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

GUSTAVO OLIVEIRA DIAS

ALGORITMO PARA VERIFICAÇÃO FORMAL DO PROTOCOLO MQTT

BANDEIRANTES–PR

2017

GUSTAVO OLIVEIRA DIAS

ALGORITMO PARA VERIFICAÇÃO FORMAL DO PROTOCOLO MQTT

Versão Preliminar de 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

ALGORITMO PARA VERIFICAÇÃO FORMAL DO PROTOCOLO MQTT

Versão Preliminar de 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. Me. Carlos Eduardo Ribeiro
Universidade Estadual do Norte do Paraná

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

Bandeirantes–PR, 15 de Setembro de 2017

SUMÁRIO

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 [1].

Advinda do avanço das áreas de sistemas embarcados, microeletrônica, comunicação e tecnologia de informações [2], a IoT é considerada uma das mais promissoras tecnologias emergentes [3]. Conforme o crescimento da quantidade de dispositivos e aplicativos conectados a *Internet*, aumenta o interesse dos usuários em adquiri-los, tal como o de empresas em investir neste mercado [4].

Em estudo realizado pela *Acquity Group*, mais de dois terços dos consumidores planejam comprar tecnologia conectada para suas casas até 2019 [5]. Também, até 2017, 82% das empresas implementarão a IoT de alguma forma, de acordo com relatório publicado pela *Business Insider UK* [6].

Através 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 [7, 8, 9].

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 [5]. Capaz de alocar até 2^{128} endereços IP, o IPv6 possibilita a conexão de uma imensa quantidade de dispositivos [4].

Segundo a Cisco, até 2020 haverá cerca de 50 bilhões de dispositivos conectados, e até 2022, o mercado de soluções para IoT acrescentará em torno de 14,4 trilhões de dólares ao PIB global [10].

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 [1, 2, 11]. 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 em IoT [8, 11].

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*) [12].

Padronizado em 2014 pela OASIS [13], 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 [14].

Com o advento do MQTT, suas propriedades e aplicações tornaram-se alvo de pesquisas, a fim de encontrar vulnerabilidades, ambiguidades e inconsistências, contribuindo com o aprimoramento da tecnologia [15, 16].

Diversos métodos são utilizados por especialistas para verificação e validação de propriedades de protocolos e sistemas complexos, entre eles, destaca-se: simulação, teste de protótipo, verificação dedutiva e verificação formal [17].

Com exceção da verificação formal, os demais métodos, embora sejam amplamente usados, não garantem total ausência de erros no funcionamento do sistema, uma vez que são incapazes de checar de forma automática todos os caminhos possíveis, visto que dependem de decisões humanas acerca de quais pontos devem ser analisados, tornando-os suscetíveis a falhas [17].

Ademais, tais métodos apresentam deficiências que podem envolver tempo excessivo para análise e observação, além de gastos elevados com especialistas encarregados de aplicá-los [17].

A verificação formal engloba a técnica de teste baseado em modelo, também conhecida como *Model Checking*, a qual proporciona uma série de vantagens em comparação com os outros métodos, tais como aplicação automática, verificação completa de todos os caminhos possíveis e produção de contraexemplos que demonstram quando uma dada propriedade não é satisfeita. Além disso, não é necessário a especificação completa do sistema, uma vez que seus módulos podem ser representados separadamente. Assim, pode-se verificar apenas os pontos críticos, poupando tempo e provendo maior segurança e confiabilidade [17].

Portanto, este trabalho tem como objetivo desenvolver um algoritmo para verificação formal da versão 3.1.1 do protoclo MQTT e suas propriedades por meio de *Model Checking*.

O algoritmo será implementado e testado por meio de um estudo de caso. Com isso, espera-se que a comunicação entre dispositivos nos projetos de IoT que utilizam o protocolo MQTT possa ser verificada antes de sua implantação.

1.1 Motivação

Como dito anteriormente, a Internet das Coisas é uma das mais poderosas tecnologias emergentes, estima-se que até 2022 ela irá adicionar de 14,4 trilhões de dólares ao PIB global, e que em 2020 haverá em torno de 50 bilhões de dispositivos interconec-

tados [10], aumentando a demanda por sensores, dispositivos de rede sem fio, serviços de armazenamento em nuvem, entre outros serviços e dispositivos que compõem este ambiente tecnológico, despertando assim, o interesse dos setores industriais 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 interoperabilidade 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 [13].

Em seu trabalho, Mladenov et al.[16] mostram que as aplicações do protocolo MQTT nem sempre implementam corretamente todos os seus requisitos normativos [16]. Porém, não atribuem tal problema a falhas na especificação do protocolo, já que o mesmo não foi profundamente estudado, embora esta seja uma possível causa.

Um modo de verificar protocolos de maneira precisa e com rigor matemático é utilizando técnicas de modelagem e verificação formal, sendo possível representar propriedades do protocolo por meio de um modelo e realizar a verificação formal das mesmas. Assim, todos os estados possíveis que dada propriedade pode assumir é automaticamente testado, e, em caso de não conformidade com o esperado, um contraexemplo é mostrado, auxiliando na busca por falhas, e, consequentemente, contribuir com correções e avanços dos campos de aplicação [17].

No trabalho desenvolvido por Aziz[15], o protocolo MQTT foi modelado por meio de uma álgebra temporal de processos, chamada *TPi*. Posteriormente, foi aplicado um processo manual de análise formal que apontou ambiguidades na especificação. Sendo assim, foram sugeridas melhorias para sanar esses problemas.

Entretanto, não se tem conhecimento de algum estudo que realize a verificação formal automática do protocolo, de maneira que possibilite que projetos de IoT que utilizam o MQTT possam ser modelados e testados antes de sua implantação.

Portanto, a motivação deste trabalho consiste em colaborar com trabalhos futuros de IoT, os quais a comunicação entre os dispositivos envolvidos possa ser modelada e testada em busca de falhas ainda na fase de projeto, encurtando o tempo de desenvolvimento e ajudando na difusão do uso do MQTT.

1.2 Formulação e Escopo do Problema

Como relatado acima, não se tem conhecimento de alguma ferramenta baseada em verificação formal que permita modelar a comunicação entre os dispositivos de projetos em IoT, assim como realizar a checagem automática de suas propriedades. Em vista disso, a principal questão a ser respondida é:

É possível modelar e verificar a comunicação entre dispositivos que utilizam o protocolo MQTT?

No intuito de responder essa questão, a resolução das seguintes subquestões devem ser fornecidas:

(1) É possível modelar o MQTT?

No estudo desenvolvido por Aziz[15], e que serve de base para este trabalho, o protocolo é modelado através de uma álgebra temporal de processos, chamada *TPi*.

(2) É possível realizar a verificação formal do MQTT?

A partir do elaborado por Aziz[15], pode-se criar um algoritmo baseado em *Model Checking* que faça a verificação de suas propriedades, o que leva ao objetivo do trabalho.

1.3 Objetivos gerais e específicos

O objetivo geral deste trabalho consiste no desenvolvimento de um algoritmo baseado em *Model Checking* que permita modelar e realizar a verificação da comunicação entre dispositivos de IoT que utilizam o MQTT, e assim, procurar por falhas na troca de mensagens entre eles.

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

- Levantar os problemas mais comuns nos protocolos de comunicação em IoT;
- Levantar um conjunto de propriedades que representam o funcionamento do protocolo MQTT;
- Definir métodos de modelagem formal de protocolos que possam ser aplicados sobre o MQTT;
- Desenvolver um algoritmo para verificação formal do protocolo com base no modelo estabelecido;
- Testar a solução desenvolvida em um estudo de caso.

1.4 Método de pesquisa

A pesquisa desenvolvida neste trabalho possui natureza explicativa, pois além de observar as conclusões obtidas, pretende-se buscar suas causas e explicações a fim de apresentar conhecimento novo.

Quanto aos procedimentos técnicos, será realizada uma pesquisa experimental, no intuito de observar e provocar alterações no ambiente pesquisado sempre que necessário.

Tais procedimentos podem ser observados abaixo:

1. Definição dos problemas mais comuns nos protocolos de comunicação em IoT;
2. Levantamento de um conjunto de propriedades que representam o funcionamento do protocolo MQTT;
3. Definição do método de modelagem formal que será aplicado;
4. Fazer a modelagem de acordo com o método escolhido;
5. Realizar a verificação das propriedades por meio de *Model Checking* a procurar falhas como:
   a) *Deadlocks*;
   b) *Livelocks*;
6. Desenvolver aplicação que implemente o protocolo MQTT suprindo possíveis falhas encontradas;
7. Implantar e testar a solução desenvolvida em um estudo de caso.

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 o 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[18] classifica a IoT em 8 tópicos:

- **Comunicação:** permiti 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.

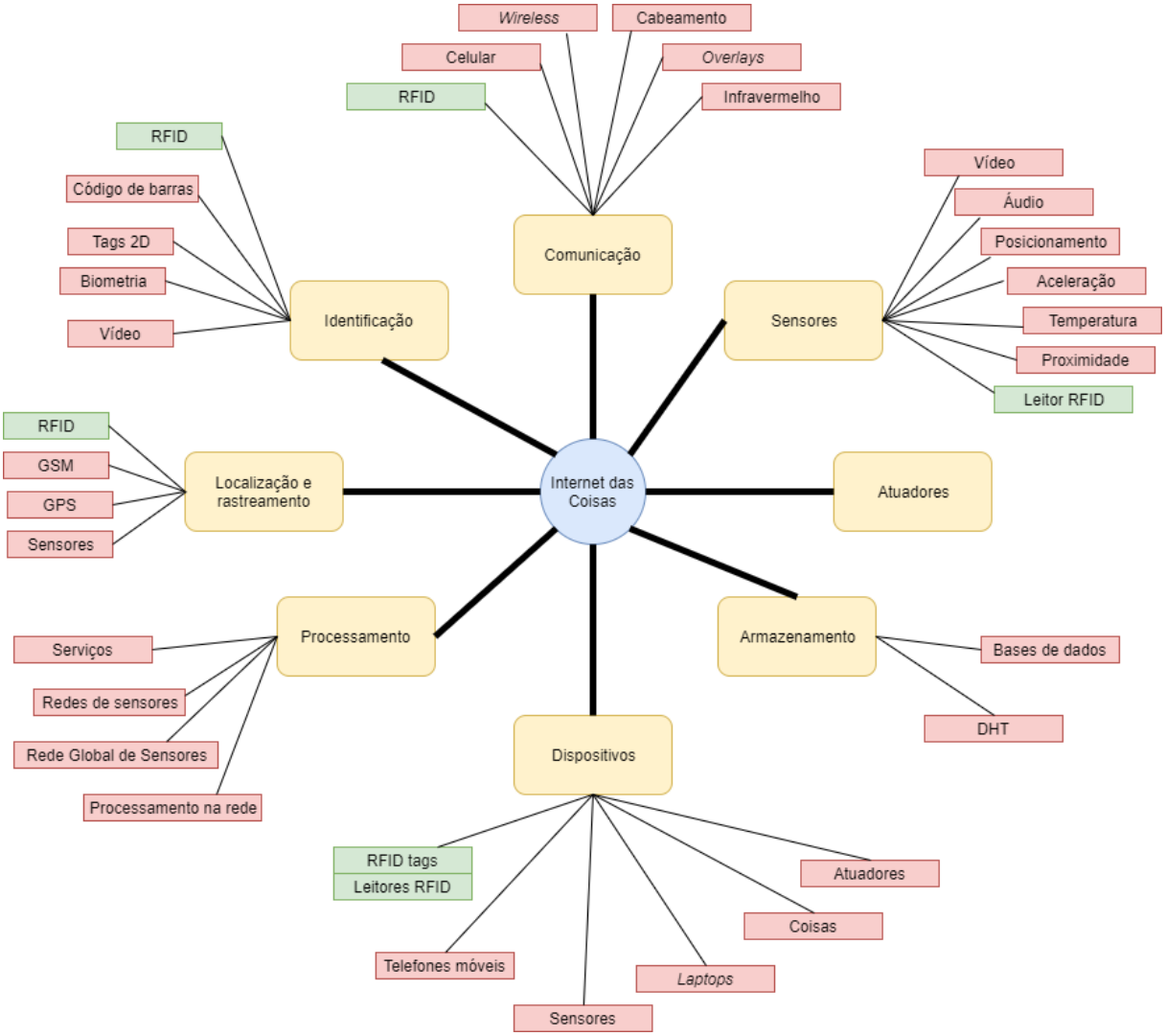

Figura 1 – Categorização dos tópicos e tecnologias em IoT [18], adaptado pelo autor.

A Internet das Coisas é um paradigma que desempenha e continuará desempenhando um papel vital num futuro próximo [7]. Sankar e Srinivasan[7] 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 2.

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 se 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.

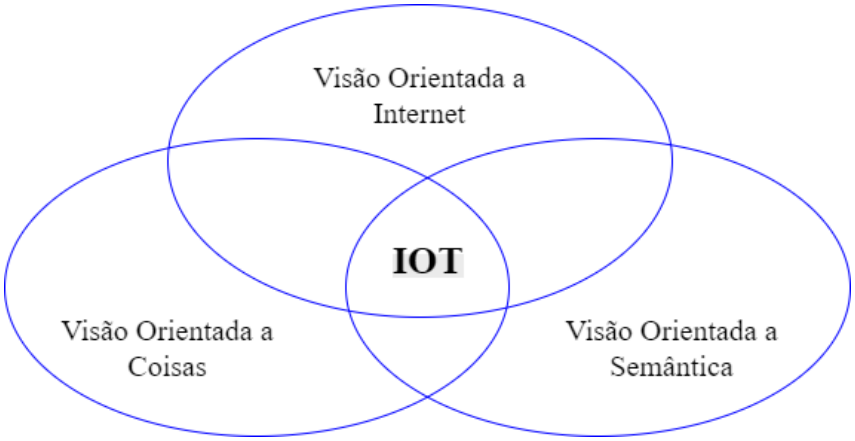

Figura 2 – Três paradigmas de visão para IoT [7], adaptado pelo autor.

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, malhar, entre muitas outras [8, 9].

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 [9].

- **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 [7].

- **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 [9].
- **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 [7, 8, 9]. Todos os casos acima são exemplos de como a IoT pode auxiliar no aperfeiçoamento de processos envolvendo diversos ambientes.

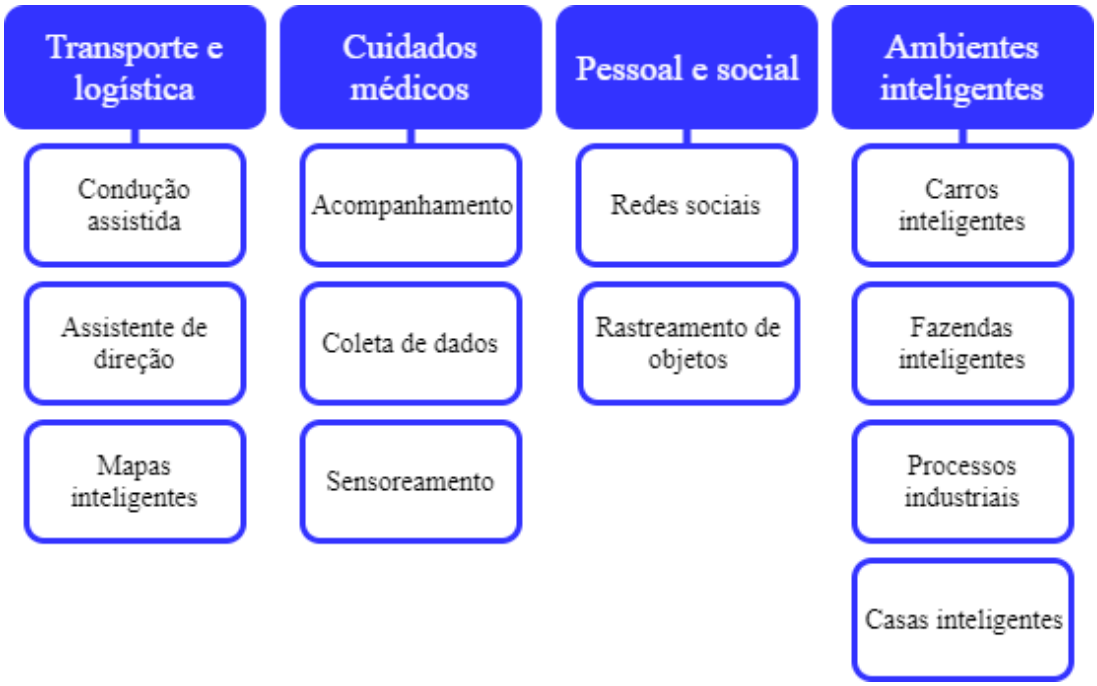

Figura 3 – Áreas de aplicação e principais contextos [9], adaptado pelo autor.

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 [19].

Sendo assim, nota-se a existência de uma série de protocolos de comunicação disponíveis, considerando as 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 4, alguns protocolos são exibidos de acordo com a camada no qual é aplicado de acordo com o modelo *TCP/IP*.

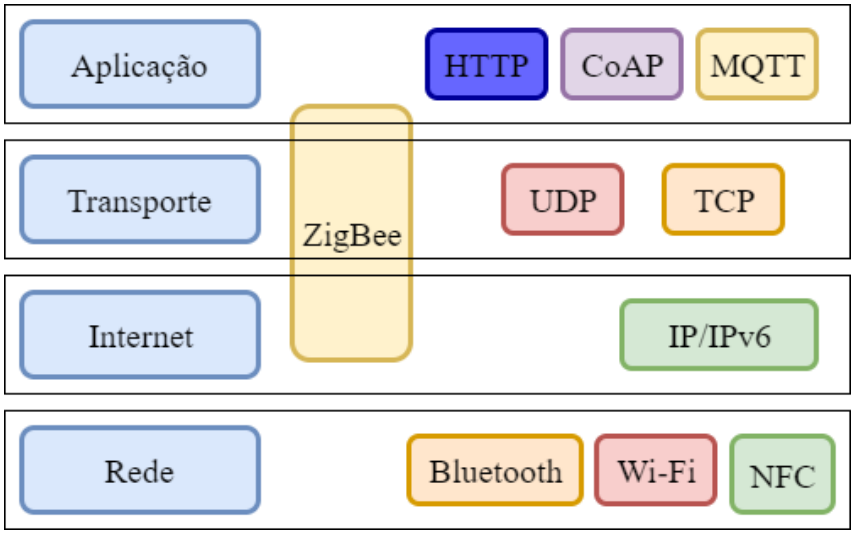

Figura 4 – Uso dos protocolos de acordo com a camada *TCP/IP* [19], adaptado pelo autor.

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 [2].

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*, dispositivos que seguem o padrão IEEE 802.15.4 (*LR-WPANs*) limitam-se a pacotes de 128 bytes [2].

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 [20].

Outro protocolo extensamente utilizado, mas que mostra-se inadequado para aplicações em IoT é o HTTP [2]. Porém, os protocolos CoAP (*Constrained Application Pro-*

tocol) 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 (*Constrained Application Protocol*) e propriedades gerais e específicas do MQTT (*Message Queue Telemetry Transport*), 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 [21], como pode ser observado na figura 5.

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 [21].

A comunicação entre o cliente e o servidor é feita mediante a mensagens, que são codificadas por meio da linguagem de marcação HTML [22]. O cliente envia uma mensagem de requisição de um determinado recurso ao servidor, que recebe e requisição e enviar uma resposta, como ilustrado na Figura 5. As mensagens possuem cabeçalho e corpo, além de um método, no caso das mensagens de requisição [21]. 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.

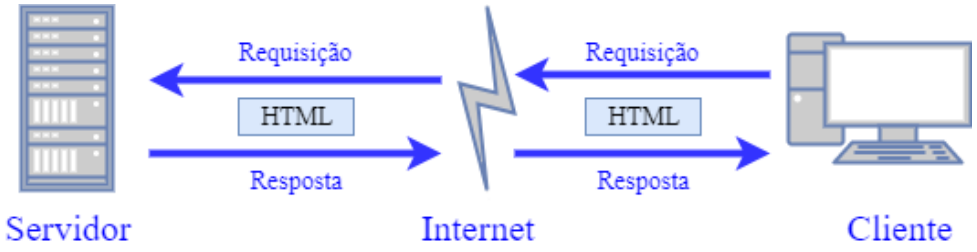

Figura 5 – Estratégia requisição/resposta sobre o paradigma cliente/servidor.

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 [21], 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 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 [21].

A Figura 6 apresenta 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 [21], com informações acerca da versão do protocolo ("HTTP/1.1"), o status do código ("*200*" = bem sucedido) e uma *Reason-Phrase* [21] ("*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.

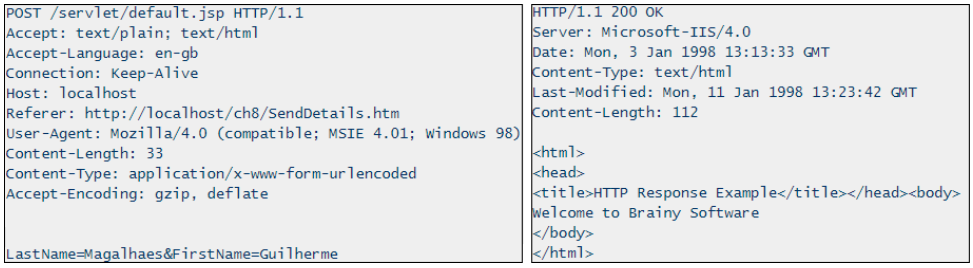

Figura 6 – Exemplo de requisição e resposta HTTP.

2.2.2 CoAP

Definido em 2010 pela IETF, o CoAP tounou-se um *Request for Comments* (*RFC*) em 2014 [23]. 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 [23].

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* [23].

Segundo o RFC 7252 [23], 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 [23] 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

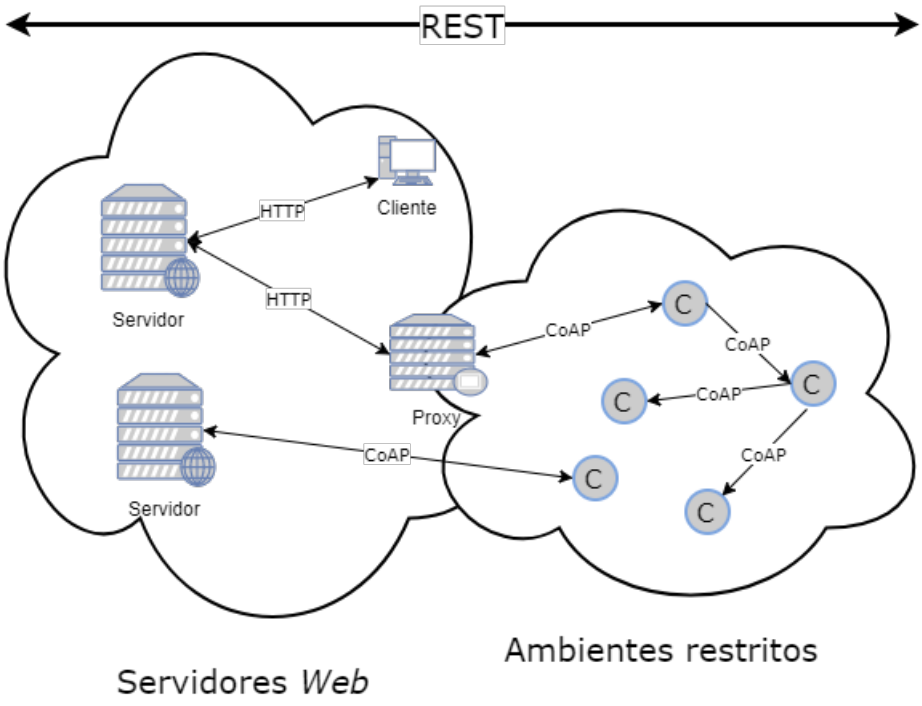

Figura 7 – Arquitetura CoAP [23].

A Figura 7 apresenta 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 *RESTfull* [23].

A comunicação com o CoAP consiste na troca de mensagens compactas transportadas sobre o UDP, sendo que, há uma camada DTLS entre as mensagens e o UDP, usada para prover segurança aos dados [23].

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 [23]. O formato da mensagem é ilustrado na Figura 8.

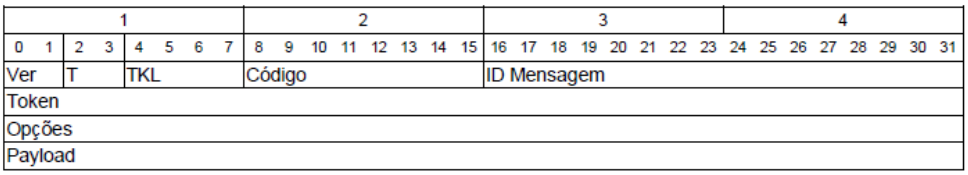

Figura 8 – Datagrama CoAP [23].

Os campos que constituem o cabeçalho do datagrama são definidos pelo RFC 7252 [23] 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 [23]

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 [23]

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 [13].

Com um cabeçalho fixo de apenas 2 *bytes* e utilizando a estratégia 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 [13].

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 [13]. Estas características incluem:

- Uso do padrão publicação/assinatura, provendo distribuição de mensagem um-para-muitos e dissociação das aplicações (Figura);
- 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 publicação/assinatura (do inglês, *publish/subscriber*), o MQTT deve implementar um servidor próprio, chamado de *broker*, que serve como intermediador para troca de mensagens entre os dispositivos.

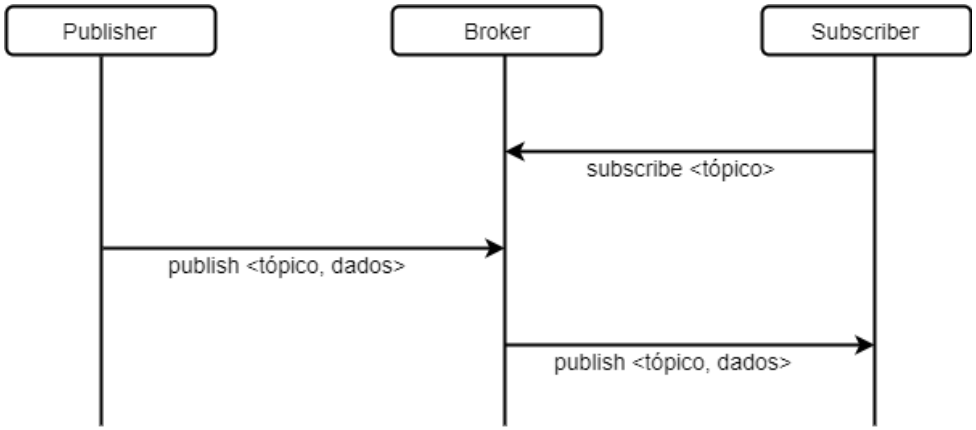

Figura 9 – Exemplo de comunicação MQTT sobre a estratégia publicação/assinatura (*publish/subscribe*).

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 9.

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 níveo 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 mostra a Figura 10. O primeiro destes é o cabeçalho fixo, presente em todos os pacotes, que pode

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

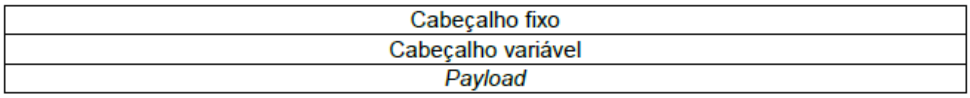

Figura 10 – Estrutura de um pacote MQTT [13].

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 11. 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*.

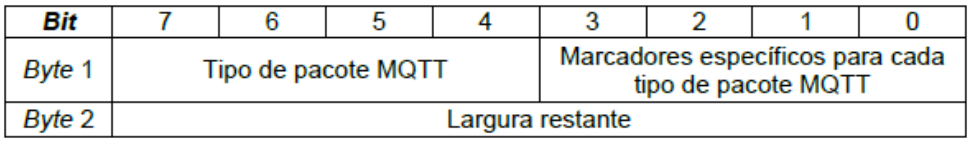

Figura 11 – Formato do cabeçalho fixo MQTT [13].

Com exceção do pacote PUBLISH, que 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. A Figura 12 apresenta 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 12 – Formato do cabeçalho fixo de um pacote PUBLISH [13].

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.

A Tabela 3 cita 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 [13]

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 publicaçã. 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 disconectando
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 (C) e o Servidor (S) 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 [13] 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 13).

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 13 – Exemplo de diagrama de fluxo para entrega com QoS 0 [13].

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 [13], 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 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 14 – Exemplo de diagrama de fluxo para entrega com QoS 1 [13].

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 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.

No entretanto, o MQTT-4.3.3-2 [13] 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 15 – Exemplo de diagrama de fluxo para entrega com QoS 2 [13].

No caso ilustrado na Figura 15, 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*) [24].

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

gura 16 cita 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 16 – Características e propriedades dos protocolos de aplicação em IoT [7], adaptado pelo autor.

O protocolo MQTT-SN é uma adaptação do MQTT voltada para redes de sensores, pois é mais leva e exige ainda menos recursos em relação ao MQTT [25].

Utilizando uma arquitetura baseada em um *broker*, o DDS (*Data Distribution Service*) [26] 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 [27] e o XMPP [28], 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 [29]. A origem etimológica do vocábulo contrato conduz ao vínculo jurídico das vontades com vistas a um objeto específico, sendo um acordo e vontades criador de direitos e obrigações entre as partes envolvidas [30]. 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 [29, 30]. Segundo [31], 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 [30].

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 [32], em que ocorrem uma ou mais contradições entre as normas [33]. 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 [32].

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 [34].

São diversas as áreas de aplicação dos contratos eletrônicos. Em meios legais, contratos físicos são convertidos ou substituídos para o meio eletrônico. Técnicas de contratação 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 expressão a intenção das partes envolvidas [34]. 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 [29].

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 [29].

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 [35].

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 [29].

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 [36]. 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 [29].

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 [37]. 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 é pode envolver ou não o uso de mediadores [38].

- **Detecção de conflitos:** A detecção de conflitos [34] visa a detecção e eliminação de conflitos normativos de um contrato, pois esta situação invalida um contrato,

induzindo a uma violação [33].

- **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 [39, 40].

- **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 [34]. 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 [34].

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 [37]. A seguir são descritas 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 [41]. 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 [41].

A lógica proposicional tem como objetivo modelar o raciocínio humano, partindo de proposições, como frases declarativas [41]. 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 [41].

A sintaxe do cálculo proposicional é definida por um alfabeto constituído por [42]:

- **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, a Tabela 4 apresenta algumas formas de leitura para linguagem natural atribuídas aos conectivos proposicionais.

Tabela 4 – Leituras dos conectivos proposicionais para a linguagem natural [43].

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 [41]. 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 [41].

A linguagem, ou conjunto de fórmulas $\mathcal{L}$, é definida recursiva ou indutivamente pelas seguintes regras [41, 43]:

- 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 [41]. Já o conjunto $\mathcal{I}$ é a coleção finita de pontos iniciais, também chamados de axiomas [41].

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) [41].

A Tabela 5 demonstra as propriedades dos conectivos de negação, conjunção, disjunção, implicação e bi-implicação, respectivamente. O operador 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 falta. 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 $(Q \wedge (Q \leftrightarrow P))$, $(Q \leftrightarrow P)$, $(P \vee Q), P, Q$. 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 deriva em suas sub-fórmulas repetidamente até que as "folhas" da árvores contenham apenas as proposições atômicas, como ilustrado na Figura 17.

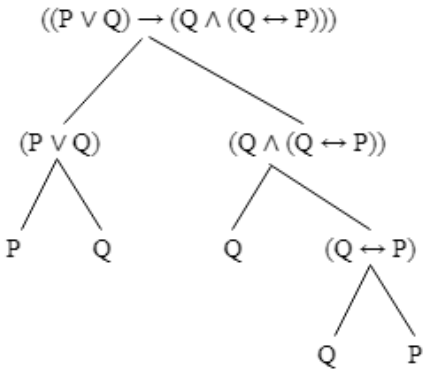

Figura 17 – Árvore de derivação associada à fórmula $((P \vee Q) \rightarrow (Q \wedge (Q \leftrightarrow P)))$.

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 calculo 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 [44], as lógicas modais [45], e a lógica temporal [46].

2.4.2 Lógica temporal

A lógica temporal é um formalismo usado para descrever sequências de transações entre estados em sistemas reativos [17]. Portanto, uma fórmula lógica temporal geralmente é avaliada sobre um sistema de transição que modela uma especificação [17], isto é, um modelo de *Kripke* [47].

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 18, definido por:

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

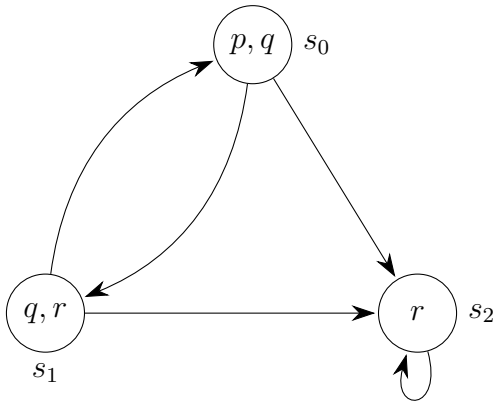

Figura 18 – Exemplo de um sistema de transição [48].

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 [17].

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 [29].

2.4.2.1 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 [49].

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 [29, 49].

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 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 [48], como exposto a seguir:

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

Figura 19 – Semântica da LTL.

As definições 2.2 e 2.3 da Figura 19 indicam as proposições verdadeiras e falsas, respectivamente. Na definição 2.4 é dito que a proposição p só é válida se estiver no pri-

meiro 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 contrato, 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 [29, 48]. Huth e Ryan[48] descreve 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* [48].

2.4.2.2 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 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 [48].

Na Figura 20, 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.

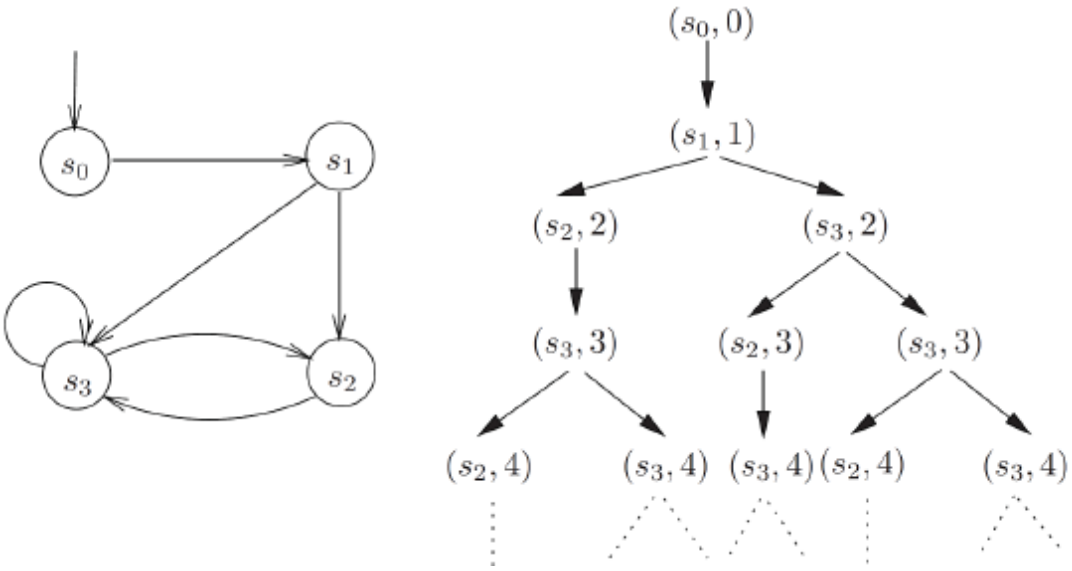

Figura 20 – Exemplo de árvore de computação num sistema de transição [48].

A sintaxe de construção de fórmulas CTL estende a LTL e adiciona os operadores existencial (E) e universal (A) 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 [50]. 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. [50]. 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 21, em que $s \in \mathcal{M}$ e φ é uma fórmula em CTL [29].

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

Figura 21 – Semântica da CTL.

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*) [51].

Outro 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 [52].

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 [52].

A seguir são descritos alguns cálculos utilizados para modelagem de processos concorrentes. O Cálculo de Sistemas de Comunicação (CCS) [53] serve como base para o desenvolvimento do Cálculo-π [54, 55], que por sua vez, possui uma extensão com aspecto temporal denominada TPi [56].

2.5.1 Cálculo de Sistemas de Comunicação

O CCS é um cálculo de processos desenvolvido por Milner[53] para modelagem de sistemas concorrentes sob duas ideias centrais: comunicação e sincronização [53]. Para isso, o formalismo utiliza de 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* [53].

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* representa que uma dada condição que 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 22, considerando os seguintes processos e suas definições:

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

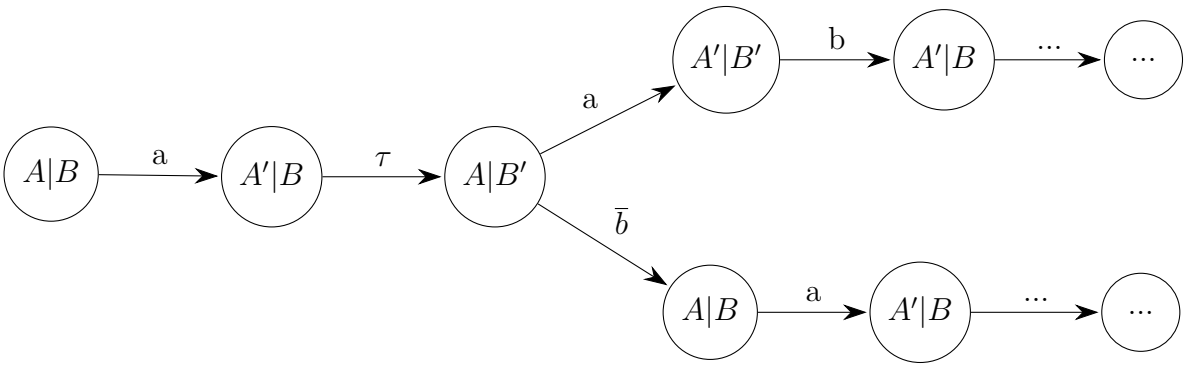

Figura 22 – Exemplo de árvore de computação num sistema de transição.

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*).

Tabela 7 – Semântica Estrutural Operacional do CCS [53].

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	$\frac{(P\{y_i/x_i\}) \xrightarrow{\alpha} P'}{K(y_i) \xrightarrow{f(\alpha)} P'}$	$fornece\ K(x_i) \stackrel{def}{=} P$

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 23 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.

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

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

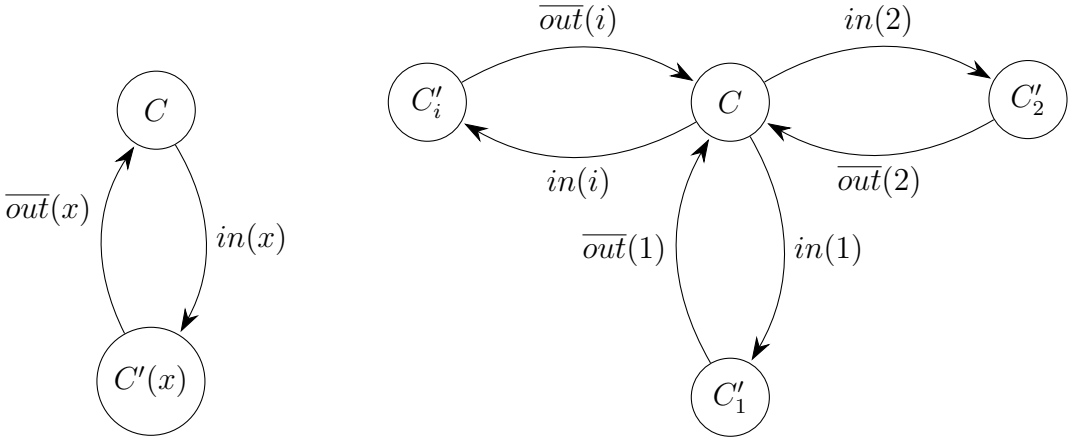

Figura 23 – Passagem de valores em CSS.

2.5.2 Cálculo-π

O Cálculo-π, desenvolvido por Milner, Parrow e Walker[54], 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 [54].

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 [54].

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 [55].

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** $\overline{a}y$: $P \xrightarrow{\overline{a}y} Q$ 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 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 está presente 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 Figura **??**, 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.

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.

Tabela 8 – Semântica Operacional Estrutural do Cálculo-π [55].

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	$\frac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'}$	$y \notin n(\alpha)$

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 24 a seguir apresenta o grafo de fluxo do sistema:

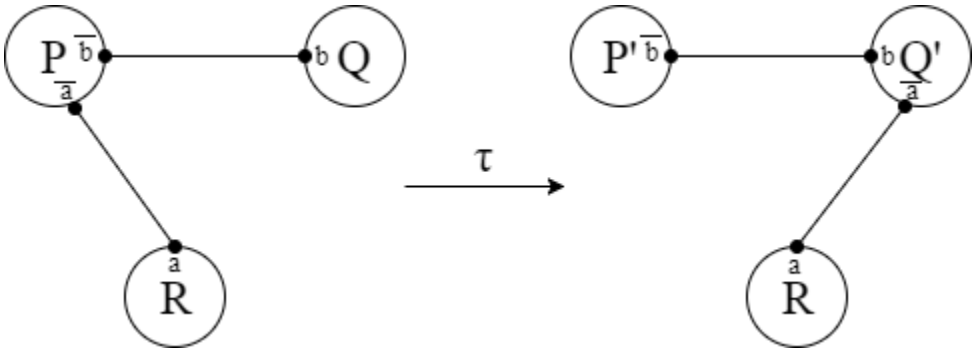

Figura 24 – Grafo de fluxo do sistema (2.46) [54].

2.5.3 Álgebra temporal de processos TPi

A TPi é uma álgebra temporal de processos inspirada no Cálculo-π [54, 55] e desenvolvida por Berger e Honda[56]. O formalismo apresenta um cálculo para transmissão

síncrona de mensagens capaz de expressar entradas cronometradas [15].

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 [15].

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 $A() \stackrel{def}{=} P$ [15].

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

Definições de $\equiv$ são padronizadas [55], 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 parametrização e não parametrizadas, respectivamente.

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

As regras de transição $\stackrel{\alpha}{\longrightarrow}$ são descritas detalhadamente por [56], 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 $\equiv$:

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

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 25 – Semântica da LTL.

3 PROJETO

Na Figura 26 é apresentado o escopo do trabalho em relação ao fluxo de funcionamento da aplicação proposta.

As elipses de contorno contínuo representam as entradas da aplicação, que irá receber as propriedades do protocolo MQTT a serem satisfeitas e o protocolo modelado. O losango indica uma condição para o direcionamento da saída da ferramenta. Por fim, as elipses de contorno tracejado determinam as possíveis saídas da aplicação.

Figura 26 – Escopo do projeto proposto.

4 CRONOGRAMA

O cronograma deste trabalho consiste na execução das atividades descritas a seguir e esquematizada na Tabela 9. O período determinado por este cronograma ocorre entre os meses de Março de 2017 a Dezembro de 2017 com granularidade mensal. Note que o símbolo $\surd$ indica uma tarefa realizada e $\bullet$ indica uma tarefa ainda não realizada.

Tabela 9 – Cronograma do trabalho

	2017								
Tarefas	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro
(1)	$\surd$	$\surd$	$\surd$	$\surd$					
(2)		$\surd$	$\surd$						
(3)			$\surd$	$\surd$					
(4)					$\surd$				
(5)					$\surd$	$\surd$			
(6)						$\surd$	$\bullet$	$\bullet$	$\bullet$
(7)		$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\bullet$	$\bullet$	$\bullet$

As tarefas listadas no cronograma são descritas a seguir:

1. Revisão bibliográfica sobre *Model Checking*, IoT e protocolos de comunicação;
2. Definir os problemas mais comuns nos protocolos de comunicação em IoT: Para isso, deve-se estudar os protocolos de comunicação, comparar suas funcionalidades, elencar as aplicações mais usadas e obter uma lista de problemas mais frequentes;
3. Levantar um conjunto de propriedades que representam o funcionamento do protocolo MQTT;
4. Definir métodos de modelagem formal de protocolos que possam ser aplicados sobre o MQTT;
5. Buscar técnicas para verificar as propriedades levantadas no modelo obtido: Para tanto, deve-se identificar e estudar as técnicas mais comuns para a verificação das propriedades, pesquisar ferramentas que implementam estas técnicas e decidir se alguma técnica será efetivamente utilizada ou se uma ferramenta será implementada no decorrer do projeto;
6. Implantar e testar a solução desenvolvida em um estudo de caso. Para tal, é preciso decidir se a solução sera verificada de forma manual ou automática;
7. Escrever o trabalho;

.

5 TRABALHOS RELACIONADOS

5.1 Especificação formal e análise de um protocolo de IoT

5.2 Linguagem para contratos multilaterais

REFERÊNCIAS

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

[2] 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.

[3] 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>.

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

[5] 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>.

[6] 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/#>.

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

[8] 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.

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

[10] 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>.

[11] 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.

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

[13] 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>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[38] 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.

[39] 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.

[40] 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.

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

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

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

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

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

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

[47] 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.

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

[49] 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.

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

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

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

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

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

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

[56] 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.