enviada pelo sensor de incêndio sob o tópico *"Fire/Detected"*.

Tanto na descrição do cenário quanto nos esquemas apresentados na Figura 29 e na Tabela 10, o envio da mensagem de alerta de incêndio do sensor para o *sprinkler* não é especificado. Sendo assim, foram adicionados procedimentos para assinatura do tópico "*Fire/Detected*" e recebimento do alerta por meio do tópico na Figura 29, e a descrição do envio da mensagem na Tabela 10.

5.1 Aplicando o algoritmo de conversão

Com base na descrição e objetivos do sistema, três processos que caracterizam seus compromissos são identificados: Detecção/Alerta; Checagem; e Supressão. Adiante, cada processo é modelado por meio da TPi de acordo com o comportamento operacional do MQTT quando QoS= 0 e convertido em contrato multilateral por meio do algoritmo descrito na Seção 4.3. Por fim, o grafo de compromissos do sistema completo é obtido.

Primeiramente, os papéis dos agentes envolvidos no sistema devem ser identificados. O único processo (ou parte) identificado por mais de um papel é o processo *App*, que representa a aplicação móvel. O *App* assume o papel *Ap* no procedimento de checagem e

papel Ap' no mecanismo de supressão. Já os processos *Sensor*, *Broker* e *Sprinkler* têm seus papéis representados por S, Br e Sp, respectivamente.

Processo 1: Detecção/Alerta

O procedimento de detecção e alerta de incêndio requer o envolvimento do sensor de incêndio, do *Broker*, do aplicativo e do *sprinkler*. Seu modelo em TPi é descrito como segue:

$$
\begin{aligned}
&Sensor(FireDetection) \mid Broker() \mid App() \mid Sprinkler(),\ em\ que: \\
&Sensor(z) \stackrel{def}{=} \overline{fd}\langle z\rangle \\
&Broker() \stackrel{def}{=} fd(x).\overline{fd'}\langle x\rangle \\
&App() \stackrel{def}{=} fd'(y) \\
&Sprinkler() \stackrel{def}{=} fd'(y')
\end{aligned} \tag{5.1}
$$

O sistema inicia assim que a mensagem "*FireDetection*" está pronta para ser enviada. Adiante, a mensagem é entregue ao *Broker*, que por sua vez, a disponibiliza no canal de saída $\overline{fd'}$, e, por fim, é captada pelos processos *App* e *Sprinkler*. Se nenhum fator externo impedir a entrega da mensagem ao *Sprinkler*, então o dispositivo é ativado sem a necessidade do processo de Supressão.

Aplicação do algoritmo

O resultado obtido após a aplicação de cada passo do algoritmo é descrito como segue:

1. $name = C\_detectarIncendio$;
2. $P_{ini} = (Sensor, FireDetection)$;
3. $\mathcal{P} = \{Sensor, Broker, App, Sprinkler\}$;
4. $\mathcal{D} = \{(Sensor, \{\overline{fd}\langle z\rangle\}), (Broker, \{fd(x).\overline{fd'}\langle x\rangle\}), (App, \{fd'(y)\}), (Sprinkler, \{fd'(y')\})\}$
5. $\mathcal{A} = \{(\overline{fd}\langle z\rangle, Sensor), (fd(x), Broker), (\overline{fd'}\langle x\rangle, Broker), (fd'(y), App), (fd'(y'), Sprinkler)\}$

De acordo com as regras estabelecidas na Seção 4.3, o conjunto de ações é definido

a seguir:

$$\{(\overline{fd}\langle z\rangle, S, Br, ?),$$
$$(fd(x), Br, Br, ?),$$
$$(\overline{fd'}\langle x\rangle, Br, Ap, ?),$$
$$(fd'(y), Ap, Ap, ?),$$
$$(\overline{fd'2}\langle x\rangle, Br, Sp, ?),$$
$$(fd'2(y'), Sp, Sp, ?)\}$$

O compromisso é declarado como:

$$m_1 = (C\_detectarIncendio, Sensor, Sprinkler, 6, \{ \quad (\overline{fd}\langle z\rangle, tr), (fd(x), fi), (\overline{fd'}\langle x\rangle, tr), (fd'(y), fi), (\overline{fd'2}\langle x\rangle, tr), (fd'2(y'), fi)\ \}) \quad (5.2)$$

Processo 2: Checagem

O processo de checagem do status do *sprinkler* requer o envolvimento do aplicativo, do *Broker* e do *sprinkler*. Seu modelo em TPi é descrito a seguir:

$$App(SprinklerRequest) \mid Broker() \mid Sprinkler(),\ em\ que:$$
$$App(z) \stackrel{def}{=} \overline{req}\langle z\rangle.rep'(y)$$
$$Broker() \stackrel{def}{=} req(x).\overline{req'}\langle x\rangle.rep(x').\overline{rep'}\langle x'\rangle$$
$$Sprinkler() \stackrel{def}{=} req'(v).\overline{rep}\langle reqReply\rangle \quad (5.3)$$

Assim que a mensagem "*SprinklerRequest*" encontra-se pronta, o *App* a envia para o *Broker*, que então é direcionada para a porta de saída $\overline{req'}$. Na sequência, o *Sprinkler* recebe a mensagem de requisição e lança a resposta "*reqReply*" no canal de saída $\overline{rep}$. Por fim, o *Broker* recebe a resposta e a divulga no canal $\overline{req'}$, que é recebida pelo *App*, encerrando assim o ciclo de interação do sistema.

Aplicação do algoritmo

O resultado obtido após a aplicação do algoritmo é exposto a seguir:

1. $name = C\_detectarIncendio$;
2. $P_{ini} = (App, SprinklerRequest)$;
3. $\mathcal{P} = \{App, Broker, Sprinkler\}$

4. $\mathcal{D} = \{(App, \{\overline{req}\langle z\rangle.rep'(y)\}), (Broker, \{req(x).\overline{req'}\langle x\rangle.rep(x').\overline{rep'}\langle x'\rangle\}),$
   $(Sprinkler, \{req'(v).\overline{rep}\langle reqReply\rangle\})\}$;

5. $\mathcal{A} = \{(\overline{req}\langle z\rangle, App), (req(x), Broker), (\overline{req'}\langle x\rangle, Broker), (req'(v), Sprinkler),$
   $(\overline{rep}\langle reqReply\rangle, Sprinkler), (rep(x'), Broker), (\overline{rep'}\langle x'\rangle, Broker), (rep'(y), App)\}$.

O conjunto de ações é representado por:

$$\begin{aligned}
&\{(\overline{req}\langle z\rangle, Ap, Br, ?),\\
&(rep'(y), Ap, Ap, ?),\\
&(req(x), Br, Br, ?),\\
&(\overline{req'}\langle x\rangle, Br, Sp, ?),\\
&(rep(x'), Br, Br, ?),\\
&(\overline{rep'}\langle x'\rangle, Br, Ap, ?),\\
&(req'(v), Sp, Sp, ?),\\
&(\overline{rep}\langle reqReply\rangle, Sp, Br, ?)\}
\end{aligned}$$

Deste modo, o compromisso é definido como:

$$\begin{aligned}
m_2 = (C_checarStatus, Ap, Ap, 8, \{ \quad & (\overline{req}\langle z\rangle, tr), (req(x), fi), (\overline{req'}\langle x\rangle, tr),\\
& (req'(v), fi), (\overline{rep}\langle reqReply\rangle, tr), (rep(x'), fi),\\
& (\overline{rep'}\langle x'\rangle, tr), (rep'(y), fi)\ \}) \qquad (5.4)
\end{aligned}$$

Processo 3: Supressão

O processo de supressão de incêndio é executado em caso de não acionamento do *Sprinkler* pelo processo de Detecção/Alerta, e requer o envolvimento do App, do Broker e do Sprinkler. Seu modelo em TPi é descrito abaixo:

$$\begin{aligned}
&App(StartSprinkelr) \mid Broker() \mid Sprinkler(),\ em\ que:\\
&App(z) \stackrel{def}{=} \overline{st}\langle z\rangle\\
&Broker \stackrel{def}{=} st(x).\overline{st'}\langle x\rangle\\
&Sprinkler() \stackrel{def}{=} st'(y) \qquad (5.5)
\end{aligned}$$

Com a mensagem "*StartSprinkler*" pronta para entrega, o *App* a envia para o *Broker*, que a recebe e disponibiliza no canal $\overline{st'}$. Enfim, a mensagem de ativação é receptada pelo *Sprinkler* e o ciclo de interação do sistema é encerrado.

Aplicação do algoritmo

O resultado atingido após a aplicação do algoritmo é descrito a seguir:

1. $name = C\_suprirIncendio$;
2. $P_{ini} = (Sensor, FireDetection)$;
3. $\mathcal{P} = \{Sensor, Broker, App, Sprinkler\}$;
4. $\mathcal{D} = \{(App, \{\overline{st}\langle z\rangle\}), (Broker, \{st(x).\overline{st'}\langle x\rangle\}), (Sprinkler, \{st'(y)\})\}$;
5. $\mathcal{A} = \{(\overline{st}\langle z\rangle, App), (st(x), Broker), (\overline{st'}\langle x\rangle, Broker), (st'(y), App)$

O conjunto de ações é representado por:

$$\{(\overline{st}\langle z\rangle, Ap', Br, ?),$$
$$(st(x), Br, Br, ?),$$
$$(\overline{st'}\langle x\rangle, Br, Ap', ?),$$
$$(st(x'), Br, Br, ?)\}$$

O compromisso é definido como segue:

$$m_3 = (C\_suprirIncendio, Sensor, Sprinkler, 4, \{ \quad (\overline{st}\langle z\rangle, tr), (st(x), fi), \quad (\overline{st'}\langle x\rangle, tr)(st'(y), fi) \ \}) \tag{5.6}$$

5.2 Grafo de compromisso do sistema

Dado os compromissos m_1, m_2, e m_3, o grafo de compromissos pode ser criado. Na Tabela 11 e na Figura 30 são apresentados a tabela e o grafo de compromissos do sistema, respectivamente. Para melhor compreensão, o nome das ações foi alterado, de forma a deixá-las mais intuitivamente interpretáveis, além de acrescentar o prefixo "$A\_$", utilizado para identificar as ações.

As arestas do grafo são determinadas pelas ações, representadas no grafo de acordo com os rótulos atribuídos na Tabela 11, enquanto que os papéis desempenhados pelos agentes definem os vértices.

Tabela 11 – Tabela de compromisso do sistema de detecção e supressão de incêndio

Compromisso	**Classificação das ações e compromissos**		**Rótulo**
	tr	*fi*	
C_detectarIncendio (DI)	A_publicarAlerta		DI, 1
		A_receberAlerta	DI, 2
	A_disponibilizarAlerta		DI, 3
		A_receberPubAlerta	DI, 4
	A_disponibilizarAlerta2		DI, 5
		A_receberPubAlerta2	DI, 6
C_checarStatusSprinkler (CS)	A_requisitarStatus		CS, 1
		A_receberRequisicao	CS, 2
	A_disponibilizarRequisicao		CS, 3
		A_receberPubRequisicao	CS, 4
	A_publicarStatus		CS, 5
		A_receberStatus	CS, 6
	A_disponibilizarStatus		CS, 7
		A_receberPubStatus	CS, 8
C_suprirIncencio (SI)	A_publicarAtivacao		SI, 1
		A_receberAtivacao	SI, 2
	A_disponibilizarAtivacao		SI, 3
		A_receberPubAtiv	SI, 4

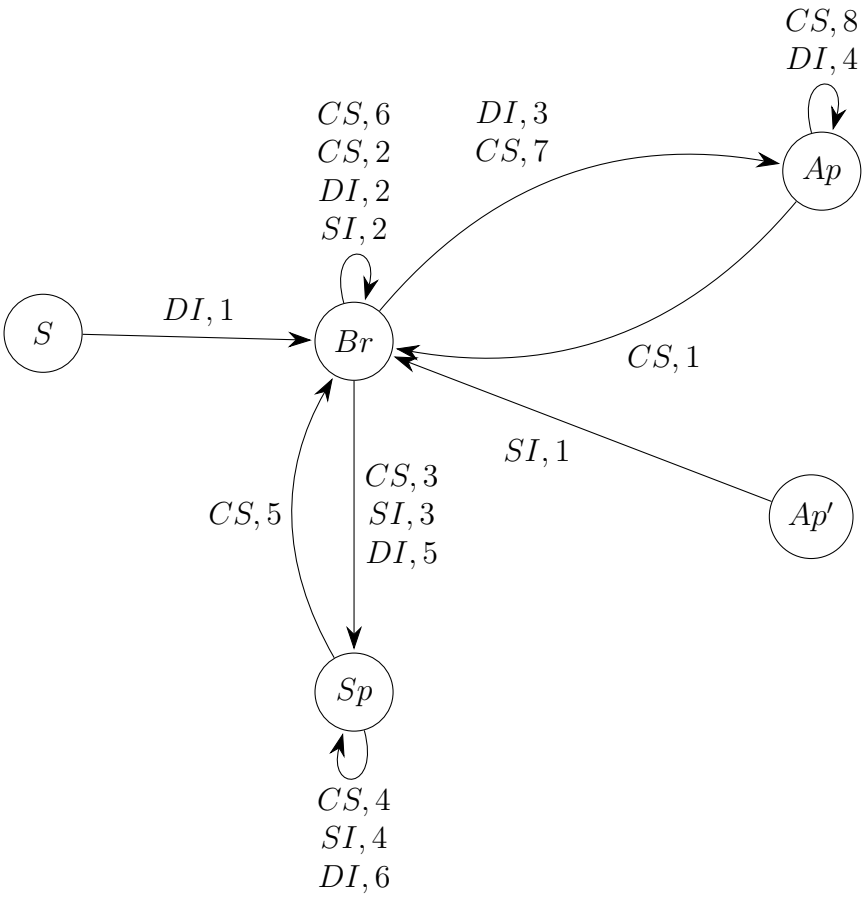

Figura 30 – Grafo de compromisso do sistema de detecção e supressão de incêndio.

Fonte: Próprio autor.

O próximo passo para a formalização do contrato consiste em definir as ordens de compromisso. Como mencionado anteriormente, o compromisso de detecção é acionado assim que o sensor de incêndio detecta a presença de chamas. Na sequência, o compromisso de checagem é executado para que o aplicativo confira o status do *splinkler*. Caso o *sprinkler* não esteja ativado após a detecção, então o compromisso de supressão é acionado. O compromisso de checagem, por sua vez, pode ser requisitado a qualquer momento. Considerando os compromissos m_1, m_2 e m_3 (Detecção/Alerta, Checagem e Supressão), o conjunto de todas as ordens de compromisso é definido a seguir:

$$ordercommitment = \{(m_2), (m_1 \cdot m_2), (m_1 \cdot m_2 \cdot m_3)\} \tag{5.7}$$

5.3 Aplicação do algoritmo de monitoramento reativo

No cenário escolhido para a aplicação do monitoramento, o sensor de incêndio detecta a presença de chamas no local e envia a mensagem de alerta para o aplicativo e o *sprinkler*. Em razão de alguma interferência, a mensagem de ativação não foi entregue ao *sprinkler*. Ao realizar a checagem, o aplicativo notificou ao usuário que o *sprinkler* continuava desativado mesmo com a presença de chamas. O usuário então aciona o mecanismo de ativação via aplicativo, dando início ao compromisso de supressão de incêndio. Porém, devido a uma falha interna do aplicativo, a mensagem novamente não foi entregue ao *sprinkler*, ou teve seus dados corrompidos.

Sendo assim, a ação perdida é definida como a_{miss} = A_receberAtivacao $(SI, 2)$, a qual é realizada por meio do papel Br, e a última mensagem executada é declarada como a_{done} =A_publicarAtivacao $(SI, 1)$, exercida pelo papel Ap'. Em seguida, é realizada a aplicação do algoritmo de restruturação do grafo de compromisso, descrito na Sessão 3.3.

Algoritmo 3: Aplicando a simplificação do grafo de compromisso

Entrada: $Contract = \{\mathbb{A}, \mathbb{M}, G\}$,
Ação perdida: a_{miss} = A_receberAtivacao (SI_2)
Ação feita: a_{done} = A_publicarAtivacao (SI_1)

begin
- ▷ /*Primeiro, encontrar ações que ainda não ocorreram*/
  $M_{current} = \{SI : edge(\text{A\_receberAtivacao } (SI_2))\}$;
- $M_{not\_triggered} = \emptyset$;
- PARA cada ação em $M_{current}$ posteriores à a_{miss}, então inserir
  A_disponibilizarAtivacao (SI_3) e A_receberPubAtiv (SI_4) em E_{not_occur};
- $E_{not\_occur} = \{SI_3, SI_4\}$;
- ▷ /*Segundo, obter ações já executadas*/
- $E_{done} = \emptyset$;
- E_{cut} =
  $\{DI_1, DI_2, DI_3, DI_4, DI_5, DI_6, CS_1, CS_2, CS_3, CS_4, CS_5, CS_6, CS_7, CS_8\}$;

end

Após o acionamento do Algoritmo 1, descrito na Sessão 4.3, o novo grafo de compromisso é obtido, como mostra a Figura 31, em que a parte destacada em vermelho e verde indicam a ocorrência das ações a_{miss} e a_{done}, respectivamente. A computação do Algoritmo 1 é reproduzida pelo Algoritmo 3.

Na Tabela 12 são expostas as propriedades dos papéis desempenhados pelas partes envolvidas, necessárias para apontar os responsáveis pela violação.

Tabela 12 – Papéis e suas propriedades

Papel	**Propriedades do papel**		
	Entrada	Saída	Regras
Ap	Requisição interna, *RequestReply*	*SprinklerRequest*	CS,7 → CS,8; DI,3 → DI,4; CS,1
Ap'	Sinal externo	*StartSprinkler*	SI,1
S	Sinal externo	*FireDetection*	DI,1
Br	*SprinklerRequest, FiraDetection, StartSprinkler, RequestReply*	*SprinklerRequest, FiraDetection, StartSprinkler, RequestReply*	CS,5 → CS,6; CS,1 → CS,2; DI,1 → DI,2; SI,1 → SI,2
Sp	*SprinklerRequest, FiraDetection, StartSprinkler*	*RequestReply*	CS,3 → CS,4; SI,3 → SI,4; DI,5 → DI,6; CS,5

Responsável por enviar a mensagem de ativação "*StartSprinkler*" ao *sprinkler*, o papel Ap' (o aplicativo móvel) é acionado. Assim que o papel Br receber a mensagem, o *broker* deve verificar se os dados da mensagem estão de acordo com o padrão estabelecido. Se a resposta for negativa, então Ap' é o responsável. Caso contrário, a mensagem direcionada por Br para o papel Sp é verificada, e se estiver de acordo, o responsável é o Br. Por outro lado, se a integridade da mensagem for mantida até seu direcionamento para o Sp, então Sp é o responsável.

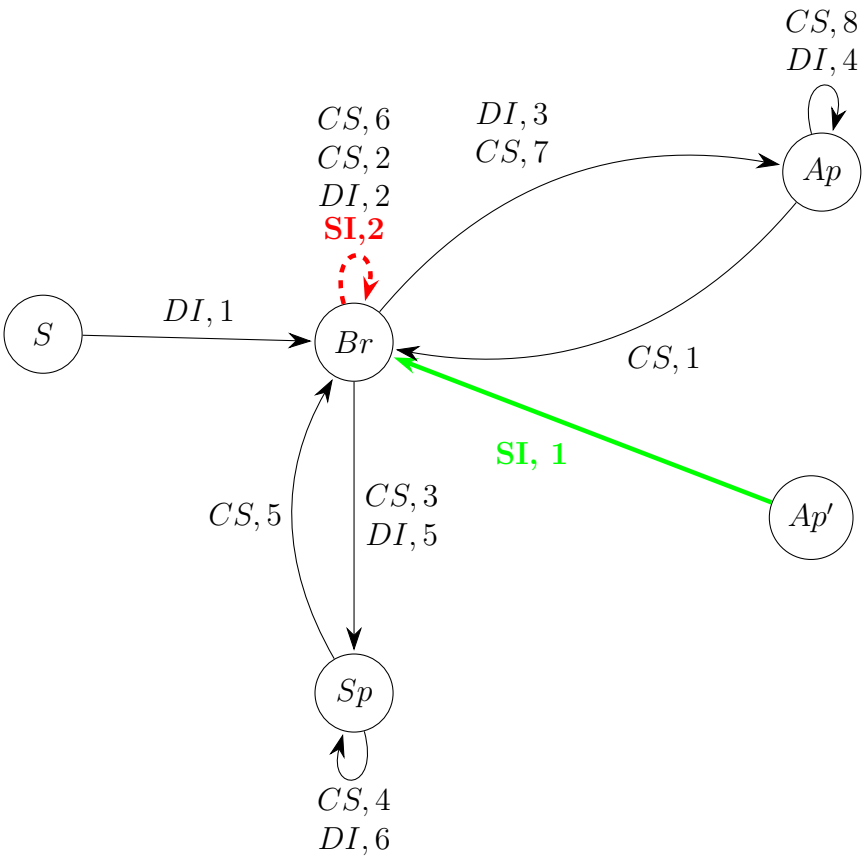

Figura 31 – Grafo de compromisso restruturado do sistema de detecção e supressão de incêndio.

Fonte: Próprio autor.

6 CONCLUSÃO

Este trabalho apresentou um método de monitoramento reativo de protocolos de sistemas em IoT sobre o MQTT. A abordagem é baseada em três procedimentos: a obtenção do modelo do sistema IoT por meio do TPi; a conversão do modelo em TPi para o modelo em contrato multilateral; e a aplicação do algoritmo de monitoramento reativo.

O cálculo temporal de processos proposto para modelagem do sistema foi derivado do Cálculo-π e do CCS, ambos propostos por Milner[14] [9, 15], adotando o uso de operadores lógicos e utilização de nomes abstratos para representação de processos, ações, variáveis e canais de comunicação, além da adição do aspecto temporal que determina o tempo de execução de um determinado processo.

Anterior ao método de conversão dos modelos, foi realizada uma análise do comportamento operacional do protocolo MQTT com nível de QoS igual a zero. Desta forma, foi possível compreender a estrutura desta estratégia de troca de mensagens, auxiliando na modelagem do estudo de caso proposto.

O processo de conversão do modelo em TPi foi desenvolvido com base na identificação dos elementos da linguagem capazes de serem estendidos para propriedades do modelo em contrato multilateral, como ações e compromissos.

O método proposto foi aplicado sobre um estudo de caso real de um sistema de detecção e supressão de incêndio presente no trabalho de Kang et al.[12]. Após a análise do sistema, o mesmo foi modelado com base na troca de mensagens entre os agentes envolvidos, de acordo com o comportamento operacional definido pelo MQTT com QoS igual a zero.

Dos resultados obtidos, constatou-se que o objetivo proposto foi atingido, no que se refere à identificação e detecção de violações contratuais e seus responsáveis. Para tanto, foi essencial a aplicação do modelo para representação de sistemas em contratos multilaterais desenvolvido por Xu[27], pois possibilitou a formalização de relações complexas entre os agentes envolvidos por meio de uma especificação detalhada e estruturada de cada elemento que compõe suas propriedades.

O modelo em TPi do protocolo MQTT elaborado por Aziz[11], teve um papel fundamental para compreensão do comportamento operacional do protocolo em relação aos níveis de QoS, visto que a linguagem oferece uma estratégia de alto nível para representação da comunicação entre sistemas concorrentes.

A partir do modelo em TPi do sistema de detecção e supressão de incêndio, o método de conversão proposto na Seção 4.3 foi capaz de identificar os compromissos e

ações do sistema. Porém, não foi possível determinar os papéis exercidos pelos agentes e suas propriedades, assim como as ordens de execução dos compromissos. Desta forma, os elementos pendentes foram determinados explicitamente, de modo a completar o modelo em contrato multilateral do sistema.

Como trabalhos futuros, sugere-se:

- Determinar meios para representação das propriedades que ficaram pendentes no modelo em TPi, no que se refere a conversão para contrato multilateral;
- Adaptar o método de conversão para casos mais complexos, visto que, apenas um subconjunto dos elementos sintáticos do TPi foram considerados;
- Realizar novamente as etapas do método proposto considerando os níveis de QoS restantes (1 e 2);
- Desenvolver um *broker* MQTT que implemente o monitoramento reativo entre dispositivos conectados.

REFERÊNCIAS

[1] MAYER, C. P. Security and privacy challenges in the internet of things. *Electronic Communications of the EASST*, v. 17, 2009.

[2] SANKAR, S.; SRINIVASAN, P. Internet of things (iot): A survey on empowering technologies, research opportunities and applications. *International Journal of Pharmacy Technology*, 2016.

[3] ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

[4] BUYYA, R.; DASTJERDI, A. *Internet of Things: Principles and Paradigms*. [S.l.]: Elsevier Science, 2016. ISBN 9780128093474.

[5] SHELBY, Z.; HARTKE, K.; BORMANN, C. The constrained application protocol (coap). IETF, 2014.

[6] BANKS, A.; GUPTA, R. *Message Queuing Telemetry Transport (MQTT) v3.1.1*. OASIS Standard, 2014. Disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>.

[7] HUTH, M.; RYAN, M. *Logic in Computer Science: Modelling and Reasoning about Systems*. [S.l.]: Cambridge University Press, 2004. ISBN 9781139453059.

[8] MURA, W. A. D. *Conflict detection on multiparty contracts*. 133 p. Dissertação (Mestrado) — State University of Londrina, Londrina-PR, 2016.

[9] MILNER, R.; PARROW, J.; WALKER, D. A calculus of mobile processes, i. *Information and computation*, Elsevier, v. 100, n. 1, p. 1–40, 1992.

[10] BERGER, M.; HONDA, K. The two-phase commitment protocol in an extended π-calculus. *Electronic Notes in Theoretical Computer Science*, Elsevier, v. 39, n. 1, p. 21–46, 2003.

[11] AZIZ, B. A formal model and analysis of an iot protocol. *Ad Hoc Networks*, Elsevier, v. 36, p. 49–57, 2016.

[12] KANG, D.-H. et al. Room temperature control and fire alarm/suppression iot service using mqtt on aws. In: IEEE. *Platform Technology and Service (PlatCon), 2017 International Conference on*. [S.l.], 2017. p. 1–5.

[13] ABE, J. M.; SCALZITTI, A.; FILHO, J. Inácio da S. *Introdução à Lógica para a Ciência da Computação*. [S.l.]: Arte & Ciência, 2001.

[14] MILNER, R. A calculus os communicating systems. *Laboratory for Foundations of Computer Science*, LFCS Report Series, 1986.

[15] MILNER, R.; PARROW, J.; WALKER, D. A calculus of mobile processes, ii. *Information and Computation*, Elsevier, v. 100, n. 1, p. 41–77, 1992.

[16] WEBER, R. H. Internet of things–new security and privacy challenges. *Computer law & security review*, Elsevier, v. 26, n. 1, p. 23–30, 2010.

[17] SANTOS, B. P. et al. Internet das coisas: da teoria à prática. *Minicursos / XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 1–50, 2016.

[18] GARTNER'S 2015 hype cycle for emerging technologies identifies the computing innovations that organizations should monitor. *Gartner Web site*, 2015. Disponível em: <http://www.gartner.com/newsroom/id/3114217>.

[19] PRESS, G. Internet of things by the numbers: Market estimates and forecasts. *Forbes*, 2014. Disponível em: <https://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/#6b493e34b919>.

[20] DANOVA, T. The internet of everything. *Business Insider*, 2014. Disponível em: <http://www.businessinsider.com/the-internet-of-everything-2014-slide-deck-sai-2014-2?op=1/#>.

[21] BANDYOPADHYAY, D.; SEN, J. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, Springer, v. 58, n. 1, p. 49–69, 2011.

[22] MUKHERJEE, S. *Ranking System for IoT Industry Platform*. Dissertação (Mestrado) — KTH Royal Institute of Technology, 2016.

[23] SUO, H. et al. Security in the internet of things: a review. In: IEEE. *Computer Science and Electronics Engineering (ICCSEE), 2012 international conference on*. [S.l.], 2012. v. 3, p. 648–651.

[24] LOCKE, D. Mq telemetry transport (mqtt) v3. 1 protocol specification. *IBM developerWorks Technical Library*, 2010.

[25] CHEN, W.-J. et al. *Responsive Mobile User Experience Using MQTT and IBM MessageSight*. [S.l.]: IBM Redbooks, 2014.

[26] FENECH, S.; PACE, G. J.; SCHNEIDER, G. Conflict analysis of deontic contracts. *NWPT*, v. 8, 2008.

[27] XU, L. A multi-party contract model. *ACM SIGecom Exchanges*, ACM, v. 5, n. 1, p. 13–23, 2004.

[28] XU, L.; JEUSFELD, M. A.; GREFEN, P. W. Detection tests for identifying violators of multi-party contracts. *ACM SIGecom Exchanges*, ACM, v. 5, n. 3, p. 19–28, 2005.

[29] MORGAN, J. Everything you need to know about the internet of things. *Forbes*, 2014. Disponível em: <https://www.forbes.com/sites/jacobmorgan/2014/10/30/everything-you-need-to-know-about-the-internet-of-things/#2c360eea3ae7>.

[30] MLADENOV, K. et al. Formal verification of the implementation of the mqtt protocol in iot devices. *SNE Master Research Projects 2016 - 2017*, 2017.

[31] KUSHALNAGAR, N.; MONTENEGRO, G.; SCHUMACHER, C. *IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals*. [S.l.], 2007.

[32] FIELDING, R.; RESCHKE, J. Hypertext transfer protocol (http/1.1): Message syntax and routing. IETF, 2014.

[33] RAGGETT, D. et al. Html 4.01 specification. *W3C recommendation*, v. 24, 1999.

[34] AL-FUQAHA, A. et al. Toward better horizontal integration among iot services. *IEEE Communications Magazine*, IEEE, v. 53, n. 9, p. 72–79, 2015.

[35] STANFORD-CLARK, A.; TRUONG, H. L. Mqtt for sensor networks (mqtt-sn) protocol specification. *International business machines (IBM) Corporation version*, v. 1, 2013.

[36] DATA distribution services specification, V1.2. Object Manage Group (OMG), 2015.

[37] GODFREY, R.; INGHAM, D.; SCHLOMING, R. *Advanced Message Queuing Protocol (AMQP) Version 1.0*. [S.l.]: OASIS Standard, 2012.

[38] SAINT-ANDRE, P. Extensible messaging and presence protocol (xmpp): Core. 2011.

[39] MIRANDA, M. B. Teoria geral dos contratos. *Revista Virtual Direito Brasil*, v. 2, n. 2, p. 15, 2008.

[40] GOMES, O. Contratos. atual. por antonio junqueira de azevedo e francisco paulo de crescenzo marino. *Rio de janeiro: Forense*, p. 10, 2007.

[41] AUSÍN, T. *Entre la lógica y el derecho*. Barcelona: Plaza y Valdés, 2005. v. 1.

[42] BOBBIO, N.; CICCO, C. D. *Teoria do ordenamento jurídico*. 6rd. ed. [S.l.]: UnB, 1995. 105–109 p.

[43] XU, L. *Monitoring multi-party contracts for e-business*. [S.l.]: Paul de Vrieze, 2004.

[44] ANGELOV, S.; GREFEN, P. B2b econtract handling-a survey of projects, papers and standards. University of Twente, Centre for Telematics and Information Technology, 2001.

[45] BOSSE, T. et al. Automated formal analysis of human multi-issue negotiation processes. *Multiagent and Grid Systems*, IOS Press, v. 4, n. 2, p. 213–233, 2008.

[46] CHADHA, R.; KREMER, S.; SCEDROV, A. Formal analysis of multiparty contract signing. *Journal of Automated Reasoning*, Springer, v. 36, n. 1-2, p. 39–83, 2006.

[47] KORDY, B.; RADOMIROVIC, S. Constructing optimistic multi-party contract signing protocols. In: IEEE. *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*. [S.l.], 2012. p. 215–229.

[48] BEDREGAL, B.; ACIÓLY, B. Lógica para a ciência da computação. *Versão Preliminar, Natal, RN*, 2002.

[49] SOUZA, J. *Lógica para ciência da computação*. 3rd. ed. [S.l.]: Elsevier Brasil, 2017.

[50] SMULLYAN, R. M. *First-order logic*. [S.l.]: Courier Corporation, 1995.

[51] CHELLAS, B. F. *Modal logic: an introduction*. [S.l.]: Cambridge university press, 1980.

[52] PNUELI, A. The temporal logic of programs. In: IEEE. *Foundations of Computer Science, 1977., 18th Annual Symposium on*. [S.l.], 1977. p. 46–57.

[53] CLARKE, E.; GRUMBERG, O.; PELED, D. *Model Checking*. [S.l.]: MIT Press, 1999. ISBN 9780262032704.

[54] KRIPKE, S. A. A completeness theorem in modal logic. *The journal of symbolic logic*, Cambridge University Press, v. 24, n. 1, p. 1–14, 1959.

[55] MULLER-OLM, M.; SCHMIDT, D.; STEFFEN, B. Invited talks and tutorials-model-checking. a tutorial introduction. *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 1973-, v. 1694, p. 330–354, 1999.

[56] KATOEN, J.-P. *Concepts, algorithms, and tools for model checking*. [S.l.]: IMMD Erlangen, 1999.

[57] BAIER, C.; KATOEN, J. *Principles of Model Checking*. [S.l.]: MIT Press, 2008. ISBN 9780262026499.

[58] BAETEN, J. C. A brief history of process algebra. *Theoretical Computer Science*, Elsevier, v. 335, n. 2-3, p. 131–146, 2005.

[59] HAUGEN, B.; FLETCHER, T. Multi-party electronic business transactions. *Valid on 060630*, 2002.