



UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

Transmissão Confiável de Imagens em Pub/Sub para IoT

Everson da Silva Flores

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Marcelo Rita Pias

Rio Grande, 2024

Ficha Catalográfica

F634t Flores, Everson da Silva.
Transmissão confiável de imagens em Pub/Sub para IoT /
Everson da Silva Flores. – 2024.
145 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande –
FURG, Programa de Pós-Graduação em Computação, Rio
Grande/RS, 2024.
Orientador: Dr. Marcelo Rita Pias.
Coorientadora: Dra: Bruna de Vargas Guterres.

1. Digital *Twins* 2. Confiabilidade em IoT 3. Microplásticos
4. Monitoramento ambiental 5. MQTT 6. Protocolo de fila de
mensagens 7. *PlanktonScope* 8. Ciência cidadã I. Pias, Marcelo Rita
II. Guterres, Bruna de Vargas III. Título.

CDU 004

Catálogo na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

ATA DE SESSÃO DE DEFESA DE DISSERTAÇÃO DE MESTRADO

Ata nº 12/2024

Aos vinte e nove dias do mês de novembro de dois mil e vinte e quatro, às nove horas, ocorreu a sessão de Defesa de Dissertação de Mestrado de EVERSON DA SILVA FLORES, que apresentou a dissertação intitulada **“Transmissão Confiável de Imagens em Pub/sub para IoT”**, realizada sob a orientação do Prof. Dr. Marcelo Rita Pias. A banca examinadora foi constituída pelos Prof. Dr. Marcelo de Gomensoro Malheiros (FURG) e Prof. Dr. Gerson Geraldo Homrich Cavalheiro (UFPel), sob a presidência do orientador. Após a apresentação do trabalho, a banca arguiu o candidato e, a seguir, deliberou pela

- () aprovação da Dissertação
- (X) aprovação da Dissertação, sugerindo modificações no texto
- () reprovação da Dissertação

Rio Grande, 29 de novembro de 2024.

Documento assinado digitalmente
 **MARCELO DE GOMENSORO MALHEIROS**
Data: 04/12/2024 08:13:13-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Marcelo de Gomensoro Malheiros

Documento assinado digitalmente
 **GERSON GERALDO HOMRICH CAVALHEIRO**
Data: 03/12/2024 14:21:50-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Gerson Geraldo Homrich Cavalheiro

Documento assinado digitalmente
 **MARCELO RITA PIAS**
Data: 29/11/2024 19:19:18-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Marcelo Rita Pias
Orientador

Agradecimentos

Este trabalho marca a conclusão de uma jornada que exigiu dedicação, aprendizado constante e a superação de inúmeros desafios. Não teria sido possível chegar até aqui sem a colaboração e o apoio de tantas pessoas que, direta ou indiretamente, fizeram parte desse processo. Cada contribuição, por menor que pareça, teve um impacto significativo na concretização desta dissertação.

Por isso, meu primeiro agradecimento é a Deus, por Sua graça, sabedoria e força, que me sustentaram em cada momento. Sua presença foi a base que me deu coragem para seguir em frente e alcançar as conquistas que celebro neste momento.

À minha família, cuja presença constante e apoio incondicional me deram forças para enfrentar os obstáculos e me motivaram a seguir em frente.

Ao meu orientador, professor Dr. Marcelo Rita Pias, e à professora Bruna Guterres, por sua colaboração exemplar, paciência e dedicação, que foram essenciais para o desenvolvimento deste trabalho. Suas sugestões e conselhos foram cruciais em todas as etapas deste projeto.

Minha gratidão aos professores do PPGCOMP, especialmente ao Dr. Vagner da Rosa, que, com suas valiosas dicas, contribuiu significativamente para a evolução deste trabalho e para meu crescimento acadêmico.

Um agradecimento especial aos colegas Msc. Thomaz Pereira da Silva Junior e Alberto Cabral, pela parceria, comprometimento e pelas contribuições indispensáveis durante o desenvolvimento e a aplicação da metodologia que sustentam esta dissertação.

Aos colegas de trabalho, especialmente ao técnico Cedenir Borges, pela dedicação e cuidado em assumir minhas responsabilidades profissionais durante os períodos em que estive focado nas atividades do mestrado. Sua compreensão e apoio foram fundamentais ao longo desta jornada.

Aos bolsistas do Laboratório de Hardware, em especial a Lucas Corrêa, que contribuiu de forma significativa auxiliando na elaboração textual deste trabalho, e a Samuel Vasconcelos, Maiara Rodrigues e Antônio Lopes, agradeço pela competência e dedicação em manter o laboratório funcionando com excelência durante este período.

Minha gratidão também aos colegas Matheus Daltro, Rafael Oliveira e Rodrigo Fagundes, pelo apoio, incentivo e pela colaboração em suas respectivas áreas, que foram essenciais para o meu progresso acadêmico.

Aos amigos e familiares que, de forma direta ou indireta, ofereceram palavras de encorajamento, compreensão e motivação nos momentos mais desafiadores, meu sincero agradecimento. Sem o apoio de vocês, este caminho teria sido muito mais difícil.

Por fim, agradeço à Universidade Federal do Rio Grande (FURG), ao Centro de Ciências Computacionais (C3) e ao Programa de Pós-Graduação em Engenharia da Computação (PPGCOMP), por proporcionarem o ambiente e os recursos necessários para que este trabalho fosse desenvolvido.

A todos vocês, meu muito obrigado!

Resumo

A presente dissertação explora a confiabilidade na transmissão de imagens em sistemas IoT baseados no modelo de publicação/assinatura (Pub/Sub). Especificamente, propõe-se a estruturação de uma rede de comunicação capaz de compartilhar imagens, mesmo em condições de conectividade limitada. A pesquisa faz uso da plataforma PlanktonScope, um dispositivo modular e acessível de citometria de fluxo adaptado para a oceanografia cidadã, que permite a análise de microplásticos em diferentes localidades, associando-os a suas respectivas localizações e facilitando a coleta de dados. Esse processo é especialmente desafiador em cenários com interrupções na rede e alta densidade de tráfego de dados. O principal objetivo deste trabalho é desenvolver e avaliar uma solução baseada no protocolo MQTT para a transmissão satisfatória e confiável de imagens de microplásticos, mesmo em cenários de conectividade restrita, contribuindo para o avanço do monitoramento ambiental e da ciência cidadã. A computação em borda é empregada para aproximar o processamento dos dados dos dispositivos de origem, reduzindo a latência e melhorando o desempenho geral. O estudo abrange a adaptação do PlanktonScope, a implementação do protocolo MQTT para o envio de grandes pacotes de dados e o desenvolvimento de módulos para particionamento e recuperação de imagens. Os resultados indicam que a rede proposta é viável para o monitoramento em tempo real de microplásticos, oferecendo uma solução tecnológica para a gestão ambiental e uma plataforma qualificada e escalável para a ciência cidadã. Este trabalho ressalta a importância de tecnologias emergentes na preservação de ecossistemas aquáticos e na promoção de práticas sustentáveis.

Palavras-chave: Digital Twins, Confiabilidade em IoT, Microplásticos, Monitoramento ambiental, MQTT, Protocolo de fila de mensagens, PlanktonScope, Ciência cidadã.

Abstract

The present dissertation explores the reliability of image transmission in IoT systems based on the publish/subscribe (Pub/Sub) model. Specifically, it proposes the structuring of a communication network capable of sharing microplastic images, even under limited connectivity conditions. The research utilizes the PlanktonScope platform, a modular and accessible flow cytometry device adapted for citizen oceanography, which enables the analysis of microplastics in different locations, associating them with their respective geographical positions and facilitating data collection. This process is particularly challenging in scenarios with network disruptions and high data traffic density. The main objective of this work is to develop and evaluate a solution based on the MQTT protocol for the efficient and reliable transmission of microplastic images, even in restricted connectivity scenarios, thereby contributing to the advancement of environmental monitoring and citizen science. Edge computing is employed to bring data processing closer to the source devices, reducing latency and improving overall performance. The study encompasses the adaptation of PlanktonScope, the implementation of the MQTT protocol for transmitting large data packets, and the development of modules for image partitioning and recovery. The results indicate that the proposed network is feasible for real-time microplastic monitoring, offering a technological solution for environmental management and an efficient and scalable platform for citizen science. This work highlights the importance of emerging technologies in preserving aquatic ecosystems and promoting sustainable practices.

Keywords: Digital Twins, IoT, Microplastics, Environmental Monitoring, MQTT, Message queue protocol, PlanktonScope, Citizen science.

Este projeto de mestrado resultou em publicações, com revisão por pares:

- Publicações em conferências
 - **22nd IEEE International Conference on Industrial Informatics (INDIN)**
 - * **Título:** On Efficient Data Sharing for Planetary Digital Twins: Distributed Microplastic Monitoring.
 - * **Autores:** Everson Flores, Thiago Teixeira, Paula Barros, Bruna Guterres, Thomaz Pereira Junior, Alberto Cabral, Marcelo Malheiros, Cristiana Lima Dora, Luis Poersch, Wilson Wasielesky, Marcelo Rita Pias.
- Datasets públicos
 - **EU Open Research Repository - Zenodo (2024)**
 - * **Título:** Microplastic Dataset: Supporting microplastic monitoring based on cost-effective open-hardware solutions.
 - * **Autores:** Flores, E., Teixeira, T., Barros, P., Guterres, B. de V., Malheiros, M., Dora, C. L., Orazi, G., Khelifi, I., Poersch, L., Wasielesky, W., & Pias, M. R. .
 - * <https://doi.org/10.5281/zenodo.13836144>
- Publicações de resumos estendidos
 - **WORLD AQUACULTURE Society - AQUA 2024**
 - * **Título:** Planktoscope-based technology: addressing micro-scale monitoring needs in IMTA systems.
 - * **Autores:** M. Pias, B. Guterres, G. Orazi,, P. Drews, V. Oliveira, S. Botelho, N. Duarte, L. Poersch, W. Wasielesky, M. Malheiros, T. Teixeira, C. Dora, P. Barros, E.Flores and E. Ravagnan.
 - * <https://www.was.org/Meeting/Program/PaperDetail/163399>
- Publicação submetida
 - **16th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)**
 - * **Título:** Reliability of Image Transmission in IoT-based Pub/Sub Systems.
 - * **Autores:** Everson Flores, Marcelo Pias, Bruna Guterres, Thomaz Junior, Alberto Cabral, Marcelo Malheiros and Cristiana Lima Dora.

Lista de ilustrações

Figura 1 – Erosão e fragmentação de objetos de plástico ao longo do tempo .	22
Figura 2 – Alcance da ingestão de plástico e seus resíduos em espécies marinhas.	22
Figura 3 – Componentes de um DT	24
Figura 4 – Os três pilares do <i>Digital Twin</i>	26
Figura 5 – Arquitetura de IoT integrada com Edge Computing	27
Figura 6 – Representação do modelo Pub/Sub	30
Figura 7 – Diagrama de comunicação MQTT	32
Figura 8 – Visão geral dos principais componentes do PlanktoScope.	38
Figura 9 – Arquitetura global de comunicação por mensagens MQTT entre diferentes tipos de nós em uma rede distribuída	49
Figura 10 – Rede federada de nós de processamento distribuído com orques- tração MQTT	50
Figura 11 – Diagrama do sistema distribuído (Testbed)	56
Figura 12 – Setup para experimentação montado no laboratório de hardware do Centro de Ciências Computacionais (C3)	57
Figura 13 – Microscópio digital PlanktoScope, configuração para coleta de amostras de microplásticos.	57
Figura 14 – Dispositivos utilizados para os testes do módulo óptico	59
Figura 15 – Conjunto de dados: amostras de microplásticos segmentadas . . .	61
Figura 16 – Servidor utilizado no trabalho com especificação: (a) Processador Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz, com 4 núcleos e 4 <i>threads</i> , (b) Memória RAM (média) de 5.7 Gb disponível e 3.1 Gb livre, e (c) Armazenamento Swap de 2 Gb, não utilizados.	62
Figura 17 – Diagrama de transição de momentos no método V2	68
Figura 18 – Diagrama dos momentos na comunicação de envio entre <i>NodePro-</i> <i>ductor</i> e <i>NodeOrquestrator</i> , e entre <i>NodeOrquestrator</i> e o <i>NodeHybrid</i> . . .	69
Figura 19 – Topologia do ambiente de rede (Testbed)	71
Figura 20 – Cartão de filme usado para calibração das lentes	74
Figura 21 – Imagens dos testes com a mudança no setup das câmeras e lentes	75
Figura 22 – Desempenho da transferência de dados com arquivos de 20 imagens e intervalos de 2 segundos	78
Figura 23 – Desempenho da transferência de dados com arquivos de 50 imagens e intervalos de 2 segundos	79
Figura 24 – Desempenho da transferência de dados com arquivos de 100 ima- gens e intervalos de 2 segundos	80

Figura 25 – Desempenho da transferência de dados com arquivos de 20 imagens e intervalos de 1 segundos	81
Figura 26 – Desempenho da transferência de dados com arquivos de 50 imagens e intervalos de 1 segundos	82
Figura 27 – Desempenho da transferência de dados com arquivos de 100 imagens e intervalos de 1 segundos	83
Figura 28 – Desempenho da transferência de dados com arquivos de 445 imagens e intervalos de 2 segundos	84
Figura 29 – Desempenho da transferência de dados com arquivos de 445 imagens e intervalos de 1 segundos	85
Figura 30 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, sem interrupções de Internet e sem injeção artificial de pacotes	90
Figura 31 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com interrupções de Internet e sem injeção artificial de pacotes	93
Figura 32 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com alteração de nome de arquivo, nome de pasta e com repetição . .	95
Figura 33 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com interrupções de Internet e com injeção artificial de pacotes	97
Figura 34 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com solicitações de pacotes de imagens realizadas por <i>NodeHybrid</i> ao <i>NodeProductor</i>	98
Figura 35 – Fluxograma dos módulos de comunicação e suas interações: componentes do sistema	126
Figura 36 – Fluxograma dos módulos de comunicação e suas interações: Momento 1	127
Figura 37 – Fluxograma dos módulos de comunicação e suas interações: <i>ImageSplitter</i>	127
Figura 38 – Fluxograma dos módulos de comunicação e suas interações: Cabeçalho e fragmentos de imagem	128
Figura 39 – Fluxograma dos módulos de comunicação e suas interações: Envio de imagens	128
Figura 40 – Fluxograma dos módulos de comunicação e suas interações: Momento 2	129

Figura 41 – Fluxograma dos módulos de comunicação e suas interações: Chegada das partes	129
Figura 42 – Fluxograma dos módulos de comunicação e suas interações: Restauração	130
Figura 43 – Fluxograma dos módulos de comunicação e suas interações: Momento 3	130
Figura 44 – Fluxograma dos módulos de comunicação e suas interações (V2): componentes do método V2	131
Figura 45 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 1	131
Figura 46 – Fluxograma dos módulos de comunicação e suas interações (V2): <i>ImageSplitter</i>	132
Figura 47 – Fluxograma dos módulos de comunicação e suas interações (V2): Cabeçalho e fragmentos de imagem	132
Figura 48 – Fluxograma dos módulos de comunicação e suas interações (V2): Identificador único	133
Figura 49 – Fluxograma dos módulos de comunicação e suas interações (V2): Cabeçalho	133
Figura 50 – Fluxograma dos módulos de comunicação e suas interações (V2): Envio do cabeçalho	134
Figura 51 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 2	134
Figura 52 – Fluxograma dos módulos de comunicação e suas interações (V2): Extração de <i>hash</i>	135
Figura 53 – Fluxograma dos módulos de comunicação e suas interações (V2): <i>File Status</i>	135
Figura 54 – Fluxograma dos módulos de comunicação e suas interações (V2): Fim do momento 2	136
Figura 55 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 3	136
Figura 56 – Fluxograma dos módulos de comunicação e suas interações (V2): Atualização do "file status"	137
Figura 57 – Fluxograma dos módulos de comunicação e suas interações (V2): Envio dos fragmentos de imagem	137
Figura 58 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 4	138
Figura 59 – Fluxograma dos módulos de comunicação e suas interações (V2): <i>ImageRestore</i>	138

Figura 60 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Restauração	139
Figura 61 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Fim do momento 4	139
Figura 62 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Solicitações de reenvios	140
Figura 63 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Momento 6	140
Figura 64 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Busca das partes	141
Figura 65 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Fim do momento 6	141
Figura 66 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Momento 7	142
Figura 67 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Atualização de status	142
Figura 68 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Fim do momento 7	143
Figura 69 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Recebimento de sucesso	143
Figura 70 – Fluxograma dos módulos de comunicação e suas interações (V2):	
Atualização de status e limpeza	144

Lista de tabelas

Tabela 1 – Características de cada nível de QoS	36
Tabela 2 – Recursos de segurança do Eclipse Mosquitto	37
Tabela 3 – Resumo dos trabalhos relacionados sobre tecnologia MQTT aplicada ao envio de imagens	45
Tabela 4 – Tabela simplificada do processamento e gerenciamento de imagens por meio do MQTT	53
Tabela 5 – Detalhes dos pacotes de imagens para testes de envio por MQTT	61
Tabela 6 – Tabela de parâmetros e métricas de desempenho conforme o <i>iPerf3</i>	72
Tabela 7 – Comparativo entre câmeras e lentes na avaliação das imagens . .	76
Tabela 8 – Análise de eficiência na transferência de dados: relação entre tamanho das partes, taxa de sucesso e tempo de transferência . .	86
Tabela 9 – Resultados das coletas: relação entre quantidade de imagens, tempo total e reenvio de partes	88
Tabela 10 – Parâmetros utilizados nos experimentos 001 a 008 no cenário 1 .	90
Tabela 11 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 001 a 008	91
Tabela 12 – Parâmetros utilizados nos experimentos 009 a 012 no cenário 2 .	92
Tabela 13 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 009 a 012	92
Tabela 14 – Parâmetros utilizados nos experimentos 013 a 015 no cenário 3 .	94
Tabela 15 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 013 a 015	94
Tabela 16 – Parâmetros utilizados nos experimentos 016 a 019 no cenário 4 .	95
Tabela 17 – Mapa da rede de transmissão de pacotes entre cliente/servidor configurado com o uso de <i>iPerf3</i>	96
Tabela 18 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 016 a 018 com desconexão	97
Tabela 19 – Descrição do fluxo de solicitação de imagens com uso do <i>NodeHybrid</i> ao <i>NodeOrquestrator</i> no experimento 020	98
Tabela 20 – Vantagens e Desvantagens da Computação Federada	113
Tabela 21 – Vantagens e Desvantagens da Arquitetura Federada	118
Tabela 22 – Exemplos de Entidades Federadas em Diferentes Contextos	122
Tabela 23 – Vantagens e Desvantagens das Entidades Federadas	124

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
CAE	<i>Engenharia Auxiliada por Computador</i>
CFD	<i>Dinâmica de Fluidos Computacional</i>
CPU	<i>Unidade Central de Processamento</i>
CPS	<i>Sistemas Ciber-Físico</i>
DE	<i>Digital Ecosystems</i>
DITEN	<i>Redes de Borda de Gêmeos Digitais</i>
DLT	<i>Tecnologia de Ledger Distribuído</i>
DT	<i>Digital Twin</i>
ECG	<i>Eletrocardiogramas</i>
ECUs	<i>Unidades de Controle Eletrônico</i>
EDT	<i>Earth Digital Twins</i>
ESB	<i>Enterprise Service Bus</i>
FEM	<i>Métodos de Elementos Finito</i>
FML	<i>Aprendizado de Máquina Federado</i>
HABs	<i>Harmful Algae Blooms</i>
HMI	<i>Human-Machine Interface</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IA	<i>Inteligência Artificial</i>
IIoT	<i>Internet das Coisas da Indústria</i>
IoT	<i>Internet das Coisas</i>
IP	<i>Protocolo de Internet</i>
IVHM	<i>Sistema Integrado de Gerenciamento de Saúde do Veículo</i>

JSON	<i>JavaScript Object Notation</i>
M2M	<i>Machine-to-Machine</i>
M2S	<i>Machine-to-Server</i>
MBSE	<i>Ferramentas de engenharia de sistemas baseados em modelo</i>
MQ	<i>Message Queuing</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
MP	<i>Megapixel</i>
MP's	<i>Microplásticos</i>
MSMQ	<i>Microsoft Message Queuing</i>
NOVAAS	<i>NOVA Asset Administration Shell</i>
não-IID	<i>Não Independentes e Identicamente Distribuídos</i>
PA	<i>Product Avatar</i>
P2P	<i>Peer-to-Peer</i>
QoS	<i>Qualidade de Serviço</i>
SDK	<i>Kit de Desenvolvimento de Software</i>
SDG	<i>Metas de Desenvolvimento Sustentável</i>
SOA	<i>Arquitetura Orientada a Serviços</i>
S2S	<i>Server-to-Server</i>
TCP	<i>Transmission Control Protocol</i>
TI	<i>Tecnologia da Informação</i>
VPH	<i>Fisiologia Virtual Humana</i>

Sumário

1	INTRODUÇÃO	17
1.1	Objetivos	19
2	FUNDAMENTAÇÃO TEÓRICA	21
2.1	Microplásticos (MPs)	21
2.2	<i>Digital Twin</i>	23
2.3	Edge Computing	26
2.4	Sistemas Distribuídos	28
2.5	Message Queuing Telemetry Transport (MQTT)	31
2.6	Internet das Coisas (IoT)	32
2.7	Eclipse Mosquitto: Broker MQTT para Sistemas Distribuídos	34
2.8	PlanktoScope	37
3	TRABALHOS RELACIONADOS	40
3.1	Desafios, problemas e limitações na implementação de redes baseadas em MQTT	40
3.2	Síntese dos trabalhos pesquisados	44
4	DESENVOLVIMENTO	48
4.1	Arquitetura da rede federada	48
4.1.1	Agentes e componentes do sistema	49
4.1.2	Componentes para transmissão	51
4.1.3	Interação entre os módulos	52
4.1.4	Resumo dos agentes e componentes do sistema	53
4.1.5	Cabeçalho de imagens	53
4.1.6	Tópicos de comunicação	54
4.2	Implementação e ambiente experimental Testbed	56
4.2.1	Ambiente usado para experimentação	56
4.2.2	Geração de dataset	57
4.2.3	Módulo óptico	58
4.2.4	Produção das amostras utilizadas no Dataset	60
4.2.5	Ambiente computacional	61
4.2.6	Servidor	62
4.2.7	Hardware utilizado	62
4.2.8	Conexão de rede	64
4.2.9	Broker Mosquitto	64

4.2.10	Protocolo MQTT	65
4.3	Método de transmissão e reconstrução de imagens	65
4.3.1	Primeiro método (V1)	65
4.3.2	Segundo método (V2)	66
4.3.3	Metodologia V2 aplicada nos nós	68
4.3.4	Implementação do protocolo MQTT	69
4.4	Implementação de ambiente de rede controlada (<i>Testbed</i>)	70
4.4.1	Topologia da rede	70
4.4.2	Equipamentos utilizados	71
4.4.3	Recursos computacionais utilizados	71
4.4.4	Protocolo de comunicação	72
4.4.5	Ferramenta utilizada	72
4.4.6	Parâmetros utilizados	72
4.4.7	Coleta de dados	73
5	RESULTADOS E DISCUSSÃO	74
5.1	Sistema PlanktoScope ajustado	74
5.2	Transferência de arquivos entre os nós	77
5.3	Resultados da validação em ambiente de rede controlada	89
6	CONSIDERAÇÕES FINAIS	100
6.1	Sugestões para trabalhos futuros	101
	REFERÊNCIAS	102
	APÊNDICE A – COMPUTAÇÃO FEDERADA	110
A.1	Aplicações	111
A.2	Vantagens e Desvantagens da Computação Federada	111
	APÊNDICE B – ARQUITETURA FEDERADA	114
B.1	Aplicações	115
B.2	Vantagens e Desvantagens da Arquitetura Federada	116
B.3	Entidades Federadas	119
	APÊNDICE C – FLUXOGRAMA DE COMUNICAÇÃO	126
C.1	Fluxogramas do método V1	126
C.2	Método V2	130

1 Introdução

Embora o destino de grande parte de microplásticos flutuantes permaneça desconhecido, estes podem prejudicar a vida marinha por diferentes vias, como por exemplo ao liberar compostos tóxicos e propagar micro-organismos que podem ser nocivos [Egger, Sulu-Gambari e Lebreton 2020]. Neste sentido, há uma necessidade de monitorar microplásticos dentro dos ecossistemas aquáticos. Este monitoramento torna-se importante para melhorar a avaliação de risco, criar ações de mitigação nos estágios iniciais e investigar a dinâmica do microplástico dentro de ecossistemas aquáticos.

Os microplásticos (MPs) receberam atenção mundial após relatórios sobre enormes ilhas de lixo dentro de todas as giras oceânicas [Hale et al. 2020] sugerindo que MPs estão poluindo estes ecossistemas em uma escala ainda não vista [Egger, Sulu-Gambari e Lebreton 2020, Connan et al. 2021]. A produção anual de plástico atingiu 367 milhões de toneladas em 2020 [Europe 2021], o que leva ao aumento da poluição uma vez que aproximadamente 80% deste material é depositado em aterros sanitários ou diretamente em ambientes aquáticos [Geyer, Jambeck e Law 2017]. Além de permanecer por décadas no meio ambiente, os plásticos continuam se decompondo em pedaços cada vez menores.

Vários estudos têm se utilizado de equipamentos de citometria de fluxo para detectar microplásticos em amostras de água [Kaile et al. 2020], [Park et al. 2020], [Wang et al. 2021]. Imaging FlowCytobots (IFCB) [Olson e Sosik 2007] e *FlowCam* [Buskey e Hyatt 2006, Yokogawa 2023] são equipamentos de imagem de alta produtividade e de fluxo altamente automatizados. Entretanto, eles são equipamentos de alto custo para análises em escala e de dimensões físicas que necessitam de uma bancada de laboratório, impossibilitando desta forma o seu uso em aplicações de baixo custo e de grande escala. Por sua vez, o PlanktonScope é uma plataforma de citometria de fluxo modular acessível e adaptada para o conceito de *oceanografia cidadã* (ou *citizen science*, em inglês). Trata-se de um equipamento de baixo custo que pode ser facilmente adaptado a muitas aplicações de campo incluindo a avaliação da presença de microplástico [Pollina et al. 2020].

O PlanktonScope, se adaptado com software específico, pode permitir ao usuário a avaliação de microplásticos em diferentes localidades, vinculando suas localizações obtidas aos dados coletados, por exemplo, por meio de sistemas de GPS. Entretanto, um *feedback* inteligente com *loop* fechado e um controle em tempo real são aspectos importantes para um monitoramento e uma avaliação dinâmica. Segundo

[Society 2020], os *Digital Twins* (DTs) Planetários exploram o *feedback* e o *loop* de controle para construir uma réplica virtual de objetos, instalações e até mesmo pessoas sendo assim uma ferramenta de suporte que pode também utilizada na aplicação de monitoramento de microplásticos. A interoperabilidade e comunicação destes elementos de um sistema de monitoramento baseado em *twins* garantem que os dados coletados sejam processados e a informação entregue ao usuário. Para tal, utiliza-se a Internet das Coisas (IoT) que estabelece a uma rede de dispositivos físicos equipados com tecnologia embarcada, sensores e atuadores, conectados à Internet para troca de dados. O IoT possibilita que sistemas interconectados compartilhem informações de sensores, modelos de Inteligência Artificial (IA), simulações e estratégias de *feedback* de controle fechado ¹ [Yassein et al. 2017]. O avanço da IoT tem gerado aumento no volume de dados em setores como manufatura, saúde e cidades inteligentes. Protocolos como o *Message Queuing Telemetry Transport* (MQTT) são elementos importantes dentro do contexto de confiabilidade para a transmissão dos dados em redes IoT, em que os dispositivos integrantes tem limitações de recursos [Yassein et al. 2017]. Além disso, a integração da IoT com sistemas de análise de dados viabiliza aplicações como manutenção preditiva, monitoramento em tempo real e a conexão de dispositivos heterogêneos [Nazir e Kaleem 2019].

O protocolo MQTT foi desenvolvido para viabilizar a comunicação em dispositivos de poucos recursos computacionais e em ambientes com baixa largura de banda para comunicação de dados [Alam e Akram 2016]. Originalmente utilizado em aplicações industriais e de automação residencial, o MQTT consolidou-se como uma solução utilizada em sistemas de IoT, devido à sua eficiência e escalabilidade no modelo de comunicação de publicação e assinatura por tópicos de interesse (*Pub/Sub*) [Leo et al. 2014].

Mais recentemente, o MQTT tem sido empregado em cenários desafiadores, como no envio de pacotes de dados maiores, incluindo imagens de alta resolução coletadas por dispositivos científicos e sensores ambientais. A simplicidade operacional, torna este protocolo adequado para aplicações em ambientes com largura de banda limitada para comunicação de dados [Hwang, Park e Shon 2016]. No modelo *Pub/Sub*, a comunicação assíncrona permite a distribuição de dados para múltiplos assinantes (geralmente anônimos), e desta forma, tornando a arquitetura do sistema distribuído mais escalável [NATIVI e CRAGLIA 2020]. Este protocolo não apenas suporta a troca de informações, mas também contribui para a criação de plataformas que integram sensores, modelos de IA e estratégias de controle [Alam e Akram 2016, Teixeira et al. 2022]. O escopo de uso de sensores é amplo, e engloba dados coletados multi-modal,

¹ *feedback* de controle fechado é uma técnica que monitora continuamente a saída de um sistema, compara-a com um valor de referência e ajusta sua entrada em tempo real para minimizar os erros. [Pan et al. 2024]

variando desde séries temporais de uma variável (ex. sensor de temperatura para ambientes) até dados multi-mídia como imagens e *feeds* de vídeo. Por exemplo, o MQTT alçado no seu modelo Pub/Sub pode ser empregado na distribuição de imagens para múltiplos assinantes, permitindo a integração de diferentes sistemas com acesso em tempo real aos dados de sensores incluindo câmeras dedicadas à captura de imagens de microplásticos. Entretanto, ainda é necessário garantir a confiabilidade na distribuição destas imagens neste tipo de sistema distribuído.

Para mitigar esses problemas relacionados à transmissão confiável de imagens em sistemas Pub/Sub, este trabalho propõe o uso de *Digital Twins e IoT* para o monitoramento baseado em sensores de microplásticos [Society 2020, Nazir e Kaleem 2019, Kaile et al. 2020]. A solução proposta utiliza o PlanktonScope, um dispositivo de hardware modular e *open-source*, que integrado ao protocolo MQTT para comunicação de dados, oferece funcionalidade de coleta, transmissão e processamento de imagens de microplásticos em redes com baixa largura de banda para comunicação de dados [Alam e Akram 2016, Hwang, Park e Shon 2016]. Baseada em computação em borda, a abordagem busca reduzir a latência, otimizar recursos e possibilitar o monitoramento em tempo real [Yassein et al. 2017, Cao et al. 2020].

1.1 Objetivos

O presente trabalho tem por objetivo geral:

- Desenvolver e avaliar uma solução baseada no protocolo MQTT para a transmissão confiável de imagens em alta resolução provenientes de uma aplicação de monitoramento de microplásticos, mesmo em cenários de conectividade limitada, contribuindo para o avanço do monitoramento ambiental e da ciência aberta e cidadã.

Para isso, são identificados abaixo os objetivos específicos:

1. Investigar o uso de MQTT para envio de pacotes de dados com tamanho maior, contendo imagens de microplásticos coletadas por meio do hardware aberto PlanktoScope;
2. Desenvolver módulos e rotinas capazes de particionar imagens transmitidas via MQTT;
3. Avaliar o desempenho do protocolo MQTT na transmissão confiável em redes de baixa largura de banda de pacotes de dados contendo imagens de alta resolução;

4. Avaliar o desempenho da rede de comunicação para as imagens em cenários experimentais controlados.

Este trabalho contribui para o avanço do conhecimento acerca do compartilhamento de imagens em redes com conexões de baixa qualidade. A contribuição secundária do trabalho é a criação de uma metodologia para desenvolvimento de aplicações voltadas para o monitoramento de microplásticos com baixo custo, e permitindo uma escala além da banca de laboratório, amplamente utilizada no atual estado da arte. A arquitetura de sistema proposta e validada integra tecnologias emergentes, como IoT e Digital Twins, em aplicações ambientais.

2 Fundamentação Teórica

Esta seção aborda os principais conceitos relacionados ao estudo, iniciando com os Microplásticos (MPs) (Seção 2.1), abrangendo sua origem, classificação e os impactos nos ecossistemas aquáticos. Em seguida, a Seção 2.2 apresenta o conceito de Digital Twin (DT), definido como réplicas digitais de ativos físicos, com destaque para suas aplicações na otimização de processos. Introduz os conceitos de *Edge Computing* (Seção 2.3) e Sistemas Distribuídos (Seção 2.4), com ênfase em eficiência, escalabilidade e segurança, além do protocolo MQTT (Seção 2.5), uma solução leve para comunicação em Internet das Coisas (IoT), amplamente utilizado em sistemas distribuídos e na IoT (Seção 2.6), com referência à implementação no Eclipse Mosquitto (Seção 2.7). Por fim, discute o PlanktoScope (Seção 2.8), uma plataforma voltada à análise em tempo real de microplásticos, integrando essas tecnologias às demandas do monitoramento ambiental.

2.1 Microplásticos (MPs)

Os plásticos são polímeros sintéticos produzidos para uso humano. Registros de microplásticos começaram a ser coletados na década de 1970. Os primeiros registros foram feitos no nordeste do Oceano Pacífico, em meados dos anos 1970 [Nyadjro et al. 2023]. Itens de plástico, como roupas, esponjas, garrafas e luvas são comumente usados no dia a dia. Os plásticos podem ser dispersos no meio ambiente como resíduos e contaminar indiscriminadamente a água, o solo e o ar [Cera, Cesarini e Scalici 2020]. Segundo Horton et al. 2017 a contaminação por plásticos é amplamente difundida em todo o mundo, sendo considerada um dos principais problemas ambientais e de gestão de recursos hídricos. Isso motivou pesquisadores e especialistas na área de proteção ambiental a aumentar e melhorar as atividades de monitoramento com o intuito de avaliar os impactos dos plásticos no meio ambiente. Itens de plástico, que podem ser de diferentes formas de polímeros, podem ser classificados de acordo com o seu tamanho. Em particular, itens que variam entre $1\ \mu m$ e $5\ mm$ são chamados de microplásticos [Gigault et al. 2018].

Os microplásticos (MPs) podem ser classificados como primários ou secundários, dependendo da forma como são produzidos. MPs primários são pequenas partículas plásticas de tamanho microscópico, liberadas diretamente no meio ambiente por meio de efluentes domésticos e industriais, derramamentos e descarga de esgoto [Sá et al. 2018]. Eles incluem microesferas oriundos de produtos para cuidado pessoal e também provenientes de abrasivos industriais para superfícies delicadas.

Essas microesferas também são utilizadas em agentes de limpeza, revestimentos e tintas, fluidos de perfuração na indústria de petróleo e gás, e como resinas precursoras e *pellets* para a fabricação de produtos plásticos acabados [Hale et al. 2020]. Os MPs secundários são derivados da fragmentação de plásticos maiores, já presentes no ambiente, devido, por exemplo, à radiação UV (foto-oxidação), transformação mecânica por meio de abrasão física (por exemplo, partículas de desgaste de pneus) e degradação biológica por microorganismos [Ward et al. 2019].

A erosão e fragmentação de objetos de plástico levam a um número crescente de microplásticos conforme ilustrado na Figura 1. Dessa forma, partículas de plástico ficam biodisponíveis para um grupo crescente de espécies. Por exemplo, na Figura 2, os objetos e as partículas maiores podem estar biodisponíveis por meio da ingestão para peixes maiores. As partículas menores estão biodisponíveis para pequenos organismos e também para os maiores, por meio da bioacumulação [Koelmans et al. 2022].

Figura 1 – Erosão e fragmentação de objetos de plástico ao longo do tempo



Fonte: Adaptado de [Koelmans et al. 2022].

Figura 2 – Alcance da ingestão de plástico e seus resíduos em espécies marinhas.



Fonte: Adaptado de [Koelmans et al. 2022].

A identificação de microplásticos no ambiente apresenta um desafio multi-dimensional que ainda não foi atacado de forma satisfatória. Sua complexidade é paralela à da matéria orgânica particulada que ocorre naturalmente [Hoellein et al. 2019]. MPs foram encontrados nos oceanos, em água doce, em sedimentos, em solos e no ar [Blackburn e Green 2021]. Esses fragmentos de plástico de natureza micrométrica (ou até nanométrica) são persistentes no ecossistema marinho sendo facilmente confundidos como alimento por uma variedade de biota marinha. Dentro desta diversidade pode-se citar os corais, os fitoplânctons, os zooplânctons, os ouriços-do-mar, as lagostas, os peixes [Chatterjee e Sharma 2019]. Dessa forma, detectar a presença de MPs e seus efeitos biológicos adversos torna-se uma tarefa importante e desafiadora. Determinar o nível de prejudicialidade atrelada à poluição por MPs nas áreas costeiras e oceano aberto é de suma importância. Isso pode ser alcançado considerando a probabilidade de ocorrência e impacto dos riscos associados. Dessa forma, há uma necessidade de desenvolver soluções para monitoramento de microplásticos de forma abrangente. Tais soluções devem contemplar cenários de vida selvagem, habitats naturais e saúde humana [Shim e Thomposon 2015].

2.2 *Digital Twin*

Esta seção apresenta o conceito de *Digital Twin* (DT), suas definições principais, aplicações práticas e vantagens e desvantagens.

Conceito

Um DT é uma réplica virtual de um ativo físico que permite monitorar, prever e otimizar seu desempenho por meio de simulações e análises contínuas [Society 2020]. Ele opera com um *loop* de feedback entre o ativo físico e sua representação digital, utilizando dados em tempo real para realizar ajustes e prever falhas [Madni, Madni e Lucero 2019, Barricelli, Casiraghi e Fogli 2019].

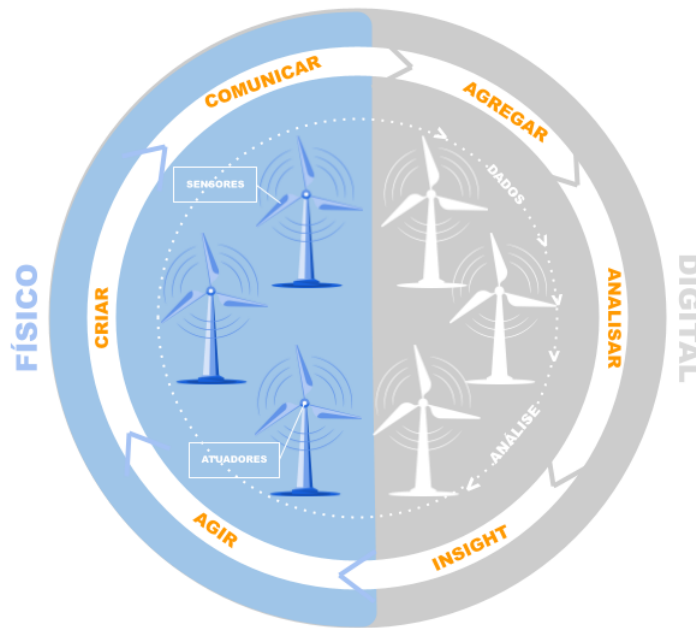
O DT combina modelos computacionais, simulações e inteligência artificial para evoluir junto com o ativo físico, refletindo alterações no sistema e possibilitando o controle e a manutenção preditiva [Barricelli, Casiraghi e Fogli 2019, Zheng, Yang e Cheng 2019]. Além disso, sua estrutura modular facilita a integração com novas tecnologias e adaptação a mudanças operacionais [Liu, Meyendorf e Mrad 2018, Bauer et al. 2021].

De acordo com [Cheng et al. 2018] e [Bauer et al. 2021], os DTs também suportam processos de otimização e autoajuste em sistemas complexos, tornando-se uma ferramenta indispensável em diversas indústrias e áreas de pesquisa.

Componentes de um DT

Os componentes de um DT, ilustrados na Figura 3, incluem sensores para coleta de dados, atuadores para ações físicas, dados para alimentar modelos, simulações para prever comportamentos e visualizações intuitivas. Este conjunto de elementos permite monitorar e otimizar ativos físicos em tempo real [Society 2020].

Figura 3 – Componentes de um DT



Fonte: Adaptado de [Society 2020].

Outras definições

Os *Digital Twins* (DTs) são descritos na literatura como réplicas digitais de sistemas físicos, que combinam dados em tempo real, modelos e simulações para monitorar, prever e otimizar desempenho. Algumas definições destacam características específicas:

- [Ríos et al. 2015] define DTs como modelos que simulam a vida de entidades físicas, podendo representar objetos, processos ou recursos humanos.
- Segundo [Barricelli, Casiraghi e Fogli 2019], um DT é um modelo em constante evolução, utilizado para prever defeitos e testar configurações, integrando dados em tempo real e *big data*.
- Para [Glaessgen e Stargel 2012], um DT é uma simulação integrada multifísica e multiescala, que reflete o ciclo de vida de um sistema físico, como veículos aeroespaciais.

- [Madni, Madni e Lucero 2019] enfatiza que DTs utilizam simulações e modelos para explorar cenários e atualizar continuamente sua réplica digital com dados do ativo físico.
- De acordo com [Schluse e Rossmann 2016], DTs estão alinhados com conceitos da Indústria 4.0 ¹, sendo representações virtuais detalhadas de objetos ou sistemas do mundo real.
- [Fuller et al. 2020] define um DT como um sistema integrado, onde alterações no objeto físico ou digital são automaticamente refletidas no outro.

Essas definições mostram a ampla aplicabilidade e flexibilidade dos DTs, que vão desde a manutenção preditiva até a otimização de sistemas complexos em tempo real.

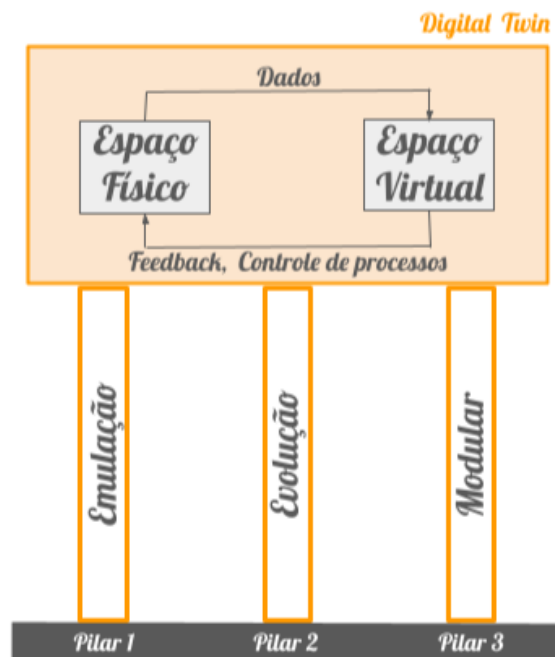
Os Três Pilares dos DTs

A arquitetura dos *Digital Twins* (DTs) fundamenta-se em três pilares principais: emulação, evolução e modularidade. Esses elementos são cruciais para o desenvolvimento e a operação dos DTs, garantindo sua capacidade de representar com precisão ativos físicos em sistemas complexos [Pias et al. 2023, Lima, Rocha e Roque 2019].

- **Emulação:** Consiste na habilidade do DT de reproduzir, em tempo real, o comportamento do ativo físico. Essa funcionalidade permite simulações com base em condições atuais e históricas, apoiando decisões relacionadas à manutenção e intervenções.
- **Evolução:** Relaciona-se à capacidade do DT de se adaptar ao longo do tempo, acompanhando alterações no ativo físico. Essa adaptabilidade é alcançada por meio de aprendizado contínuo, atualizações de modelos e análise baseada em inteligência artificial.
- **Modularidade:** Refere-se à estrutura flexível e organizada em componentes, que facilita a integração, manutenção e expansão do DT com novas tecnologias e funcionalidades.

Esses pilares permitem que os DTs operem como réplicas dinâmicas e adaptáveis, promovendo melhorias contínuas no desempenho e na manutenção de ativos.

¹ Indústria 4.0 é definida como uma revolução tecnológica impulsionada por tecnologias como a IoT, computação em nuvem e IA, que facilitam a realização de sistemas ciberfísicos inteligentes (Smart Cyber-Physical Systems) [Mourtzis, Angelopoulos e Panopoulos 2022]

Figura 4 – Os três pilares do *Digital Twin*

Fonte: Adaptado de [Pias et al. 2023].

A adoção dessa abordagem amplia a aplicabilidade dos DTs em diversas áreas, como gestão de ativos industriais e projetos de grande escala, reforçando a relevância da tomada de decisão baseada em dados.

Aplicações dos DTs

Os DTs têm aplicação em áreas como aviação (manutenção preditiva), manufatura (otimização de processos), saúde (simulações personalizadas) e ecossistemas globais (monitoramento ambiental). Exemplos incluem iniciativas como *Destination Earth*, que visa criar um DT da Terra para integrar dados e promover sustentabilidade [Nativi, Mazzetti e Craglia 2021, Teixeira et al. 2022].

Representam uma nova dimensão no monitoramento e otimização de sistemas, destacando-se como ferramentas essenciais em tecnologias emergentes. Integrados ao conceito de Edge Computing, os DTs potencializam a análise e a tomada de decisão local, reduzindo a latência e aprimorando o desempenho, apesar dos desafios relacionados a custos, dependência de dados precisos e segurança.

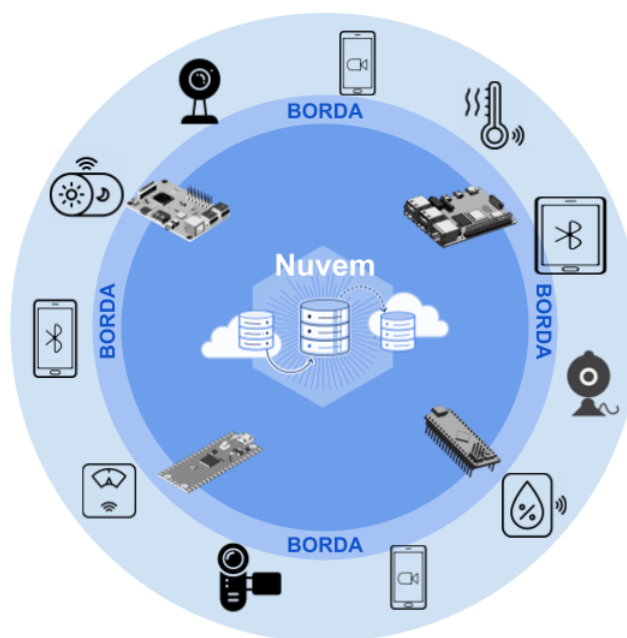
2.3 Edge Computing

Edge Computing, ou Computação de Borda, desloca tarefas computacionais para a borda da rede, reduzindo latência e tráfego enquanto melhora o desempenho.

Essa abordagem descentraliza a computação, supera limitações de dispositivos *front-end*, otimiza o consumo energético e amplia a eficiência [Varghese et al. 2016]. Diferentemente da computação em nuvem, realiza processamento e armazenamento próximos à origem dos dados [Cao et al. 2020].

O conceito surgiu com o crescimento da IoT e a demanda por processamento em tempo real, promovendo menor latência e melhor uso de largura de banda [Khan et al. 2019]. Tecnologias como redes 5G e dispositivos inteligentes impulsionam sua aplicação [Satyanarayanan 2017], permitindo operações locais mesmo com falhas de conexão [Varghese et al. 2016].

Figura 5 – Arquitetura de IoT integrada com Edge Computing



Fonte: o autor

A Figura 5 ilustra como dispositivos na borda, como sensores e microcontroladores, processam dados localmente, reduzindo latências e melhorando a eficiência.

Aplicações

Estratégias de alocação de carga ajustam o processamento de acordo com as necessidades das aplicações, otimizando latência e desempenho em cenários de alta demanda, como redes *fog computing* ² [Jia et al. 2018]. Exemplos incluem IoT, realidade aumentada e análise de vídeo, onde baixa latência e análise em tempo real são cruciais.

² *Fog Computing* é um paradigma computacional que leva recursos e serviços computacionais para a borda da rede, ou seja, mais próximo dos dispositivos do usuário. Isso reduz a latência e conecta os recursos com a computação em nuvem [Costa et al. 2022]

O Edge Computing promove inovação, eficiência e maior resiliência, consolidando-se como uma alternativa relevante em ambientes orientados por dados. Quando integrado a sistemas distribuídos, potencializa a descentralização do processamento e a escalabilidade, permitindo uma integração entre dispositivos e reduzindo a latência nas aplicações, como ocorre em DTs e outras tecnologias emergentes.

2.4 Sistemas Distribuídos

Sistemas distribuídos gerenciam dispositivos conectados e seus dados, processando-os de forma segura e escalável [Cao et al. 2020]. Esses sistemas são essenciais em diversos contextos:

- **Conectividade:** Facilitam a comunicação rápida em sistemas complexos, como casas inteligentes e redes de energia [Society 2020];
- **Edge Computing:** Oferecem processamento local, serviços de localização e suporte à mobilidade, garantindo alta disponibilidade [Khan et al. 2019];
- **Escalabilidade:** Permitem a integração de novos nós sem aumentar significativamente a latência [Varghese et al. 2016].

A importância desses sistemas é detalhada nos Apêndices A e B, com foco em arquiteturas federadas.

Características

- **Escalabilidade:** Ajustam recursos conforme a demanda [Abbas et al. 2017];
- **Resiliência:** Recuperam-se rapidamente de falhas, mantendo a operação [Abbas et al. 2017];
- **Eficiência:** Otimizam o uso de recursos computacionais [Huang et al. 2022].

Message Queuing (MQ)

Message Queuing gerencia mensagens em filas, conectando produtores e consumidores de forma assíncrona. Exemplos incluem *RabbitMQ* e *Kafka*, amplamente utilizados em aplicações como *big data* e automação residencial [Fu, Zhang e Yu 2020, Zhang e Liu 2013].

Publish-Subscribe (Pub/Sub)

O modelo *Publish-Subscribe* (Pub/Sub) é amplamente empregado em sistemas distribuídos por oferecer comunicação assíncrona, desacoplada e escalável. Nesse contexto, os participantes desempenham dois papéis principais: *publishers* (publicadores), que enviam mensagens vinculadas a tópicos específicos, e *subscribers* (assinantes), que recebem somente as mensagens referentes aos tópicos de interesse. Essa estrutura minimiza a dependência direta entre emissores e receptores [Leo et al. 2014].

A comunicação no modelo Pub/Sub é mediada por um **broker**, que gerencia os tópicos e distribui as mensagens. Os *publishers* enviam mensagens para tópicos específicos, enquanto os *subscribers* manifestam interesse ao se inscreverem nesses tópicos. Essa arquitetura é particularmente adequada em contextos com variação dinâmica de emissores e receptores, como em aplicações IoT e sistemas distribuídos [Nazir e Kaleem 2019].

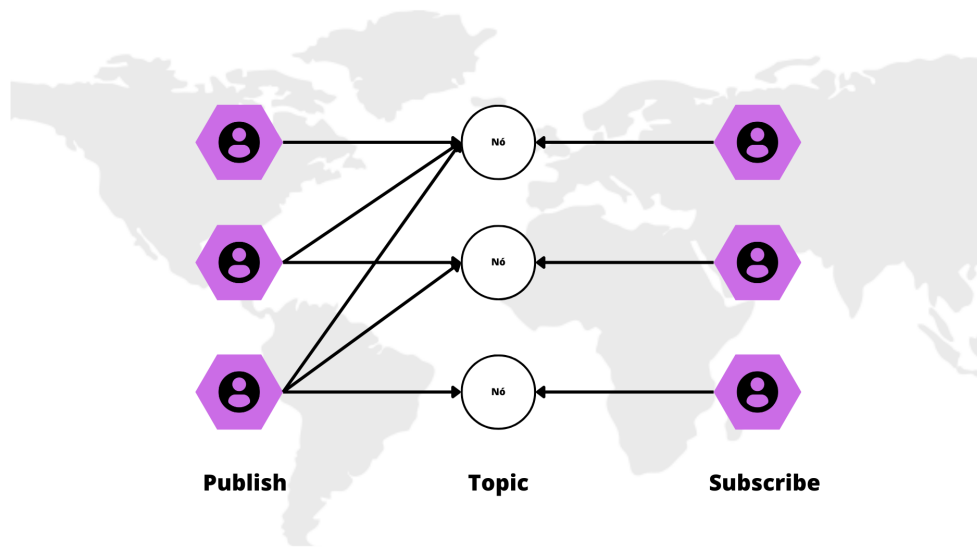
O modelo também oferece flexibilidade no gerenciamento de eventos e dados em tempo real. Em sistemas de gêmeos digitais, por exemplo, o Pub/Sub é utilizado para sincronizar dados entre o modelo digital e o ativo físico [Botín-Sanabria et al. 2022].

Elementos Principais

1. **Publicação (*Publishing*)**: Os *publishers* enviam mensagens relacionadas a eventos ou informações específicas, publicando-as em tópicos apropriados. Por exemplo, sensores podem transmitir dados ambientais para um sistema de gêmeos digitais, que os processa e atualiza o modelo [Lima, Rocha e Roque 2019].
2. **Tópicos e Assinaturas (*Topics and Subscriptions*)**: As mensagens são organizadas em tópicos, que funcionam como categorias lógicas. Os *subscribers* escolhem os tópicos de interesse e recebem mensagens publicadas nesses tópicos [Alam e Akram 2016].

Os publicadores são responsáveis por enviar mensagens ou eventos relacionados a tópicos específicos, enquanto os assinantes se interessam por um ou mais tópicos e se inscrevem para receber as mensagens relacionadas a eles como visto na Figura 6.

Figura 6 – Representação do modelo Pub/Sub



Fonte: o autor

O modelo Pub/Sub apresenta diversos benefícios:

- **Escalabilidade:** Facilita a adição de novos *publishers* e *subscribers* sem impacto nos componentes existentes [Bellini, Nesi e Pantaleo 2022].
- **Desacoplamento:** Reduz a interdependência entre emissores e receptores, permitindo modificações independentes [Nativi, Mazzetti e Craglia 2021].
- **Eficiência:** As mensagens são enviadas apenas aos *subscribers* relevantes, otimizando recursos de rede [Yelamarthi, Aman e Abdelgawad 2017].

Aplicações

As aplicações do modelo Pub/Sub abrangem diversos contextos:

- **Gêmeos Digitais:** Sincronização de dados entre ativos físicos e representações digitais [Leo et al. 2014].
- **Automação Residencial:** Protocolos como MQTT utilizam o Pub/Sub para comunicação em casas inteligentes [Gambi et al. 2018].
- **IoT e Redes Sensoriais:** Integra sensores e atuadores em redes dinâmicas, mesmo com conectividade variável [Nazir e Kaleem 2019].

O Pub/Sub é importante para sistemas distribuídos, permitindo comunicação satisfatória e flexível entre componentes heterogêneos.

2.5 Message Queuing Telemetry Transport (MQTT)

O MQTT (Message Queuing Telemetry Transport) é um protocolo leve baseado no modelo Pub/Sub, projetado para ambientes com largura de banda limitada e dispositivos com recursos restritos. Sua aplicação é amplamente difundida em redes de IoT, devido à sua eficiência na utilização de recursos e simplicidade de implementação [Lima, Rocha e Roque 2019].

O protocolo opera sobre o TCP/IP, garantindo confiabilidade na comunicação. Ele também oferece diferentes níveis de Qualidade de Serviço (QoS), que asseguram a entrega das mensagens de acordo com os requisitos específicos das aplicações [Khan et al. 2019, Leo et al. 2014].

Estudos demonstram diversas aplicações práticas do MQTT. Por exemplo, [Nazir e Kaleem 2019] propôs um sistema de segurança para casas inteligentes, utilizando o protocolo para notificações de eventos e transmissão de pacotes de imagem. Já [Soni e Makwana 2017] descreveu um sistema de monitoramento de saúde em casa que utiliza MQTT para transmitir dados de diagnósticos para análise remota, reduzindo visitas hospitalares. Outro trabalho relevante [Baiocchi et al. 2019] avaliou o desempenho do MQTT em plataformas de baixo custo, como o Raspberry Pi Zero W, analisando o impacto no uso do processador e a capacidade de publicação em diferentes cenários.

Além disso, pesquisas como a de [Yassein et al. 2017] exploraram o uso de gateways inteligentes para integrar diferentes protocolos IoT, enquanto [Hwang, Park e Shon 2016] aprimorou a confiabilidade de transmissão no MQTT por meio de técnicas avançadas de controle de QoS. Comparações entre o MQTT e o HTTP também foram realizadas, indicando menor sobrecarga de rede no MQTT, embora problemas possam surgir com comprimentos extensos de tópicos [Yokotani e Sasaki 2016].

Definição e Características do MQTT

- **Leveza e Simplicidade:** Ideal para dispositivos com recursos limitados devido ao baixo consumo de largura de banda e processamento.
- **Modelo Pub/Sub:** Comunicação descentralizada onde mensagens são publicadas em tópicos e recebidas por dispositivos inscritos.
- **Broker:** Servidor central que gerencia mensagens e facilita a escalabilidade.
- **Qualidade de Serviço (QoS):** Três níveis de entrega garantem flexibilidade nas aplicações.

Protocolos de Transporte do MQTT

- **TCP:** Comunicação orientada à conexão, garantia de entrega de mensagens.
- **WebSocket:** Alternativa para uso em aplicações *web*, permitindo comunicação bidirecional em tempo real, mesmo em redes restritas.

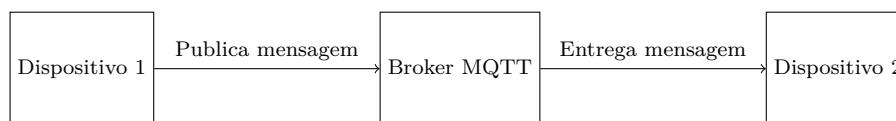
Desafios do MQTT

1. **Perda de Pacotes:** Pode ocorrer em redes congestionadas ou com muitos dispositivos conectados [Nugraha et al. 2024].
2. **Latência:** Problemática em aplicações com alta taxa de amostragem [Yassein et al. 2017].
3. **Sobrecarga de Protocolo:** Relevante em tópicos extensos, mitigável com técnicas de compressão [Yokotani e Sasaki 2016].
4. **Confiabilidade de Mensagens:** Crítica em aplicações sensíveis à ordem de entrega, como automação residencial [Hwang et al. 2017].
5. **Consumo de Recursos:** Exige conexões TCP persistentes, impactando dispositivos com pouca memória ou processamento [Yassein et al. 2017].

Arquitetura do MQTT

A Figura 7 ilustra a arquitetura básica do MQTT, composta por um broker e dispositivos que atuam como publicadores e assinantes.

Figura 7 – Diagrama de comunicação MQTT



Fonte: o autor

O MQTT, com sua arquitetura simples e adaptabilidade, continua sendo uma solução robusta para aplicações IoT, embora seus desafios devam ser cuidadosamente avaliados para otimizar sua implementação.

2.6 Internet das Coisas (IoT)

A Internet das Coisas (IoT) é uma rede de dispositivos físicos interconectados que coletam e trocam dados. Esses dispositivos, que incluem desde sensores simples até

sistemas complexos de automação, promovem a comunicação e o compartilhamento de informações úteis, permitindo automação e controle remoto de processos [Merenda, Porcaro e Iero 2020]. Conforme [Khanna e Kaur 2020], a IoT evoluiu para uma troca de informações entre múltiplos dispositivos físicos.

Os avanços tecnológicos e as expectativas crescentes continuam a impulsionar a pesquisa na área. Nos últimos anos, a IoT tem se relacionado a outras tecnologias, como computação em nuvem e *big data*, aumentando seu impacto [Imteaj et al. 2021]. Ainda segundo [Koohang et al. 2022], a conscientização sobre IoT também envolve compreender os riscos crescentes de segurança e privacidade associados a esses dispositivos.

Evolução da IoT

O conceito de IoT teve início com a ideia de conectar dispositivos físicos à Internet para monitoramento e controle remoto. Com o avanço da comunicação sem fio e a miniaturização de componentes, a IoT expandiu suas aplicações para áreas como saúde, transporte, agricultura e cidades inteligentes. Esse progresso é impulsionado pela demanda por automação e eficiência, assim como pela integração de tecnologias como inteligência artificial e computação em nuvem [Khanna e Kaur 2020].

Componentes e Arquitetura da IoT

Sensores e Atuadores

Sensores captam informações do ambiente, como temperatura e movimento, convertendo-as em sinais processáveis. Atuadores executam ações com base nos dados recebidos, controlando dispositivos como motores e luzes [Laghari et al. 2021, Gomathi et al. 2018, Yelamarthi, Aman e Abdelgawad 2017].

Conectividade

A conectividade é parte importante para o funcionamento e escalabilidade da IoT, permitindo a transmissão de dados. Redes como Wi-Fi, Bluetooth, Zigbee e LoRa são usadas conforme as necessidades de alcance, consumo de energia e volume de dados [Bellini, Nesi e Pantaleo 2022].

Protocolos

Protocolos como MQTT, CoAP e HTTP/HTTPS garantem a comunicação segura entre dispositivos, sendo essenciais para a integração e o desempenho da

IoT [Pereira et al. 2020].

Tecnologias

A IoT emprega tecnologias como comunicação por retroespalhamento, transferência de energia sem fio e dispositivos sem chip para reduzir consumo e custos. Avanços como 5G e inteligência artificial aprimoram a análise de dados e a eficiência dos sistemas [Imteaj et al. 2021].

Aplicações da IoT

A IoT é aplicada em diversos setores:

- **Agricultura de Precisão:** Otimização do uso de água e recursos para aumentar a produtividade [Khanna e Kaur 2020].
- **Monitoramento Industrial:** Controle de processos e manutenção preditiva [Pereira et al. 2020].
- **Casas Inteligentes:** Automação de dispositivos para maior conforto e eficiência energética [Gomathi et al. 2018].

Desafios da IoT

Entre os desafios da IoT estão a gestão de energia e comunicação, além de questões de privacidade e segurança. Tecnologias como retroespalhamento e transferência de energia sem fio são exploradas para minimizar esses problemas [Pereira et al. 2020, Koohang et al. 2022].

2.7 Eclipse Mosquitto: Broker MQTT para Sistemas Distribuídos

O **Eclipse Mosquitto**³ é um *broker* MQTT de código aberto amplamente utilizado, desenvolvido pela Eclipse Foundation. Ele implementa o protocolo MQTT para viabilizar a comunicação entre sensores, dispositivos e aplicações em arquiteturas de publicação/assinatura. Essa abordagem oferece maior flexibilidade em comparação ao padrão cliente/servidor, facilitando a troca de mensagens entre publicadores e assinantes por meio de tópicos específicos. Publicadores enviam mensagens utilizando a API Eclipse Paho, enquanto assinantes recebem as mensagens ao se inscreverem nos

³ Mosquitto.org

tópicos. Essa estrutura desacoplada possibilita uma comunicação escalável, mesmo em ambientes com restrições de largura de banda. O Mosquitto é frequentemente utilizado em sistemas IoT, sendo projetado para aplicações leves e cenários onde a eficiência de rede é importante [Solanki e Dhall 2017].

Casos de Uso

O Mosquitto é amplamente aplicado em diversos setores, incluindo [Mosquitto 2018, Solanki e Dhall 2017, Silva et al. 2017]:

- **Serviços Financeiros:** Detecção de fraudes.
- **Redes Inteligentes:** Dashboards e gráficos.
- **Edifícios Inteligentes:** Manutenção preditiva.
- **Varejo e Logística:** Monitoramento de inventário em tempo real.
- **Saúde:** Controle de infecções em tempo real.
- **Fábricas Inteligentes:** Monitoramento de condições operacionais.
- **IoT:** Conexão de dispositivos à nuvem.
- **Automação Residencial:** Controle de dispositivos como luzes e termostatos.
- **Monitoramento Industrial:** Supervisão de sensores e equipamentos.
- **Aplicações Móveis:** Comunicação em tempo real com servidores.

Aplicações em Sistemas Distribuídos

Em sistemas distribuídos, especialmente na IoT, o Mosquitto centraliza a troca de mensagens entre dispositivos, sensores e aplicações. Sua capacidade de processar grandes volumes de mensagens com baixa latência o torna ideal para aplicações em tempo real, como monitoramento ambiental, controle de automação residencial e gestão de infraestruturas críticas.

Características Principais

O Mosquitto oferece uma implementação robusta das versões 3.1, 3.1.1 e 5.0 do protocolo MQTT, sendo uma solução para sistemas de publicação/assinatura. Entre suas principais características, destacam-se:

- **Suporte a QoS:** Permite diferentes níveis de qualidade de serviço (QoS 0, 1 e 2), adequando-se a diferentes exigências de confiabilidade na entrega de mensagens.
- **Retenção de Mensagens:** Mensagens podem ser armazenadas no *broker* para novos assinantes.
- **Pontes MQTT:** Integração com outros *brokers*, permitindo expansão de redes MQTT.

A Tabela 1 apresenta detalhes dos níveis de QoS e seus usos recomendados.

Tabela 1 – Características de cada nível de QoS

Nível de QoS	Descrição	Uso Recomendado
QoS 0	Entrega no máximo uma vez (<i>fire and forget</i>).	Dados não críticos.
QoS 1	Garante entrega pelo menos uma vez.	Dados importantes que requerem confirmação.
QoS 2	Garante entrega exatamente uma vez.	Dados críticos sem tolerância a duplicações.

Fonte: O autor

Segurança e Suporte

A segurança é importante em sistemas distribuídos. O Mosquitto oferece:

- **Autenticação:** Suporte a autenticação básica (usuário e senha) e por certificados TLS.
- **Criptografia:** TLS para proteger dados em trânsito.
- **Gerenciamento:** Administração centralizada pelo Management Center.
- **Suporte:** Serviços profissionais da Cedalo, incluindo monitoramento e integração avançada.

A Tabela 2 detalha os recursos de segurança recomendados.

Tabela 2 – Recursos de segurança do Eclipse Mosquitto

Recurso	Descrição	Configuração recomendada
Autenticação Básica	Usuário e senha.	Redes controladas ou testes.
Certificados TLS	Criptografia e autenticação via TLS/SSL.	Produção em redes abertas.
Criptografia de Dados	Proteção em trânsito com TLS.	Importante para dados sensíveis.

Fonte: o autor

O Eclipse Mosquitto combina leveza, segurança e versatilidade, sendo ideal para aplicações IoT e sistemas distribuídos. Sua arquitetura flexível e recursos avançados o tornam uma escolha adequada para projetos que exigem comunicação entre dispositivos conectados [Mosquitto 2018].

2.8 PlanktoScope

O PlanktoScope (Figura 8) é uma plataforma de imagem modular projetada para oceanografia cidadã, combinando acessibilidade e flexibilidade. Ele possibilita a análise de plâncton por meio de uma estrutura modular de software e hardware. As versões atuais do PlanktoScope, produzem imagens de 1,7 mL por minuto com resolução de 1,5 μm , de forma autônoma. Seu baixo custo facilita a aplicação em novos contextos e a utilização prolongada. Embora tenha sido desenvolvido principalmente para análises planctônicas, o PlanktoScope é uma plataforma versátil, adaptável a diferentes finalidades [Pollina et al. 2020].

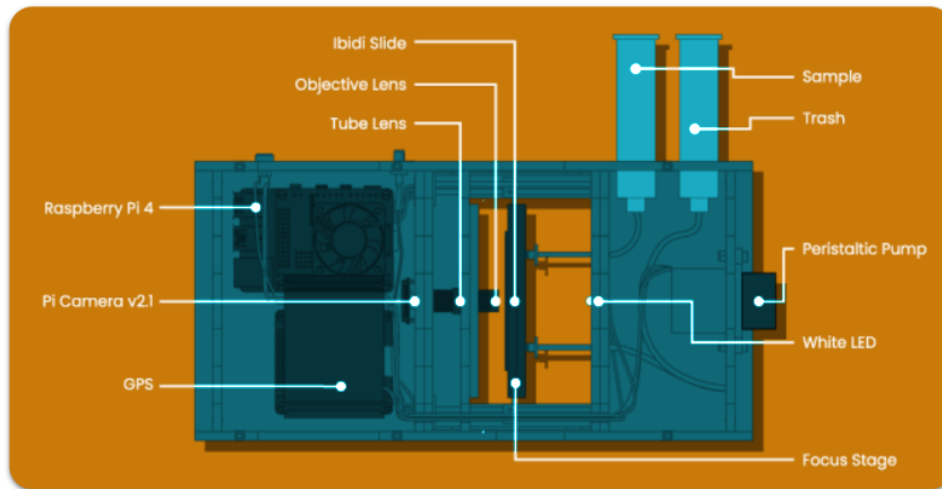
O microscópio de plâncton permite capturar imagens de micropartículas com tamanhos entre 20 μm e 200 μm [Mériguet et al. 2022]. Apesar da ampla variação no tamanho dos microplásticos, o PlanktoScope pode contribuir para a avaliação do risco ecológico. Tradicionalmente, a identificação de microplásticos, na faixa de 1 mm a 5 mm, tem sido realizada por meio de classificação visual. No entanto, é importante desenvolver métodos para monitorar a presença de microplásticos no ambiente e avaliar seus efeitos biológicos, incluindo a análise de partículas menores, como nano-plásticos [Shim, Hong e Eo 2017].

A diversidade de tamanhos, formas, cores e tipos de polímeros dificulta a criação de bases de dados consistentes sobre microplásticos. Essa limitação compromete a comparação de dados e impede a avaliação do estado de poluição, a compreensão da dinâmica dos microplásticos, a identificação de pontos de concentração, o acom-

panhamento de tendências históricas e a análise do impacto nos organismos expostos em diferentes ecossistemas e regiões [Lusher et al. 2021].

Imagens obtidas por microscopia permitem a análise detalhada da textura da superfície e da estrutura de objetos, sendo essenciais para a identificação de partículas ambíguas ou semelhantes a plástico.

Figura 8 – Visão geral dos principais componentes do PlanktoScope.



Fonte: [PlanktoScope.org. 2020].

O PlanktoScope é uma plataforma de código aberto, composta por uma rede colaborativa de designers e usuários que buscam expandir o impacto e a acessibilidade das ferramentas. Ele consiste em um microscópio digital compacto, modular e acessível, formado por diferentes módulos simples [PlanktoScope.org. 2020]. Dentre estes, destacam-se 4 principais módulos:

- **Módulo de computação:** Este módulo é composto de um Raspberry Pi 4 (4Gb de LPDDR4 SDRAM) e uma Pi câmera de 8 MP.
- **Módulo óptico:** É composto por duas lentes M12 invertidas, a lente do tubo é fixa quando a lente objetiva pode ser trocada, oferecendo uma variedade de configurações ópticas. Uma platina motorizada ($\sim 30 \mu m/pass$) permite uma calibração de foco fino da platina intercambiável na qual é possível montar μ -Slide ou uma lâmina de vidro tradicional, dependendo do modo (observação fluídica ou estática).
- **Módulo Fluido-Mecânico:** Composto por uma bomba peristáltica acionada por um motor de passo possuindo uma ordem de precisão ($\sim 0,1 \text{ ml/min}$) é usada para arrastar o fluido para a célula de fluxo, se necessário.

- **Módulo GPS:** Um GPS é utilizado para gravar a localização da coleta da amostra.

3 Trabalhos relacionados

Esta seção analisa estudos sobre o uso do MQTT em aplicações de envio de imagens, abordando desafios como confiabilidade, segurança, escalabilidade e controle de fluxo.

3.1 Desafios, problemas e limitações na implementação de redes baseadas em MQTT

A Qos Approach For Internet Of Things (Iot) Environment Using Mqtt Protocol [Sadeq et al. 2019]

[Sadeq et al. 2019], aborda o problema de controle de fluxo entre *publishers* e *subscribers* no MQTT, onde a ausência de controle de fluxo pode resultar em perda de pacotes e atrasos. O estudo propõe um mecanismo de controle de fluxo para reduzir a perda de pacotes e o atraso. A seguir, detalham-se esses desafios.

1. Ausência de Controle de Fluxo de Ponta a Ponta

O MQTT não possui controle de fluxo, o que pode causar sobrecarga nos assinantes, perdas de pacotes e atrasos na comunicação.

2. Perdas de Pacotes e Altos Atrasos

A falta de controle de fluxo aumenta o atraso e pode levar ao descarte de pacotes devido à sobrecarga nos assinantes.

3. Falta de Garantia de Controle de Fluxo no Protocolo Padrão

O MQTT, em sua configuração padrão, não inclui um mecanismo nativo de controle de fluxo. Isso agrava os problemas de confiabilidade em redes IoT e CPS, especialmente sob alta carga de tráfego.

4. Necessidade de Diferentes Níveis de QoS

As aplicações IoT demandam diferentes níveis de Qualidade de Serviço (QoS), dependendo do tipo de dados. Dados críticos, como os de monitoramento de saúde ou segurança, requerem baixa tolerância a atrasos e perdas. A dificuldade em adaptar o QoS para diferentes categorias de tráfego é um desafio para atender a esses requisitos.

As redes baseadas em MQTT carecem de controle de fluxo de ponta a ponta, o que pode causar atrasos e perda de pacotes, especialmente em aplicações sensíveis

como monitoramento de saúde e segurança. Essa limitação ocorre porque o protocolo não regula o tráfego entre publicadores e assinantes, permitindo sobrecargas.

Reliable Image Notifications for Smart Home Security with MQTT [Nazir e Kaleem 2019]

Com base em [Nazir e Kaleem 2019], a implementação de redes baseadas em MQTT para transmissão de dados em sistemas IoT e CPS apresenta desafios e limitações que devem ser considerados:

1. **Confiabilidade da Rede:** Embora o MQTT seja projetado para operar em redes com baixa largura de banda e alta latência, sua confiabilidade pode ser comprometida em ambientes com falhas frequentes de conexão.
2. **Consumo de Energia:** Apesar de adequado para dispositivos de baixa potência, o MQTT pode aumentar o consumo de energia dependendo do nível de Qualidade de Serviço (QoS) escolhido, devido ao maior número de trocas de pacotes.
3. **Segurança:** A proteção de dados sensíveis, como imagens transmitidas para sistemas de segurança, exige medidas robustas para evitar interceptações e acessos não autorizados.
4. **Escalabilidade:** O aumento do número de dispositivos conectados pode sobrecarregar o broker MQTT, caso não seja dimensionado adequadamente para lidar com grandes volumes de dados.
5. **Latência:** Apesar de oferecer uma latência média competitiva em relação a outros protocolos, como o AMQP, o MQTT pode enfrentar dificuldades para garantir tempos de resposta rápidos em redes congestionadas.
6. **Interoperabilidade:** A integração de dispositivos e plataformas heterogêneas pode ser limitada, frequentemente necessitando de adaptações para suportar o protocolo MQTT.

Esses pontos enfatizam a importância de planejamento detalhado e soluções técnicas para mitigar os desafios na implementação de redes MQTT em sistemas IoT e CPS. Redes baseadas em MQTT, carecem de segurança integrada, tornando-se vulneráveis em aplicações críticas que exigem proteção de dados sensíveis, como imagens de segurança doméstica.

An Enhanced Reliable Message Transmission System Based on MQTT Protocol in IoT Environment [Nativi, Mazzetti e Craglia 2021]

O artigo discute a vulnerabilidade do MQTT em manter a ordem das mensagens, o que é crucial em automação residencial, e propõe um sistema de transmissão de mensagens para resolver esse problema. Segundo [Nativi, Mazzetti e Craglia 2021] a implementação de redes baseadas no protocolo MQTT em sistemas IoT e CPS apresenta os seguintes desafios e limitações:

1. **Ordem das Mensagens:** O MQTT não garante a sequência das mensagens, o que é crucial em aplicações que dependem de comandos ordenados, como o controle de válvulas de gás em automação residencial.
2. **Níveis de Qualidade de Serviço (QoS):** Muitas plataformas de nuvem IoT suportam apenas os níveis de QoS 0 e 1, que não asseguram a entrega de mensagens exatamente uma vez (QoS 2). Em redes instáveis, isso pode causar duplicação ou perda de mensagens.
3. **Redes Instáveis:** Apesar de projetado para redes com baixa largura de banda e alta latência, o MQTT pode falhar na entrega de mensagens sob condições de atrasos extremos na rede.
4. **Segurança:** A segurança é uma preocupação crítica, pois falhas na transmissão de mensagens podem ocasionar erros graves, como alarmes falsos de fumaça ou falhas no monitoramento de vazamentos de gás e água.
5. **Sistemas Confiáveis:** Construir sistemas que garantam confiabilidade e ordem das mensagens exige a adição de mecanismos, como campos de sequência, para validação e entrega correta.

Esses pontos evidenciam a necessidade de melhorias no protocolo ou em suas implementações para aumentar a confiabilidade e a segurança em redes baseadas em MQTT voltadas para IoT e CPS. O documento não aborda de forma suficiente as questões de segurança no uso do protocolo MQTT em ambientes IoT/CPS, deixando de detalhar medidas específicas de mitigação de riscos e a implementação de autenticação e criptografia. Além disso, a integração do MQTT com tecnologias emergentes, como blockchain e inteligência artificial, não é explorada. O documento também carece de estudos de caso ou exemplos práticos para ilustrar os desafios e soluções propostas.

Internet of Things Survey and open issues of MQTT Protocol [Yassein et al. 2017]

De acordo com [Yassein et al. 2017], a análise comparativa entre os protocolos CoAP e MQTT revela que o MQTT apresenta maior atraso em condições de alta perda de pacotes. Essa característica é especialmente relevante para a transmissão de imagens em redes IoT, onde a confiabilidade e a eficiência na entrega de dados são essenciais. O estudo, conduzido em ambiente de laboratório com emuladores de rede celular, avaliou os tempos de transmissão e investigou possíveis otimizações nos protocolos CoAP, MQTT e OPC UA. Entre os principais desafios no uso do MQTT para transmissão de imagens em sistemas IoT, destacam-se:

1. **Atraso em Cenários de Alta Perda de Pacotes:** O MQTT apresenta maior latência em condições de alta perda de pacotes, impactando negativamente a transmissão de imagens.
2. **Consumo de Largura de Banda:** O protocolo pode demandar mais largura de banda do que o CoAP, o que é desafiador em redes com recursos limitados.

Uma limitação do estudo mencionado é a ausência de uma avaliação integrada que considere todos os protocolos em um único experimento, o que permitiria uma análise comparativa mais abrangente entre CoAP e MQTT em diferentes condições de rede. Além disso, o trabalho não explora de forma satisfatória questões de confiabilidade e segurança, aspectos essenciais para aplicações práticas em redes IoT. Por fim, a análise poderia ser aprimorada com uma avaliação mais detalhada do impacto dos diferentes perfis de Qualidade de Serviço (QoS) no desempenho do MQTT, especialmente em cenários com alta taxa de perda de pacotes.

Toward a Framework for the Composition and Assessment of IoT and CPS Capabilities : Smart Cities Applications [Halba 2023]

Segundo [Halba 2023] identificaram desafios de interoperabilidade e heterogeneidade em sistemas IoT/CPS, que dificultam a comunicação e a integração de sistemas heterogêneos, especialmente devido às diferenças entre interfaces, APIs e protocolos de comunicação [7]. Em redes baseadas em MQTT, a padronização e a compatibilidade são importantes para garantir o funcionamento. Os principais desafios na implementação de redes baseadas em MQTT para transmissão de imagens em sistemas IoT/CPS incluem:

1. **Interoperabilidade:** A comunicação entre dispositivos e plataformas distintos é limitada pela falta de interoperabilidade, dificultando a integração de serviços.

2. **Heterogeneidade dos Sistemas:** A diversidade de protocolos e interfaces de dispositivos dificulta a compatibilidade e um arranjo adequado de serviços.
3. **Escalabilidade:** Adaptar os sistemas para suportar um número crescente de dispositivos e volumes de dados é um desafio técnico significativo.
4. **Privacidade e Segurança:** A proteção de dados transmitidos, especialmente em aplicações sensíveis, é uma preocupação central.
5. **Facilidade de Desenvolvimento:** Há barreiras relacionadas ao tempo e à complexidade do desenvolvimento, personalização e prototipagem de serviços compostos.
6. **Confiabilidade:** Assegurar a transmissão de dados sem perdas é importante para a operação segura desses sistemas.

Esses desafios evidenciam a complexidade de projetar redes IoT/CPS que sejam ao mesmo tempo escaláveis e seguras. Uma das limitações do uso do protocolo MQTT para transmissão de imagens em sistemas IoT e CPS é a falta de otimização para lidar com dados volumosos. Projetado para mensagens leves e frequentes, o MQTT pode não atender adequadamente aos requisitos de largura de banda necessários para o envio de imagens, o que pode resultar em maior latência e risco de perda de dados. Além disso, a segurança e privacidade das informações são questões críticas, já que o MQTT não inclui criptografia nativa, exigindo a implementação de soluções externas para proteger dados sensíveis. Assim, é evidente a necessidade de protocolos ou frameworks complementares que aprimorem a eficiência na transmissão de dados volumosos e garantam a segurança adequada.

3.2 Síntese dos trabalhos pesquisados

A Tabela 3 sintetiza os estudos relacionados ao uso do protocolo MQTT para envio de imagens, destacando suas áreas de aplicação, problemas enfrentados, desafios superados e os principais pontos fortes. Os trabalhos analisados exploram diferentes aspectos da implementação do MQTT, oferecendo uma visão clara sobre as complexidades e benefícios da tecnologia. A tabela também aponta como este estudo pode contribuir para solucionar os desafios identificados, ampliando as aplicações do MQTT.

Tabela 3 – Resumo dos trabalhos relacionados sobre tecnologia MQTT aplicada ao envio de imagens

Autores	Área de Aplicação	Problemas	Desafios	Pontos Fortes	Colaboração do Meu Trabalho
A QoS Approach for IoT Environment Using MQTT Protocol [Sadeq et al. 2019]	Controle de fluxo entre publishers e subscribers.	Ausência de controle de fluxo; Perda de pacotes; Altos atrasos.	Não aborda QoS dinâmico para diferentes tipos de tráfego.	Mecanismo de controle de fluxo para reduzir atrasos e perdas.	A metodologia proposta para particionamento e recuperação de imagens utilizando MQTT permite otimizar o consumo de banda e de energia, especialmente relevante para redes IoT com restrições de recursos.
Reliable Image Notifications for Smart Home Security with MQTT [Nazir e Kaleem 2019]	Transmissão de imagens confiável para segurança doméstica.	Confiabilidade da rede; Consumo de energia; Segurança; Escalabilidade.	Maior consumo de energia em QoS elevado; vulnerabilidade em redes congestionadas.	Protocolo adaptativo para mitigar problemas de confiabilidade.	Implementação de arquitetura de rede baseada em MQTT para transmissão confiável de imagens de alta resolução em redes IoT.
An Enhanced Reliable Message Transmission System Based on MQTT [Nativi, Mazzetti e Craglia 2021]	Garantia de ordem nas mensagens.	Ordem de mensagens; Redes instáveis; Segurança.	Não inclui detalhes sobre autenticação e criptografia.	Campos de sequência para validação de mensagens.	Aplicação do modelo Pub/Sub em dispositivos com limitação de recursos para garantir confiabilidade e ordem na transmissão.
Internet of Things Survey and Open Issues of MQTT Protocol [Yassein et al. 2017]	Comparação entre MQTT e outros protocolos (CoAP, OPC UA).	Atrasos em alta perda de pacotes; Consumo de largura de banda.	Foco em simulações; não inclui avaliação prática em campo.	Sugestões para otimizar o QoS e o desempenho geral do protocolo.	Adaptação do protocolo MQTT para integração com sensores e computação em borda em aplicações de monitoramento ambiental.

Autores	Área de Aplicação	Problemas	Desafios	Pontos Fortes	Colaboração do Meu Trabalho
Toward a Framework for Composition and Assessment of IoT and CPS Capabilities [Halba 2023]	Desafios de interoperabilidade em sistemas heterogêneos.	Interoperabilidade; Escalabilidade; Privacidade e segurança.	Falta de otimização para dados volumosos; ausência de criptografia nativa.	Frameworks para padronização e compatibilidade.	Integração de tecnologias emergentes (IoT e Digital Twins) para promover escalabilidade em redes de monitoramento distribuídas.
Fonte: o autor					

A análise dos trabalhos relacionados revela que o protocolo MQTT possui elevado potencial para aplicações em sistemas IoT e CPS, especialmente na transmissão de dados como imagens. Contudo, sua utilização em cenários críticos enfrenta desafios relevantes, como ausência de controle de fluxo, confiabilidade reduzida em redes instáveis, falta de mecanismos nativos de segurança e dificuldades de interoperabilidade. Esses aspectos evidenciam a necessidade de aprimoramentos no protocolo ou de soluções complementares. Além disso, a escalabilidade e o consumo de recursos computacionais devem ser adequadamente planejados para atender às crescentes demandas.

4 Desenvolvimento

Este capítulo apresenta o desenvolvimento do trabalho, organizando os procedimentos de forma sistemática. Na Seção 4.1, descreve-se a arquitetura da rede, incluindo os agentes do sistema, os componentes necessários para a transmissão de dados e a interação entre módulos e tópicos de comunicação. A Seção 4.2 detalha a implementação e o ambiente experimental, abordando os dispositivos e ferramentas utilizadas, além da geração de *datasets* para validação do sistema implementado.

A Seção 4.3 foca na técnica proposta para transmissão e reconstrução de imagens por meio de tópicos de mensagens, detalhando duas abordagens distintas e os aspectos de comunicação envolvidos. Por fim, a Seção 4.4 trata da implementação em um ambiente de rede controlada na forma de um *testbed*, especificando a topologia, os dispositivos, protocolos de comunicação, bem como os parâmetros de desempenho configurados e a coleta de dados para análise.

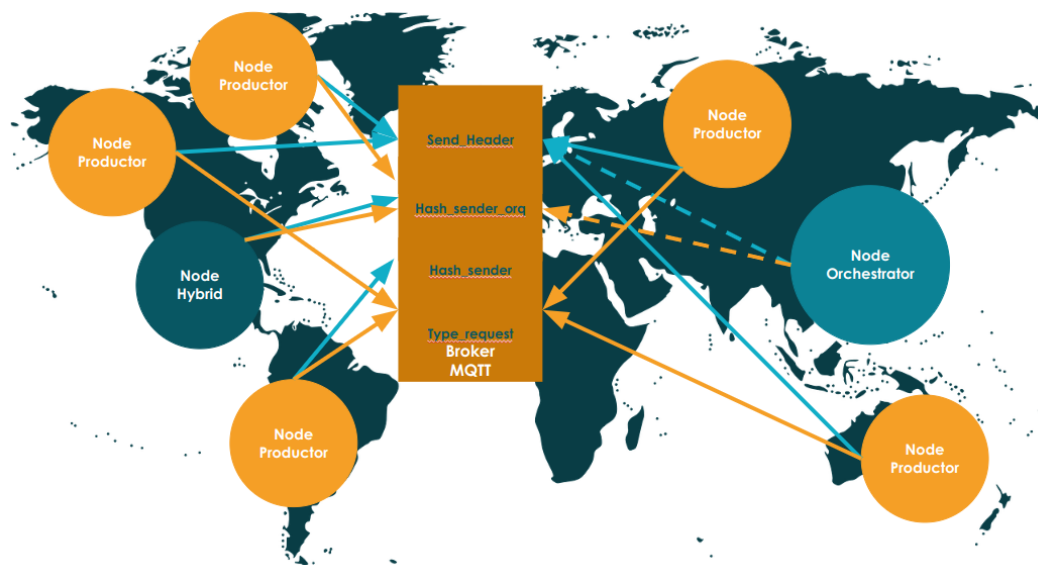
4.1 Arquitetura da rede federada

Esta seção apresenta a arquitetura em alto nível de um sistema distribuído que por meio de um Pub/Sub permite coletar, compartilhar, processar e armazenar dados.

A arquitetura consiste em nós (ou nodos) como componentes fundamentais, em que cada nó é um dispositivo computacional independente, podendo executar tarefas específicas e se comunicar com outros nós para realizar o processamento de forma distribuída.

Nesta arquitetura um dos nós utilizados é representado pelo coletor de dados, que na nossa aplicação de monitoramento ambiental é o dispositivo PlanktoScope (*NodeProducer*). Entretanto, espera-se que outros tipos de nós coletores de dados possam ser integrados à esta arquitetura. A interação entre nós pode ser observada na Figura 9.

Figura 9 – Arquitetura global de comunicação por mensagens MQTT entre diferentes tipos de nós em uma rede distribuída



Fonte: o autor

4.1.1 Agentes e componentes do sistema

Os agentes do sistema, implementados em Python, realizam ações que exigem interação com outros agentes da rede. A seguir, são apresentados os agentes e suas respectivas funções:

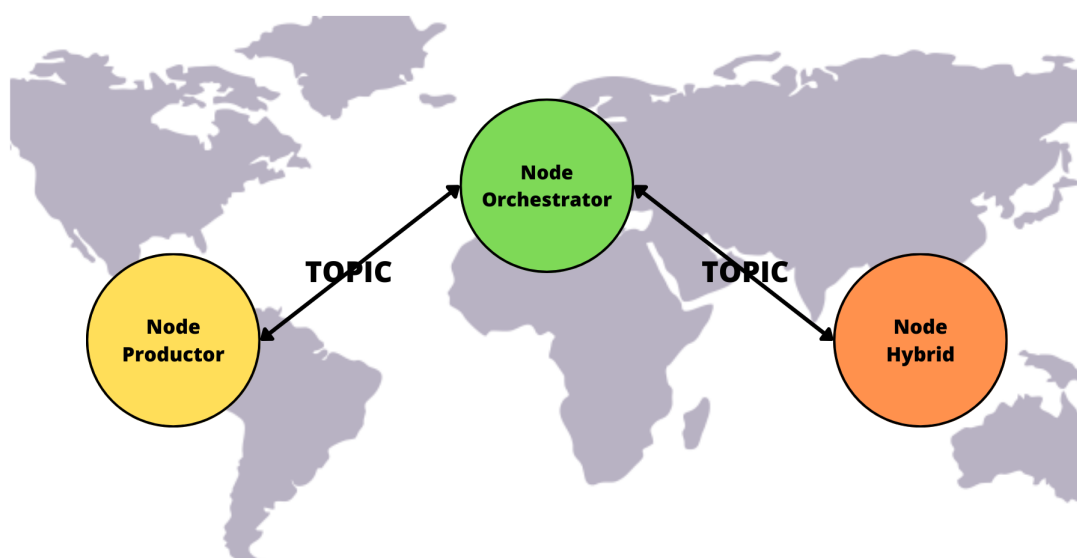
- **Asset:** é um agente opcional do sistema que pode interagir com nós capazes de enviar informações para rede. Mais especificamente, no trabalho em questão a função do *Asset* é disponibilizar ou solicitar dados (imagens e/ou informações das mesmas, metadados) para um nó com acesso à rede. Ele é responsável pelo início do processo, atuando como ponto inicial na cadeia de distribuição de dados.
- **NodeProductor** é um nó com capacidade de enviar dados para a rede, sendo estes provenientes de *Assets* ou capturados por ele próprio. Este tipo de nó é responsável por gerar ou coletar dados e enviar esses dados para outros pontos na rede que irão processá-los ou armazená-los para uso posterior. Este nó também é capaz de preparar as imagens e suas respectivas informações para serem transmitidas. Por exemplo, este tipo de nó pode estar presente em sensores para coleta de dados ambientais, dispositivos de monitoramento para informações de *status* e arquivos para dados históricos.
- **NodeOrchestrator** é responsável pela coordenação de atividades no sistema, gerenciando a comunicação entre diferentes nós, garantindo que os dados

trafeguem de forma correta. Este nó atua como o núcleo central do sistema, recebendo dados de vários *NodeProductor* e garantindo que esses dados sejam armazenados de maneira organizada.

- ***NodeHybrid***: O *NodeHybrid* age como um nó intermediário no sistema, permitindo que assinantes atrasados (ou seja, aqueles que não estavam conectados no momento da publicação) ainda possam receber as imagens publicadas. Sua função é solicitar imagens que foram enviadas para o *NodeOrchestrator* por um, ou mais de um, *NodeProductor*. Em termos de arquitetura, o *NodeHybrid* atua como um ponto de recuperação para dados publicados anteriormente (i.e. uma forma de "bufferização" na rede), garantindo que a perda de conexão não impeça os assinantes de acessar as imagens relevantes publicadas pelo *NodeProductor*.

Uma vez detalhados os agentes do sistema, a Figura 10 ilustra a interação e o fluxo de dados entre esses nós dentro da rede. Trata-se de um DT baseado em um sistema federado de nós distribuídos no qual três tipos de nós — *Productor*, *Orchestrator* e *Hybrid* — estão interconectados por meio de tópicos MQTT. Este sistema se destaca pela sua capacidade de integrar diferentes funções e operações em uma rede distribuída.

Figura 10 – Rede federada de nós de processamento distribuído com orquestração MQTT



Fonte: o autor

4.1.2 Componentes para transmissão

Para que os nós operem dentro da rede, estes precisam de funcionalidades específicas, modularizadas e descritas abaixo.

ImageSplitter

Sua função é fragmentar as imagens em partes menores para permitir o seu envio. A operação deste módulo resulta em no mínimo dois subcomponentes: (1) um cabeçalho gerado que contém informações do arquivo (metadado) e (2) a imagem propriamente dita, caso sua dimensão seja inferior ou igual ao valor estipulado para partes ou fragmentos de imagens que podem transitar na rede. Imagens maiores, poderão ser divididas em mais partes/fragmentos, porém mantendo o cabeçalho como uma das partes no envio.

ImageSender

Sua função é receber as partes anteriormente fragmentadas pelo *ImageSplitter*. Após o recebimento, envia as partes pela rede por meio de tópico específico dentro do Pub/Sub.

ImageReceiver

A função do *ImageReceiver* é receber, por meio de um tópico específico, as partes transitantes pela rede, encaminhar os dados recebidos para o módulo *ImageRestore* e, caso necessário, solicitar o reenvio de partes faltantes.

ImageRestore

Sua função é receber os fragmentos recebidos pelo módulo *ImageReceiver* e restaurá-los em imagens originais. Este módulo realiza uma verificação inicial para assegurar que todas as partes da imagem foram recebidas sem algum tipo de dado corrompido. Após a restauração bem-sucedida, as imagens são armazenadas. No caso de partes faltantes, este módulo solicita ao *ImageReceiver* o reenvio da mesma.

FileStatusManager

A função do *FileStatusManager* é monitorar e atualizar o status dos arquivos recebidos pelo *ImageReceiver*. Ele não é empregado apenas no recebimento de imagens, mas também no envio, garantindo a integridade e consistência dos dados ao longo do processo de compartilhamento entre nós.

O estados (*status*) possíveis para cada imagem são:

- No recebimento: - *working*: indica que as partes de uma imagem estão chegando ou ainda não foram restauradas; - *completed*: imagem restaurada com sucesso.
- No envio: - *pending*: partes estão prontas para envio, mas somente o cabeçalho foi enviado e aguarda confirmação de entrega; - *working*: resposta de que o cabeçalho foi recebido com sucesso e as demais partes estão sendo enviadas; - *waiting*: todas as partes foram enviadas e o sistema aguarda confirmação de que nenhuma parte necessite reenvio; - *completed*: a imagem restaurada de forma bem sucedida em outro nó da rede; - *recused*: se um cabeçalho contendo um *hash* já completado é recebido (por meio do *ImageReceiver*), este é recusado e essa informação é atribuída como status pelo *ImageSender*.

Storage

A principal função do *Storage* é armazenar as imagens processadas pelo *ImageRestore* no *NodeOrchestrator*. Desta forma, o *Storage* atua como um repositório ou um banco de dados, utilizando tecnologias de armazenamento em nuvem, dependendo da arquitetura geral do sistema. As ficam armazenadas no *NodeOrchestrator*.

4.1.3 Interação entre os módulos

A interação entre os elementos nós, tópicos e agentes (Figura 9) é importante para o funcionamento do sistema proposto. As principais interações consistem na coleta de dados, na recepção e armazenamento destes dados, conforme descrito a seguir:

- **Coleta de dados:** O *NodeProductor* coleta dados de fontes diversas e os envia para o *NodeOrchestrator* via um tópico MQTT. Este processo assegura que dados de diferentes origens sejam centralizados para processamento e/ou armazenamento.
- **Recepção e armazenamento:** O *NodeOrchestrator* recebe dados do *NodeProductor*, valida e armazena as imagens recebidas.
- **Envio sob demanda:** O *NodeHybrid* atua como um ponto de recuperação no sistema, permitindo que assinantes atrasados acessem imagens publicadas anteriormente. Ele faz a requisição de imagens ao *NodeOrchestrator* para garantir que todos os assinantes tenham acesso completo aos dados, mesmo após desconexões, reforçando a resiliência do sistema.

4.1.4 Resumo dos agentes e componentes do sistema

A Tabela 4 resume os principais agentes no DT federado e sua importância no fluxo de trabalho proposto. Trata-se de um resumo do processo desde a geração inicial de ativos pelo componente *Asset* até a gestão final dos dados no componente *Storage* e *FileStatusManager*.

Tabela 4 – Tabela simplificada do processamento e gerenciamento de imagens por meio do MQTT

Componente	Descrição	Observações
<i>Asset</i>	Ponto de partida onde um ativo é gerado ou recebido.	Geralmente uma imagem ou conjunto de imagens.
<i>ImageSplitter</i>	Divide o ativo em partes menores (fragmentos).	Facilita o envio ou processamento.
<i>ImageSender</i>	Envia as partes da imagem para um tópico MQTT.	Inscrito no tópico de recebimento de imagens.
<i>ImageReceiver</i>	Recebe as imagens do tópico MQTT. Solicita partes faltantes.	Encaminha para restauração das imagens.
<i>ImageRestore</i>	Restaura as partes da imagem e identifica partes faltantes.	Resulta na imagem original ou completa.
<i>Storage</i>	Armazena a imagem restaurada.	Em um dispositivo local ou na nuvem.
<i>FileStatusManager</i>	Gerencia o <i>status</i> do arquivo.	Atualiza o estado do processamento.

Fonte: o autor

4.1.5 Cabeçalho de imagens

No cabeçalho (ou *header*) gerado para cada imagem, encontram-se as informações relativas à mesma. As informações presentes em cada cabeçalho são:

- Resolução: contém dimensões da imagem;
- Localização: carrega a informação da localidade da amostra apresentada na imagem;
- *File_hash*: transporta um identificador único para a imagem;
- *Sender_hash*: carrega um identificador único para o dispositivo que envia a imagem;

- *Timestamp*: transporta a informação temporal de geração da imagem;
- *File_name*: carrega o nome original da imagem;
- *Category*: contém a classificação do conteúdo da imagem;
- Quantidade de partes: transporta a quantidade de frações nas quais a imagem foi dividida.

Para geração do identificador único do arquivo, chamado de *file_hash*, utiliza-se do algoritmo *SHA256* que gera 32 *bytes*. Como semente para o algoritmo, são usados o nome original da imagem concatenado de seu tamanho em *bytes*, conforme Equação 4.1.

$$file_hash = SHA256(file_name + _ + file_size) \quad (4.1)$$

Para o identificador único do dispositivo de envio, que é chamado de *sender_hash*, utiliza-se do algoritmo *MD5*, que gera 16 *bytes*. A semente é o próprio *Mac Address* do dispositivo, como visto em 4.2.

$$file_hash = MD5(mac_address) \quad (4.2)$$

Um exemplo de cabeçalho pode ser visto abaixo.

- **Resolução:** 300 x 300
- **Localização:** -32.0722348, -52.1665717, 17.37
- **File Hash:** f13d05aa7852ca29d2079e02cb40fc26d371d3202b01e0e8f41ed3779cf50435
- **Sender Hash:** 8bb5d3d4df9b9210e799b38280756f10
- **Timestamp:** 1729883559
- **Nome do Arquivo:** microplastico_laguna_dos_patos_057.jpeg
- **Categoria:** MP_20_µm
- **Quantidade de Partes:** 4

4.1.6 Tópicos de comunicação

Existem diferentes tópicos dentro do sistema Pub/Sub, cada tópico associado à uma etapa da comunicação dos dados.

- *Send_header*: Tópico para enviar informações iniciais de metadados da imagem, incluindo tamanho, formato e identificador do nó emissor (cabeçalho);
- *Hash_sender*: Tópico para confirmação de recebimento do cabeçalho e solicitação de reenvio de partes perdidas ou corrompidas;
- *Hash_sender_orq*: Tópico usado para enviar as partes fragmentadas da imagem ao receptor após a confirmação do cabeçalho;
- *Type_request*: Tópico usado para solicitar envio de imagens pelo *NodeOrchestrator*, transmitindo a categoria de imagem desejada.

1. Interação entre tópicos e módulos no fluxo de envio de imagens:

- O módulo *ImageSender* envia um cabeçalho no tópico *send_header*. Este cabeçalho contém informações na forma de metadados sobre a imagem a ser transmitida, incluindo um identificador único do nó que está enviando (*hash_sender*).
- O módulo *ImageReceiver* recebe o cabeçalho, extrai o *hash_sender* e envia uma notificação de recebimento ao remetente por meio de um tópico específico identificado pelo próprio *hash_sender*.
- Após receber a confirmação, o *ImageSender* prossegue com o envio das partes fragmentadas da imagem no tópico *hash_sender_orq* (que é o caminho de ida das informações).
- Se alguma parte da imagem estiver faltando ou estiver corrompida, o *ImageReceiver* pode solicitar o reenvio dessas partes específicas por meio do tópico *hash_sender* (caminho de retorno).

2. Solicitação de Imagens pelo *NodeHybrid*:

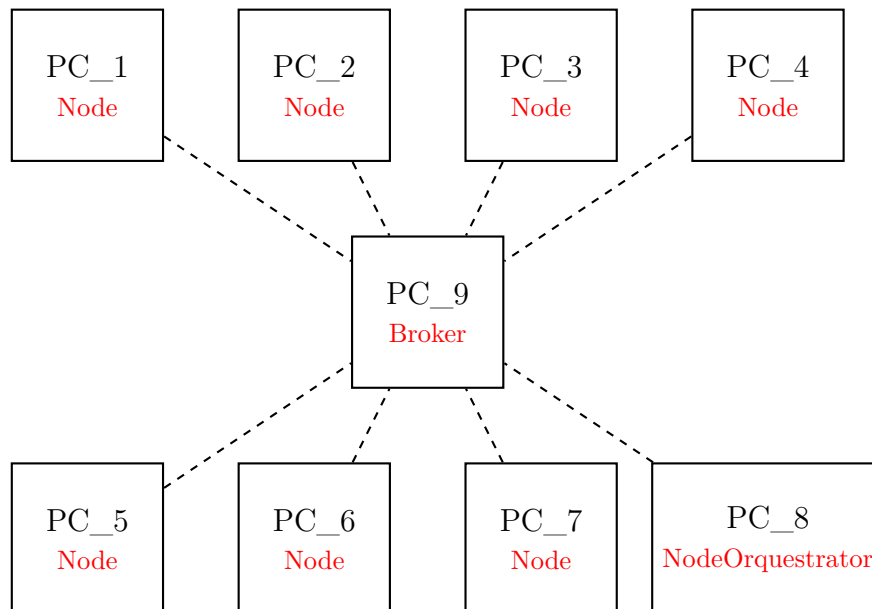
- O tópico *type_request* é utilizado pelo *NodeHybrid* para solicitar o envio de imagens. O *NodeHybrid* emite uma requisição indicando o tipo de imagens desejadas.
- O nó orquestrador responde iniciando o processo de envio por meio dos tópicos descritos anteriormente.

Este esquema de comunicação permite uma transmissão estruturada de imagens entre os diferentes nós da rede, ajudando no gerenciamento dos dados transmitidos.

4.2 Implementação e ambiente experimental Testbed

A implementação do sistema e sua validação requerem o uso de diversos dispositivos e ferramentas de software, incluindo: um microscópio para citometria de fluxo (PlanktoScope) para coleta de imagens, computadores para os nós da rede, e um servidor com o *broker* MQTT Mosquitto¹ instalado. Também são necessárias configuração do protocolo de comunicação MQTT e a criação do dataset de validação. O diagrama da Figura 11, mostra a configuração usada para experimentação, sendo o ambiente composto por sete nós (sistemas computacionais) que podem assumir a configuração de *NodeProductor* ou *NodeHybrid*. O ambiente também conta com um computador configurado para ser o *NodeOrquestrator* e outro computador configurado como o nó *Broker*.

Figura 11 – Diagrama do sistema distribuído (Testbed)



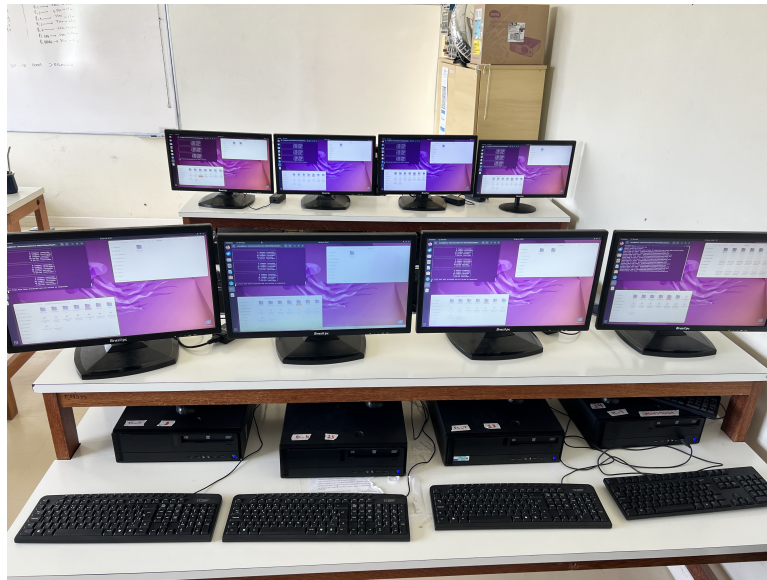
Fonte: o autor

4.2.1 Ambiente usado para experimentação

Na figura 12, mostra o laboratório de hardware do Centro de Ciências Computacionais da FURG com os computadores configurados em rede para o ambiente de experimentação.

¹ <https://mosquitto.org>

Figura 12 – Setup para experimentação montado no laboratório de hardware do Centro de Ciências Computacionais (C3)

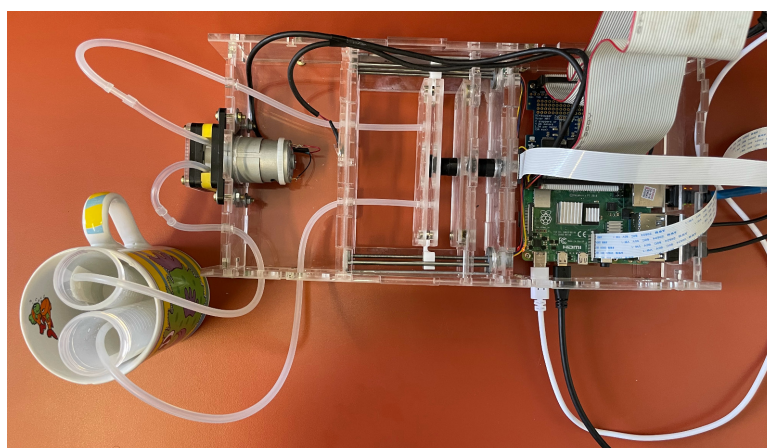


Fonte: o autor

4.2.2 Geração de dataset

A Figura 13 mostra a unidade PlanktoScope, desenvolvida na FURG como parte do projeto Europeu ASTRAL(*All Atlantic Ocean Sustainable, Profitable and Resilient Aquaculture*)².

Figura 13 – Microscópio digital PlanktoScope, configuração para coleta de amostras de microplásticos.



Fonte: o autor

² ASTRAL - O projeto ASTRAL, financiado pela UE, dentro do arcabouço da Economia Azul, desenvolve cadeias de produção em aquicultura multi-trófica integrada (IMTA). O consórcio tem parceiros na Irlanda, Escócia, África do Sul, Brasil e Argentina, visando aumentar a circularidade em até 60% e impulsionar a diversificação para aquicultores. <<https://www.astral-project.eu/the-project>>

O projeto PlanktoScope original permite o monitoramento de micropartículas, essencialmente espécies fitoplanctônicas, de até 200 μm . Entretanto, tendo em vista um objetivo de monitorar microplásticos com tamanho superior a 20 μm , alguns ajustes no *setup* óptico foram propostos. Dessa forma, as subseções a seguir versam sobre estas modificações, teste do módulo óptico e o ambiente computacional disponibilizado para este tipo de sistema.

4.2.3 Módulo óptico

O módulo óptico do PlanktoScope é uma plataforma ajustável, apresentando uma estrutura que permite a troca de câmeras e a utilização de alguns tipos de lentes para ajustes conforme a necessidade. Essas alterações permitem otimizar a resolução e o contraste das imagens, garantindo a nitidez e precisão necessárias para análises mais detalhadas, como a detecção de microplásticos. Fez-se a experimentação de uma configuração com duas câmeras e dois diferentes tipos de lentes no módulo óptico, visando otimizar a resolução e o contraste nas imagens de MPs. As câmeras selecionadas e a calibração do foco, por meio da troca de lentes, asseguraram a nitidez e a precisão para a detecção das partículas de interesse.

- **Descrição das Câmeras e Lentes** A Figura 14 mostra os componentes ópticos do sistema. O PlanktoScope permite por meio do seu módulo óptico a substituição das câmeras e lentes possibilitando testes com intuito de melhorar a captura de imagens e adequação a outras aplicações.

- (a) **Raspberry Camera V2 com sensor Sony IMX219:** Câmera equipada com sensor de imagem Sony IMX219 (resolução de 8 megapixel), permitindo a captura de imagens de alta qualidade e resolução máxima de 3280 x 2464 *pixels*. O módulo suporta gravação de vídeo em 1080p a 30 quadros por segundo, 720p a 60 quadros por segundo e 640x480p a 90 quadros por segundo. A lente no formato M12/S do módulo, possui um campo de visão fixo, mas oferece foco ajustável. Este componente é conectado ao sistema computacional embarcado Raspberry Pi por meio de uma interface CSI (Figura 14a);
- (b) **Raspberry HQ Camera, com sensor Sony IMX477R:** Camera com resolução de 4056 x 3040 e montagem de lentes no formato C/CS. Seu sensor de 12.3 megapixel permite capturar imagens de alta resolução com detalhes notáveis. O sensor tipo 1/2.3" melhor do que os encontrados nas câmeras padrão do Raspberry Pi, proporcionando um melhor desempenho em condições de baixa luminosidade e uma maior capacidade de controle sobre a profundidade de campo. Esta câmera tem suporte a lentes inter-

cambiáveis com montagem C e CS. A interface de comunicação com o Raspberry Pi é implementada por meio de um conector CSI-2, garantindo transmissão rápida de dados (Figura 14b);

- (c) **Lente M12 de 16mm:** Geralmente esta lente vem com variações de abertura de $f/1.8$ até $f/2.5$, influenciando diretamente na quantidade de luz que a lente permite entrar, alterando o parâmetro exposição das imagens. Estas lentes são compatíveis com diversos tamanhos de sensores de imagem, incluindo sensores até $2/3"$. Isso assegura que a lente possa cobrir adequadamente o sensor da câmera. O campo de visão horizontal dessas lentes varia em torno de 20° a 22° , o que determina a amplitude da cena que a câmera pode capturar, sendo um fator crucial para aplicações que exigem uma visão abrangente (Figura 14c);
- (d) **Lente M12 de 25mm:** Esta lente apresenta uma distância focal de 25mm com aberturas como $f/2.5$ ou $f/1.7$, proporcionando versatilidade em diferentes condições de iluminação. Compatível com sensores de até 5 megapixel, esta lente pode ser utilizada com uma variedade de tamanhos de sensores, desde $1/2"$ até $1/1.8"$, adequando-se a diversas necessidades de imagem. O campo de visão varia de acordo com o modelo, oferecendo entre 14.5° a 20.2° horizontalmente.(Figura 14d).

Figura 14 – Dispositivos utilizados para os testes do módulo óptico



(a) Raspberry Camera V2



(b) Raspberry HQ Camera



(c) Lente M12 16mm



(d) Lente M12 25mm

Fonte: o autor

Os resultados obtidos a partir dos testes feitos para o módulo óptico do PlanktoScope são apresentados na Seção 5.1. Nessa seção, são detalhadas as análises

e as comparações entre as diferentes configurações testadas. As modificações realizadas e os resultados são discutidos com base nos dados coletados.

Cabe salientar que não foram necessárias alterações nos demais módulos que compõem o PlanktoScope de forma que estes mantiveram suas configurações default. A configuração óptica personalizada do PlanktoScope foi importante para atender às demandas de geração de imagens de partículas de microplásticos para este estudo.

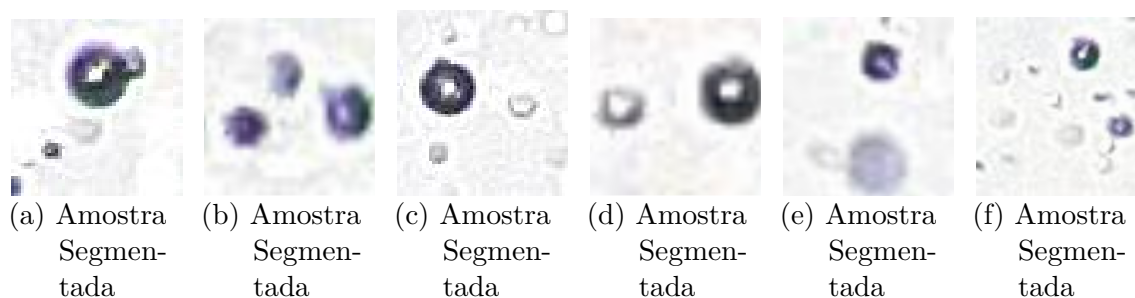
4.2.4 Produção das amostras utilizadas no Dataset

Para a geração do *dataset*, fez-se necessário a produção de amostras de microplásticos em laboratório. Essas foram produzidas em ambiente controlado por pesquisadores do Instituto de Ciências Biológicas (ICB) da FURG no laboratório de Nanotecnologia - LabNano. A confecção de microesferas poliméricas é realizada por meio da técnica de evaporação do solvente à quente, iniciando com solubilização das duas fases - orgânica (clorofórmio + poliestireno) e aquosa (álcool polivinílico + Água Mili-Q), vertendo a fase orgânica na fase aquosa por meio de uma pipeta Pasteur sob agitação magnética. Após verter uma fase na outra à solução polimérica obtida é adicionada água Mili-Q. As amostras foram utilizadas para coleta das imagens por meio do microscópio modular PlanktoScope.

Para os testes de envio de imagens usando o protocolo MQTT, criou-se dois conjuntos de dados específicos. O primeiro conjunto consiste em um pacote de 500 MBytes, contendo 445 imagens de diversos tamanhos. Este pacote é projetado para testar a capacidade do sistema em lidar com volume de dados compatível com cenário real de monitoramento de microplásticos.

O segundo conjunto de dados é um pacote menor, com tamanho médio de 100 MBytes. Este conjunto de dados é dividido em três grupos de imagens, permitindo testar diferentes cenários de carga e transmissão de dados. Este pacote inclui 20 imagens de 5 MB cada, 50 imagens de 2 MB cada e 100 imagens de 1 MB cada. Cada um desses subgrupos pode ser usado individualmente para testar a transmissão, visto que juntos eles excederiam o limite de 100 MB. Esses pacotes servem para avaliar a robustez do sistema distribuído utilizando MQTT, assegurando que este possa manejar tanto cargas pesadas quanto mais leves. A Figura 15 mostra exemplos das amostras segmentadas incluídas nos conjuntos de dados. Da mesma forma, a Tabela 5 resume os tamanhos dos pacotes e imagens utilizadas na validação do sistema DT proposto.

Figura 15 – Conjunto de dados: amostras de microplásticos segmentadas



Fonte: o autor

Tabela 5 – Detalhes dos pacotes de imagens para testes de envio por MQTT

Pacote	Imagens	Detalhes do Conteúdo
500MB	445 imagens	445 imagens de tamanhos variados
100MB	170 imagens	20 imagens de 5MB cada, 50 imagens de 2MB cada e 100 imagens de 1MB cada

Fonte: o autor

4.2.5 Ambiente computacional

O ambiente computacional dispõe de componentes de hardware e software que, integrados, permitem que o PlanktoScope ofereça flexibilidade para o monitoramento de MPs e análise ações de mitigação de impacto ambiental. O ambiente inclui:

- **Raspberry Pi SO:** o sistema operacional Raspberry Pi OS, é uma distribuição baseada em Debian desenvolvida especificamente para o hardware Raspberry Pi. Esse SO é utilizado em projetos de hardware DIY (faça você mesmo) devido à sua flexibilidade e suporte a bibliotecas e ferramentas de desenvolvimento;
- **Interface Gráfica do Usuário (GUI):** é baseada no Node-RED, uma ferramenta de desenvolvimento baseada em fluxo para programação visual. O Node-RED permite que os usuários criem fluxos de trabalho interativos conectando blocos de funcionalidade por meio de uma interface de "arrastar e soltar". Essa abordagem facilita a personalização e a automação de tarefas complexas sem a necessidade de programação extensa;
- **IDEs:** utiliza principalmente Python para o desenvolvimento das funcionalidades, por ser uma linguagem de programação de alto nível, o que a torna ideal

para projetos de prototipagem rápida e desenvolvimento de software científico. Além do Node-RED para a GUI, outras bibliotecas Python como MorphoCut são usadas para processamento de imagens;

- **Comunicação e Transmissão dos dados:** propõe-se o uso do protocolo MQTT como meio de transmissão de dados fragmentados (subseção 4.3).

4.2.6 Servidor

O servidor utilizado é definido a seguir em termos de hardware utilizado, conexão de rede, e testes de largura de banda para acesso à Internet.

4.2.7 Hardware utilizado

O protocolo MQTT, utilizado em aplicações de IoT, depende do desempenho do hardware abaixo para garantir uma comunicação adequada entre dispositivos. Ao validar uma nova solução ou arquitetura de sistema no âmbito deste trabalho, é importante que os detalhes do hardware estejam bem documentados. A Figura 16 mostra o servidor utilizado bem como lista as especificações técnicas da máquina.

Figura 16 – Servidor utilizado no trabalho com especificação: (a) Processador Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz, com 4 núcleos e 4 *threads*, (b) Memória RAM (média) de 5.7 Gb disponível e 3.1 Gb livre, e (c) Armazenamento Swap de 2 Gb, não utilizados.



Fonte: o autor

Para levantar configurações do sistema utilizado para experimentação, incluindo especificações e disponibilidade média de recursos de memória, utilizou-se a ferramenta *free* do Linux. Abaixo encontram-se as informações recolhidas por meio do terminal.

```

Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Address sizes: 39 bits physical, 48 bits virtual
Byte Order: Little Endian
CPU(s): 4
On-line CPU(s) list: 0-3
Vendor ID: GenuineIntel
Model name: Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz
CPU family: 6
Model: 94
Thread(s) per core: 1
Core(s) per socket: 4
Socket(s): 1
Stepping: 3
CPU max MHz: 3300.0000
CPU min MHz: 800.0000
BogoMIPS: 5399.81
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe
      ↪ syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
      ↪ pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt
      ↪ tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti ssbd ibrs ibpb
      ↪ stibp tpr_shadow flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap
      ↪ clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp vnm
      ↪ md_clear flush_l1d arch_capabilities
Virtualization features:
Virtualization: VT-x
Caches (sum of all):
L1d: 128 KiB (4 instances)
L1i: 128 KiB (4 instances)
L2: 1 MiB (4 instances)
L3: 6 MiB (1 instance)
NUMA:
NUMA node(s): 1
NUMA node0 CPU(s): 0-3
Vulnerabilities:
Gather data sampling: Vulnerable: No microcode
Itlb multihit: KVM: Mitigation: VMX disabled
L1tf: Mitigation; PTE Inversion; VMX conditional cache flushes, SMT disabled
Mds: Mitigation; Clear CPU buffers; SMT disabled
Meltdown: Mitigation; PTI
Mmio stale data: Mitigation; Clear CPU buffers; SMT disabled
Retbleed: Mitigation; IBRS
Spec rstack overflow: Not affected
Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Spectre v2: Mitigation; IBRS; IBPB conditional; STIBP disabled; RSB filling; PBRSE-eIBRS Not affected; BHI Not affected
Srbds: Mitigation; Microcode
Tsx async abort: Not affected

```

Listing 4.1 – Hardware/Terminal Output

```

ever@ever-To-be-filled-by-0-E-M:~$ free -h
              total used free shared buff/cache available
Mem: 5,7Gi 1,1Gi 3,1Gi 11Mi 1,5Gi 4,3Gi
Swap: 2,0Gi 0B 2,0Gi

```

Listing 4.2 – Memória/Terminal Output

4.2.8 Conexão de rede

Abaixo são resumidas os aspecto de rede de comunicação para o servidor proposto:

- **Interface de Rede:** enp1s0 que suporta até 1000baseT/Full duplex (Gigabit Ethernet)
- **Cabo de Rede:** CAT6 UTP que suporta redes de alta velocidade com até 1 Gbps para dar suporte ao *broker* MQTT de baixa latência.

Resultados do teste de velocidade de Internet

A seguir, mostra-se os resultados de um teste de banda feito em ambiente interno, para verificar se a infraestrutura pode suportar comunicação entre o *broker* MQTT e seus clientes externos.

- **Velocidade de download:** 388.39 Mbps
- **Velocidade de upload:** 162.51 Mbps
- **Latência:** 1.07 ms

Esses valores sugerem uma rede rápida, o que é importante para aplicações que dependem de comunicação em tempo real, como é o caso do sistema Pub/Sub com MQTT. A baixa latência é também relevante para manter a integridade e a rapidez na troca de mensagens entre dispositivos distribuídos. A combinação de alta taxa de dados para *upload* e *download* e com baixa latência são ideais para o MQTT, originalmente projetado na IBM para ambientes com limitações de banda e que requer baixa latência para eficiência na comunicação entre dispositivos distribuídos. Estes parâmetros de taxas são adequados para a experimentação deste trabalho.

4.2.9 Broker Mosquitto

O Eclipse Mosquitto é um broker MQTT muito utilizado para conectar sensores, dispositivos e aplicativos em uma arquitetura de Pub/Sub, oferecendo flexibilidade em comparação ao padrão de comunicação de solicitação/resposta (request/reply) da arquitetura tradicional cliente/servidor [Mosquitto 2018]. A versão do broker MQTT é Mosquitto 2.0.11.

4.2.10 Protocolo MQTT

Propõe-se o uso do protocolo MQTT, facilitando a integração e comunicação entre os dispositivos por meio de um nó intermediário, denominado de *broker*. Esta escolha se deve principalmente à sua capacidade de minimizar a sobrecarga de protocolo, o que é importante em aplicações onde a economia no uso de energia e a otimização de recursos do sistema são importantes. Por padrão para o nosso estudo, utiliza-se o QoS 0, onde a mensagem é entregue no máximo uma vez, ou seja, a entrega não é garantida.

4.3 Método de transmissão e reconstrução de imagens

A configuração proposta baseia-se em uma comunicação TCP ponto-a-ponto. A configuração envolve um *broker* MQTT central conectado inicialmente por meio de nós *NodeProductor* e *NodeOrchestrator*. Cabe lembrar que o *NodeProductor* é responsável por enviar as mensagens e *NodeOrchestrator* recebe e armazena tais mensagens. Junta-se a isso o uso do *Dataset* com as imagens para os testes. Por fim, foi estabelecido um tópico específico no *Broker*. Para avaliar o desempenho do sistema desenvolvido, foram realizados testes, no qual a conectividade à Internet foi intencionalmente manipulada por meio do desligamento da conexão.

Dois métodos são propostos para compartilhamento e transmissão de dados no âmbito do sistema DT proposto conforme apresentado nas subseções a seguir.

4.3.1 Primeiro método (V1)

O primeiro método proposto (método V1) tem como característica sua simplicidade consistindo em apenas três fases ou momentos de comunicação. A arquitetura V1 proposta para realizar as tarefas de fragmentação, envio, recebimento, restauração e armazenamento de imagens envolve a utilização dos seguintes agentes do sistema: (1) Módulo *ImageSender*, (2) Módulo *ImageSplitter*, (3) Módulo *ImageReceiver*, (4) Módulo *ImageRestore*, (5) Módulo *Storage* e (6) Tópico *SendTopic*;

- **Momento 1: Envio inicial da imagem:** O processo inicia quando um *Asset* aciona o *ImageSender* por meio de um evento de uma nova imagem. Esta imagem é dividida pelo *ImageSplitter* em partes menores ou fragmentos para facilitar a transmissão. Um *hash_file* é gerado para servir como identificador único, a partir dele é nomeado um cabeçalho com informações sobre as partes, o próprio *hash_file*, e observações relativas à imagem, se existentes. O cabeçalho então é enviado ao tópico *SendTopic* no *broker*.

- **Momento 2: Inscritos recebem as partes de imagens e seus cabeçalhos:** O processo segue com o *ImageReceiver* recebendo imagens, por meio do *SenderTopic*. Simultaneamente, o módulo *ImageRestore* começa a restauração das imagens e encaminha para o módulo *Storage*.
- **Momento 3: Imagens são restauradas e armazenadas:** O processo segue até que todas as imagens sejam restauradas por meio do módulo *ImageRestore*. No *Storage* as imagens restauradas são salvas e o processo finaliza.

O método V1 emprega um único tópico, denominado *SendTopic*, que recebe as mensagens dos *publishers* via protocolo MQTT. Essas mensagens consistem em fragmentos de imagens e um cabeçalho com dados sobre o arquivo enviado. Os *subscribers*, inscritos no tópico *SendTopic*, começam então a receber esses fragmentos, que são reagrupados para restaurar as imagens ao seu formato e tamanho originais.

4.3.2 Segundo método (V2)

O segundo método proposto (método V2) para realizar as tarefas de fragmentação, envio, recebimento, restauração e armazenamento de imagens tem como objetivo garantir um mecanismo robusto para evitar a perda de dados e assegurar uma troca de dados satisfatória ao longo do DT. O método V2 consiste em 8 fases ou momentos envolvendo a utilização dos agentes do sistema: Módulos *ImageSender*, *ImageSplitter*, *ImageReceiver*, *ImageRestore*, *Storage* e *FileStatusManager* bem como os tópicos *SendTopic* e *ReturnTopic*.

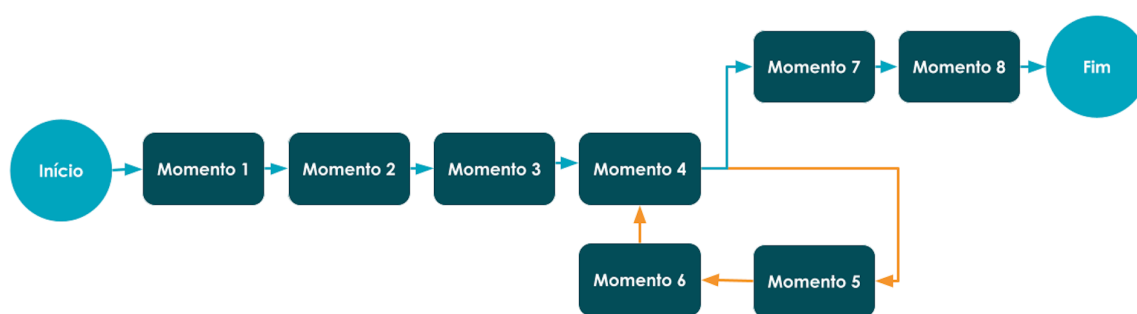
- **Momento 1: Envio inicial da imagem:** O processo inicia com o módulo *ImageSender* recebendo por meio de um *Asset* a imagem a ser transmitida. Esta imagem é então particionada utilizando o módulo *ImageSplitter*, que divide a imagem em várias partes menores para facilitar a transmissão. Simultaneamente, é gerado um *hash_file* para a imagem recebida, o qual servirá como identificador único durante todo o processo. Um cabeçalho é criado (*header*) contendo informações sobre a quantidade de partes, o *hash_file* gerado, o *hash_sender* (identificador único do nó) e quaisquer observações pertinentes à imagem. Este cabeçalho é enviado pelo tópico *SendHeader* para o *broker* de mensagens. Para gerenciar o *status* da imagem, o módulo *FileStatusManager* é utilizado. Neste ponto, o *status* "pending" é atribuído ao *hash_file* gerado, indicando que o *ImageSender* está aguardando confirmação do início da recepção. Caso ocorra um *timeout* sem a confirmação, o *header* é reenviado automaticamente pelo *broker*, garantindo a robustez do processo de transmissão.

- **Momento 2: recepção do cabeçalho:** Ao receber o *header* pelo *SendHeader*, o módulo *ImageReceiver* abre e extrai o *hash_file* e o *hash_sender* contidos nele. Utilizando o *FileStatusManager*, o *status* do *hash_file* é atualizado para "*working*", indicando que a recepção da imagem está em andamento. Isto ocorre caso já não exista essa imagem com o status "*completed*" na base de dados do *ImageReceiver*. Em seguida, o *hash_file* é devolvido pelo tópico *hash_sender* para o *broker*, sinalizando que a recepção foi iniciada com sucesso, caso a imagem já exista, o *hash_file* é acrescido de "*recused*". O tópico utilizado para devolução de informação ao *ImageReceiver* é o próprio *hash_sender*.
- **Momento 3: Confirmação e envio das partes:** O *ImageSender*, ao receber o *hash_file* de confirmação, verifica seu *status* "*working*" por meio do *FileStatusManager*, porém se a imagem foi recusada pelo *ImageReceiver*, ele atribui "*recused*" e não envia partes de imagem. Com a confirmação, o *ImageSender* prossegue para enviar todas as partes da imagem pelo tópico *hash_sender_orq*, exclusivo para comunicação com o orquestrador.
- **Momento 4: Recepção das partes e tentativa de montagem:** O *ImageReceiver* recebe as partes da imagem e tenta montar a imagem original utilizando o módulo *ImageRestore*. Se todas as partes forem recebidas corretamente, o processo avança diretamente para o Momento 7. Caso contrário, é necessário identificar as partes faltantes, o que neste caso direciona ao Momento 5.
- **Momento 5: solicitação de partes faltantes:** Se houver partes faltantes, o *ImageReceiver* solicita essas partes específicas pelo tópico *hash_sender*. Esta solicitação inclui o *hash_file* da imagem e as identificações das partes que não foram recebidas.
- **Momento 6: Reenvio das partes faltantes:** Ao receber a informação sobre as partes faltantes, o *ImageSender* reenviará as partes solicitadas pelo tópico *hash_sender_orq*. Este processo de solicitação e reenvio (Momentos 4, 5 e 6) repete até que todas as partes da imagem tenham sido recebidas corretamente e a imagem possa ser montada sem qualquer tipo de erro. =
- **Momento 7: sinalização de montagem bem sucedida:** Uma vez que a imagem tenha sido montada com sucesso, o *ImageReceiver* sinaliza a conclusão enviando o *hash_file* pelo tópico *hash_sender*. Além disso, o *status* do *hash_file* é atualizado para "*completed*" pelo *FileStatusManager*.
- **Momento 8: finalização e limpeza:** Finalmente, o *ImageSender* recebe a informação de que a montagem foi bem-sucedida. Ele confere se o *status* do *hash_file* é "*completed*" por meio do *FileStatusManager* e apaga todos

os arquivos temporários gerados, concluindo assim o ciclo de transmissão e recepção da imagem.

O uso de *hashes* para identificação e o gerenciamento de *status* por meio do *FileStatusManager* tem como objetivo garantir que cada etapa do processo seja monitorada e controlada, permitindo a detecção e correção de erros em tempo real, mesmo com a queda de conexão à Internet durante o processo. A divisão em momentos distintos facilita a compreensão e a implementação do sistema. Os momentos podem ser ilustrados pela Figura 17.

Figura 17 – Diagrama de transição de momentos no método V2



Fonte: o autor

Fluxograma de comunicação

Para facilitar o entendimento do diagrama de comunicação, visto na Figura 17, o mesmo é descrito por meio dos momentos no Apêndice C.

4.3.3 Metodologia V2 aplicada nos nós

Envio de imagens

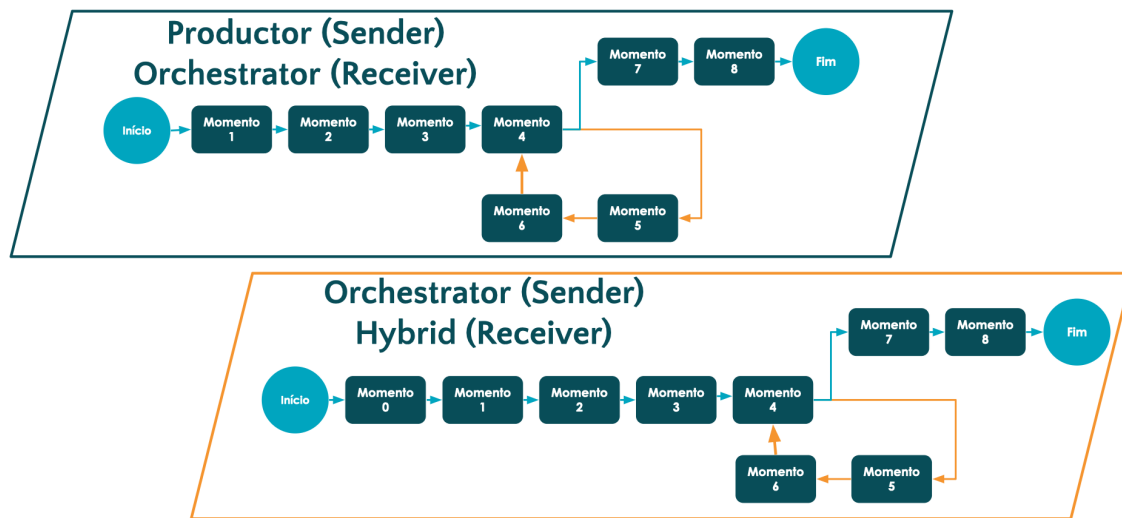
O *NodeProductor* ou *NodeHybrid* parte diretamente do Momento 1 enviando o cabeçalho para o *NodeOrchestrator*.

Solicitações de imagens sob demanda

No caso do *NodeHybrid* existe um Momento 0, no qual ele manda no tópico *type_request* a informação de qual categoria ele gostaria de receber a imagem. E a partir daí o fluxo começa no Momento 1 pelo lado do *NodeOrchestrator*.

As duas configurações de envio podem ser ilustradas conforme Figura 18.

Figura 18 – Diagrama dos momentos na comunicação de envio entre *NodeProductor* e *NodeOrquestrator*, e entre *NodeOrquestrator* e o *NodeHybrid*



Fonte: o autor

4.3.4 Implementação do protocolo MQTT

Os scripts em Python foram criados para implementação da solução baseada no protocolo MQTT, realizando tarefas como comunicação entre nós, fragmentação de imagens, gerenciamento de estados e recuperação de dados. A seguir, são apresentados a estrutura e as funcionalidades desses scripts para os métodos nas versões V1 e V2.

- Fragmentação de imagens em partes menores para transmissão (*ImageSplitter.py*);
- Envio das partes fragmentadas para tópicos MQTT (*ImageSender.py*);
- Recepção e montagem das imagens fragmentadas (*ImageReceiver.py* e *ImageRestore.py*);
- Gerenciamento do estado de transmissão e recepção (*FileStatusManager.py*);
- Coordenação entre diferentes nós, como *NodeOrchestrator.py* e *NodeHybrid.py*.

Estrutura geral

Os scripts utilizam bibliotecas específicas, descritas no arquivo `requirements.txt`:

- `paho-mqtt`: para comunicação MQTT, lidando com publicação e assinatura em tópicos;
- `Pillow`: para manipulação e fragmentação de imagens;

- **robotframework**: utilizado para testes automatizados do sistema.

A integração utiliza o modelo de mensagens MQTT, assegurando comunicação entre os nós distribuídos.

Integração das ferramentas Python

A coordenação entre os nós é gerenciada por *NodeOrchestrator.py* e *NodeHybrid.py*, que implementam:

- Solicitação e envio de imagens por meio de tópicos específicos;
- Recuperação de partes faltantes em caso de falha na transmissão;
- Sincronização de estados entre *ImageReceiver* e *FileStatusManager*.

Os *scripts* desenvolvidos para este trabalho foram disponibilizados publicamente no GitHub ³, com o objetivo de facilitar a reprodutibilidade dos resultados e permitir que outros pesquisadores os utilizem como referência para estudos futuros.

4.4 Implementação de ambiente de rede controlada (*Testbed*)

Para avaliar o desempenho dos dois métodos propostos, foram realizados testes de tráfego de rede em um ambiente controlado (*Testbed*) ⁴. O objetivo foi simular uma rede de IoT com dispositivos que enviam dados, focando especificamente na transmissão de imagens. O teste de desempenho de rede proposto visa analisar o comportamento de envio de imagens sob o tráfego de dados gerado por este envio, bem como avaliar o impacto adicional causado pelo uso do software, que permite aumentar o volume de tráfego de maneira controlada e configurável.

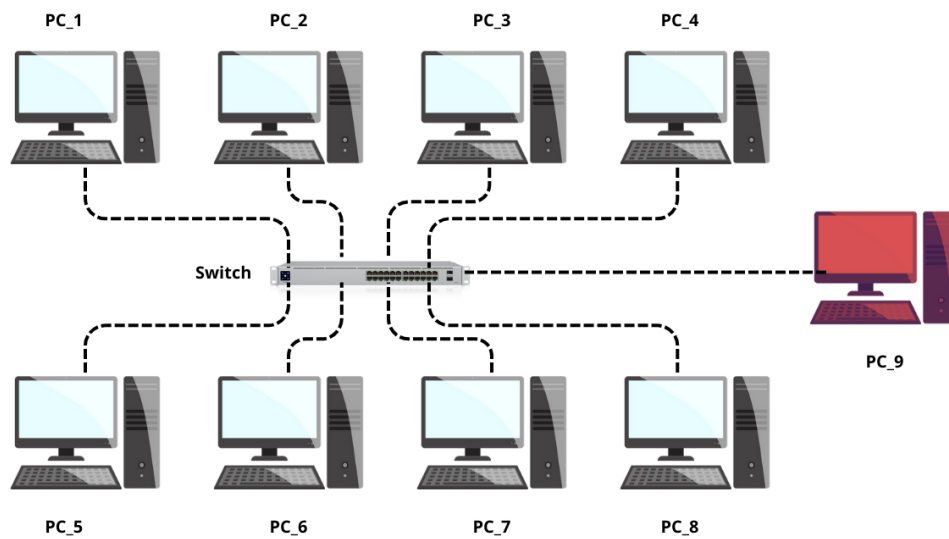
4.4.1 Topologia da rede

O ambiente é composto por nove computadores conectados via cabos Ethernet a um *switch* central, Figura 19. Sete computadores estão configurados como *NodeProductor*, o oitavo como *NodeOrquestrator* e o nono como *Broker*.

³ <https://github.com/eversondj/ImageSparseShare>

⁴ Um *Testbed* é uma plataforma projetada para realizar experimentos, testes ou simulações em um ambiente controlado, geralmente utilizado para validar conceitos, tecnologias, ou sistemas antes de sua aplicação real.

Figura 19 – Topologia do ambiente de rede (Testbed)



Fonte: o autor

4.4.2 Equipamentos utilizados

- 8 Computadores - Intel(R) Core(TM) i3-10100 CPU @ 3.60GHz, 8 GB RAM
- 1 Computador - Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz, com 4 núcleos e 4 threads, 6 GB RAM.
- 1 Switch gerenciado inteligente de 28 portas e 10 gigabits - 24 x 10/100 / 1000BASE-T portas, 4 portas 10G SFP +.

4.4.3 Recursos computacionais utilizados

Estes recursos incluem o sistema operacional, as ferramentas de programação, bibliotecas de software e ferramentas de teste utilizadas para a implementação, execução e análise dos experimentos realizados.

- Ubuntu 22.04;
- Python 3.12.6;
- iPerf3;
- Pillow 9.4.0;
- paho-mqtt 1.6.1;
- robotframework 5.0.1.

4.4.4 Protocolo de comunicação

O protocolo de comunicação empregado é o TCP/IP, que por meio do protocolo de transporte TCP fornece garantia de entrega de pacotes entre dispositivos na rede.

4.4.5 Ferramenta utilizada

- *iPerf3*

É uma ferramenta de rede usada para medir a largura de banda máxima possível em redes IP. O iPerf3 permite análises detalhadas de desempenho ao medir parâmetros como a taxa de perda de pacotes, retransmissões e a taxa de dados úteis (*goodput*), que é a quantidade de dados úteis transmitidos por segundo [Park e Balasubramanian 2023].

4.4.6 Parâmetros utilizados

Os principais parâmetros configurados para o teste são mostrados na Tabela 6.

Tabela 6 – Tabela de parâmetros e métricas de desempenho conforme o *iPerf3*

Parâmetro	Valor
Omit	0
Tempo de teste	1200 (segundos)
Paralelo	Até 8
Tamanho do Pacote (len)	131072 (bytes)
Largura de Banda	500000000 (bps)
Temporizador de Ritmo	1000 (ms)
Porta de Host	5201
Algoritmo de Congestão	cubic
SNDBUF	16384 (bytes)
RCVBUF	131072 (bytes)
Pacing de Aplicação	62500000 bytes/segundo

Fonte: o autor

Explicação dos parâmetros e métricas

- **Omit:** Define o tempo (em segundos) a ser ignorado no início do teste para que o cálculo de desempenho exclua o tempo de setup inicial.

- **Tempo:** Duração total do teste, em segundos, para medir a estabilidade e o desempenho ao longo de um período.
- **Paralelo:** Número de conexões paralelas utilizadas durante o teste, afetando a intensidade da carga sobre a rede.
- **Tamanho do pacote (len):** Tamanho dos pacotes de dados transmitidos, impactando na taxa de transmissão e a eficiência do protocolo.
- **Largura de banda:** Limita a velocidade de transmissão de dados, configurado para um valor alto para testar a capacidade máxima da rede.
- **Temporizador de Ritmo:** Define a frequência do controle de ritmo (*pacing*), regulando o intervalo de envio de pacotes para evitar congestionamentos.
- **Porta de host:** Porta de conexão no host de destino, associada ao serviço de teste de rede.
- **Algoritmo para congestionamento:** Método de controle de congestionamento usado, neste caso o “cubic,” que ajusta a taxa de envio de dados conforme a capacidade da rede.
- **SNDBUF:** Tamanho do buffer de envio no cliente, afetando a quantidade de dados acumulados antes do envio.
- **RCVBUF:** Tamanho do buffer de recepção no cliente, para armazenamento temporário dos pacotes recebidos antes do processamento.
- **Pacing de aplicação:** A taxa de pacing define uma frequência de envio de pacotes, que visa garantir que a largura de banda e o buffer do receptor sejam utilizados de maneira adequada, minimizando perdas decorrentes por sobrecarga.

4.4.7 Coleta de dados

Para este experimento, a rede está configurada de forma que 7 computadores como *NodeProducers* transmitam pacotes de imagens para o *NodeOrquestrator* por meio de um *broker* MQTT. O primeiro computador inicia a transmissão, seguido por um intervalo de tempo definido para o próximo, e assim sucessivamente até que todos os *NodeProducers* estejam enviando suas imagens ao *NodeOrquestrator*. O computador *NodeOrquestrator* será responsável por receber, armazenar e reenviar as imagens recebidas do *broker*. O objetivo é monitorar o desempenho da rede e do *broker* MQTT durante o processo de envio e recepção dos pacotes, com atenção à latência, perdas de pacotes e a capacidade de processamento dos componentes envolvidos.

5 Resultados e discussão

Este capítulo apresenta os resultados obtidos. Inicialmente, detalha-se as modificações realizadas no sistema óptico do PlanktoScope e o impacto desses ajustes na captura de imagens (Seção 5.1). Em seguida, na Seção 5.2, o desempenho da transferência de arquivos sob diferentes condições é abordado, destacando a relação entre o tamanho dos arquivos e o tempo de transferência, bem como a taxa de sucesso. Em seguida, são analisadas as perdas de dados e o impacto de diferentes estratégias para minimizar essas perdas, garantindo a integridade da transmissão. Por fim, na Seção 5.3, são discutidos os resultados da implementação e configuração do sistema em um ambiente de rede controlada, avaliando a confiabilidade da solução em cenários experimentais variados em Testbed.

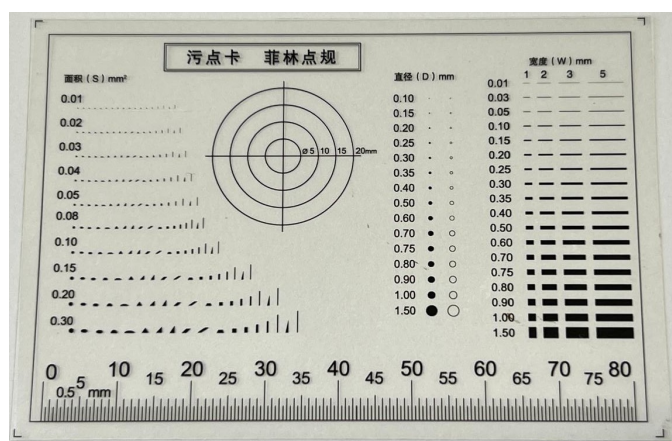
5.1 Sistema PlanktoScope ajustado

Foram realizados ajustes no projeto original do módulo óptico do PlanktoScope, com o objetivo de otimizar o desempenho na captura de imagens. Abaixo seguem os resultados.

Modificações no sistema de captura de imagens

O objetivo do teste realizado com o microscópio digital PlanktoScope foi determinar a configuração ideal de hardware para a captura de imagens de alta resolução.

Figura 20 – Cartão de filme usado para calibração das lentes



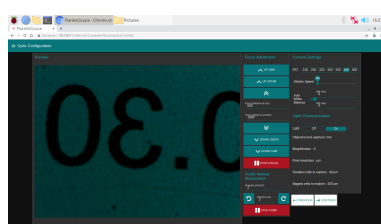
Fonte: o autor

As configurações testadas envolveram diferentes combinações de câmeras Raspberry Pi e lentes M12. Para auxílio da calibração foi usado uma cartão de filme de bitola padrão (Figura 20) utilizado para verificar a precisão de instrumentos ópticos, como microscópios ou câmeras, e para avaliar a qualidade de impressão ou a nitidez de sistemas ópticos.

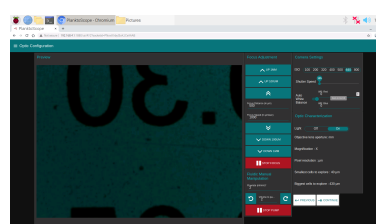
As etapas do teste são ilustradas por meio da Figura 21 e descritas a seguir:

- (a) **Primeiro momento:** Utilização da *Raspberry Câmera V2* com uma *Lente M12 de 16mm*. Esta configuração forneceu um campo de visão amplo com um aumento moderado.
- (b) **Segundo momento:** Manutenção da *Raspberry Camera V2*, substituindo a lente por uma *Lente M12 de 25mm*. Esta lente oferece um campo de visão mais estreito e um aumento maior, ideal para capturar detalhes finos.
- (c) **Terceiro momento:** Substituição da *Raspberry Câmera V2* pela *Raspberry HQ Câmera*, mantendo a *Lente M12 de 16mm*. A HQ Câmera, com um sensor superior, melhora a captação de detalhes.
- (d) **Quarto momento:** Manutenção da *Raspberry HQ Câmera* e troca da lente para a *Lente M12 de 25mm*. Esta configuração final foi determinada como a melhor, combinando alta resolução e capacidade de foco detalhado, resultando em imagens de maior clareza e detalhe.

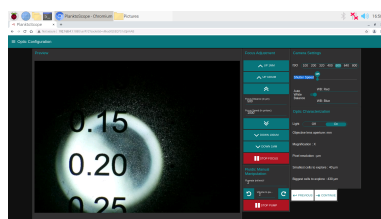
Figura 21 – Imagens dos testes com a mudança no setup das câmeras e lentes



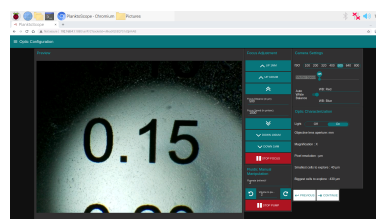
(a) Momento 1



(b) Momento 2



(c) Momento 3



(d) Momento 4

Fonte: o autor

A Tabela 7 apresenta um comparativo entre as características observadas e o desempenho das câmeras e lentes utilizadas nos testes. Esta análise considerou aspectos como resolução, desempenho em condições de baixa luminosidade, campo de visão e detalhamento de objetos. A escolha adequada destes equipamentos é importante para garantir a qualidade das imagens capturadas (Figura 21).

Tabela 7 – Comparativo entre câmeras e lentes na avaliação das imagens

Câmera		Raspberry Câmera V2	Raspberry HQ Câmera
Sensor		Menor resolução, adequado para aplicações de baixa demanda.	Maior resolução, ideal para aplicações que exigem alta qualidade de imagem.
Desempenho em baixa luz	em	Desempenho moderado em condições de baixa luminosidade.	Desempenho superior em condições de baixa luminosidade.
Lente		Lente M12 de 16mm	Lente M12 de 25mm
Campo de Visão		Mais amplo, menor aumento.	Mais estreito, maior aumento.
Detalhamento		Menor detalhamento de objetos próximos.	Maior detalhamento de objetos próximos.

Fonte: o autor

Após análise de imagens, observa-se que a Raspberry HQ Câmera possui um sensor que pode capturar imagens em uma resolução mais alta do que a Raspberry Câmera V2. Quando combinada com a Lente M12 de 25mm, essa configuração maximiza a resolução detalhada da imagem, o que resulta em uma configuração adequada para visualizar características microscópicas com clareza.

A Lente M12 de 25mm proporciona um maior detalhamento ao focar objetos pequenos devido ao seu maior comprimento focal, o que reduz o campo de visão mas aumenta o detalhe visual. Isso é ideal para aplicações de microscopia digital como análise de partículas de microplástico onde a precisão do detalhe é mais importante do que uma visão ampla. A combinação da *Raspberry HQ Câmera* com a *Lente M12 de 25mm* foi a configuração que proporcionou as melhores imagens, destacando-se pela melhor resolução e foco.

5.2 Transferência de arquivos entre os nós

Esta seção analisa o desempenho da transferência de dados sob diferentes condições de teste, considerando as variáveis de tamanho do arquivo e frequência de transferência. Utilizamos arquivos de 100Kb, 256Kb e 512Kb em pacotes¹ de 100 MB (contendo 20, 50 e 100 imagens) e pacotes de 500 MB (445 imagens), com intervalos de um e dois segundos entre as transferências.

Apresentamos o tempo de transferência e a taxa de sucesso, medida pelo percentual de arquivos recebidos. Essas métricas ajudam a identificar gargalos e oportunidades de otimização no processo de transmissão de dados. Também discutimos a relação entre o tamanho do arquivo, o tempo de transferência e a eficiência da transferência para sugerir melhorias no sistema.

A seguir, detalharemos os resultados obtidos, iniciando com a discussão dos gráficos que mostram a relação entre o tamanho do arquivo e o tempo de transferência, seguidos pelas tabelas que resumem a eficiência das transferências, incluindo detalhes sobre o sucesso das operações e o tempo total gasto. Esta análise nos permite tirar observações sobre o desempenho do sistema de transferência de dados e sugerir melhorias para aumentar a eficiência em aplicações futuras.

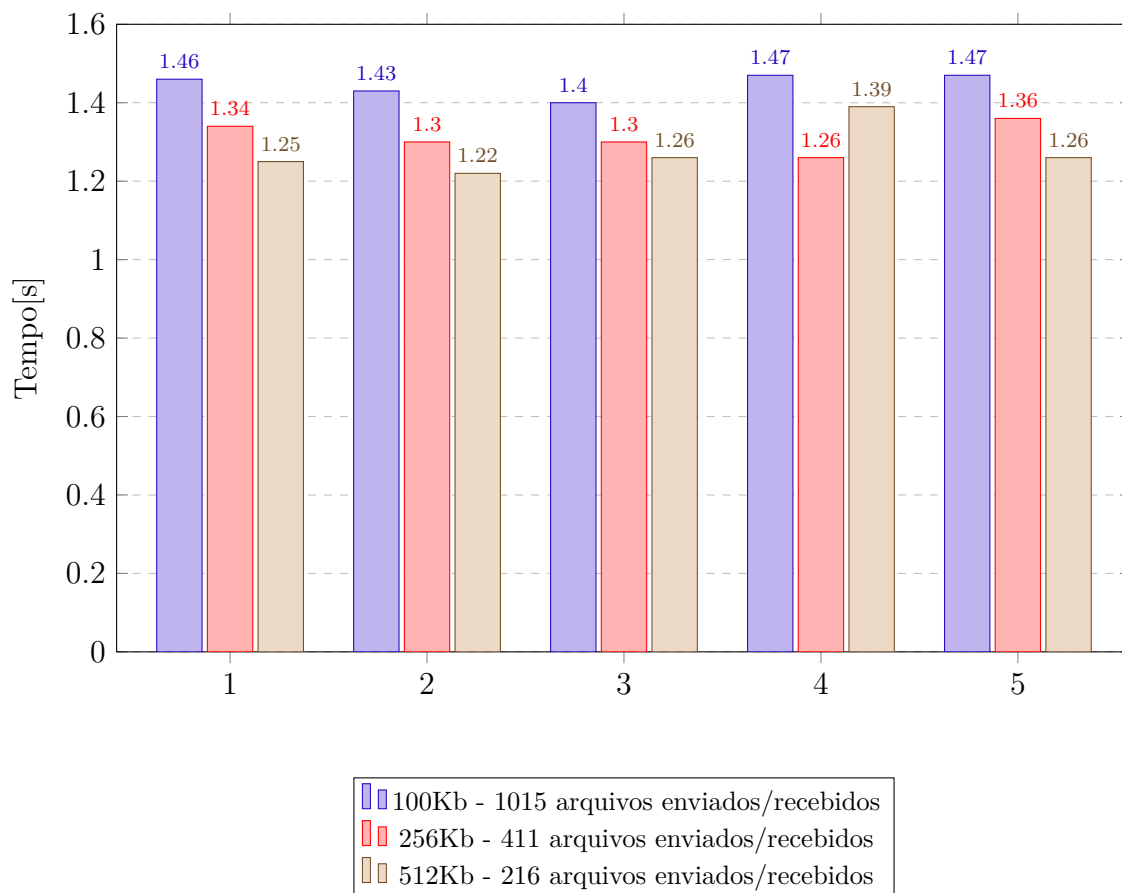
Resultados das transmissões das imagens

O experimento foi conduzido utilizando comunicação ponto-a-ponto entre dois nós, especificamente o *NodeProducer* e o *NodeOrchestrator*. Os testes também foram realizados com os *subscribers* conectados ao servidor durante a transmissão. No entanto, identificou-se uma limitação: caso o *subscriber* se conecte após o término do envio, não há como receber as imagens previamente enviadas.

¹ O pacote de dados aqui refere-se ao conjunto de arquivos para transmissão de imagens.

Pacote de 100 MB e tempo entre imagens = 2s

Figura 22 – Desempenho da transferência de dados com arquivos de 20 imagens e intervalos de 2 segundos

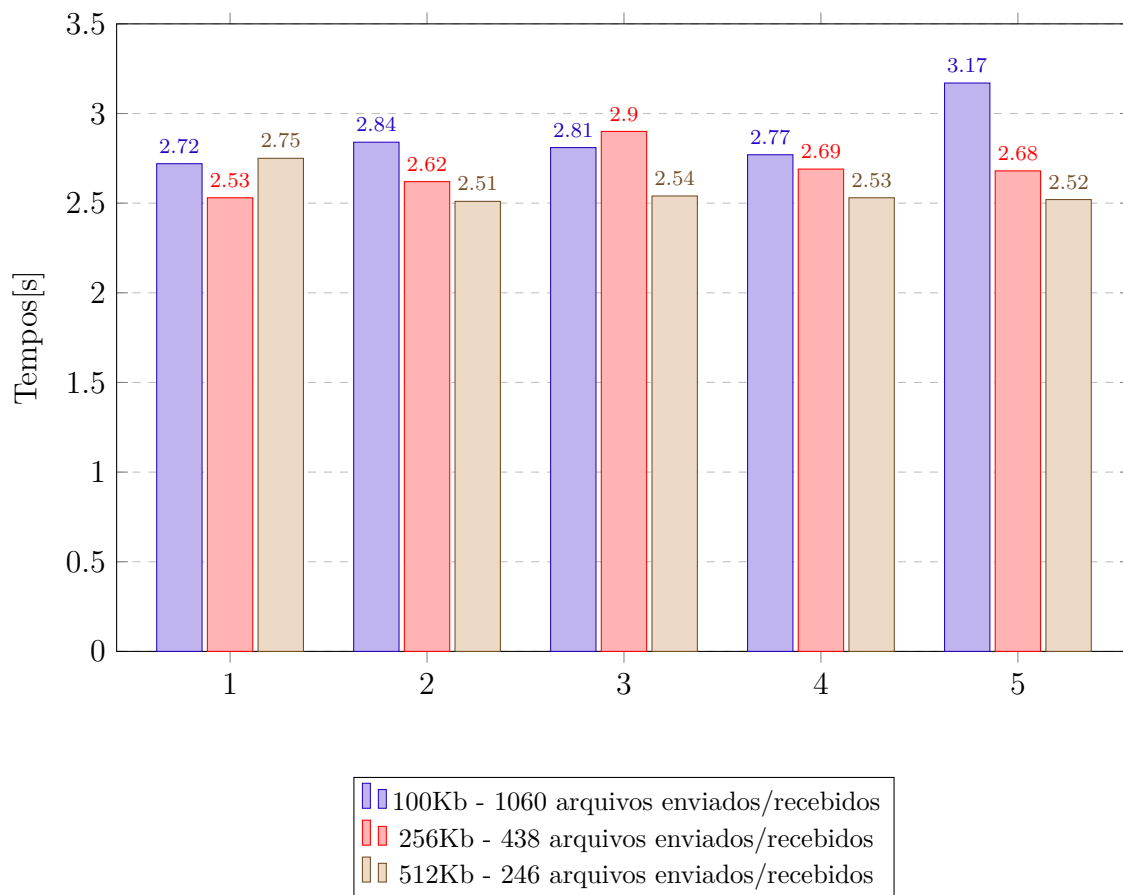


Fonte: o autor

A Figura 22 mostra o desempenho da transferência de dados para diferentes tamanhos de arquivo (100Kb, 256Kb e 512Kb) com intervalos de 2 segundos entre cada transferência. Os resultados indicam que arquivos maiores (256Kb e 512Kb) levam um tempo igual ou menor para serem transferidos do que arquivos de 100Kb. A variação mínima no tempo entre as execuções sugere estabilidade da rede. Isso pode indicar que arquivos maiores podem melhorar a eficiência da rede, reduzindo o *overhead* relativo.

As variações observadas, sugerem que outros fatores podem estar influenciando a eficiência da transmissão, indicando por exemplo que características como a configuração da rede, a eficiência do protocolo de transmissão, ou até mesmo a carga no sistema no momento do teste podem ter impactos na transmissão. Ajustar os tamanhos dos pacotes de imagens pode otimizar o sistema de transferência, levando à melhorias no desempenho.

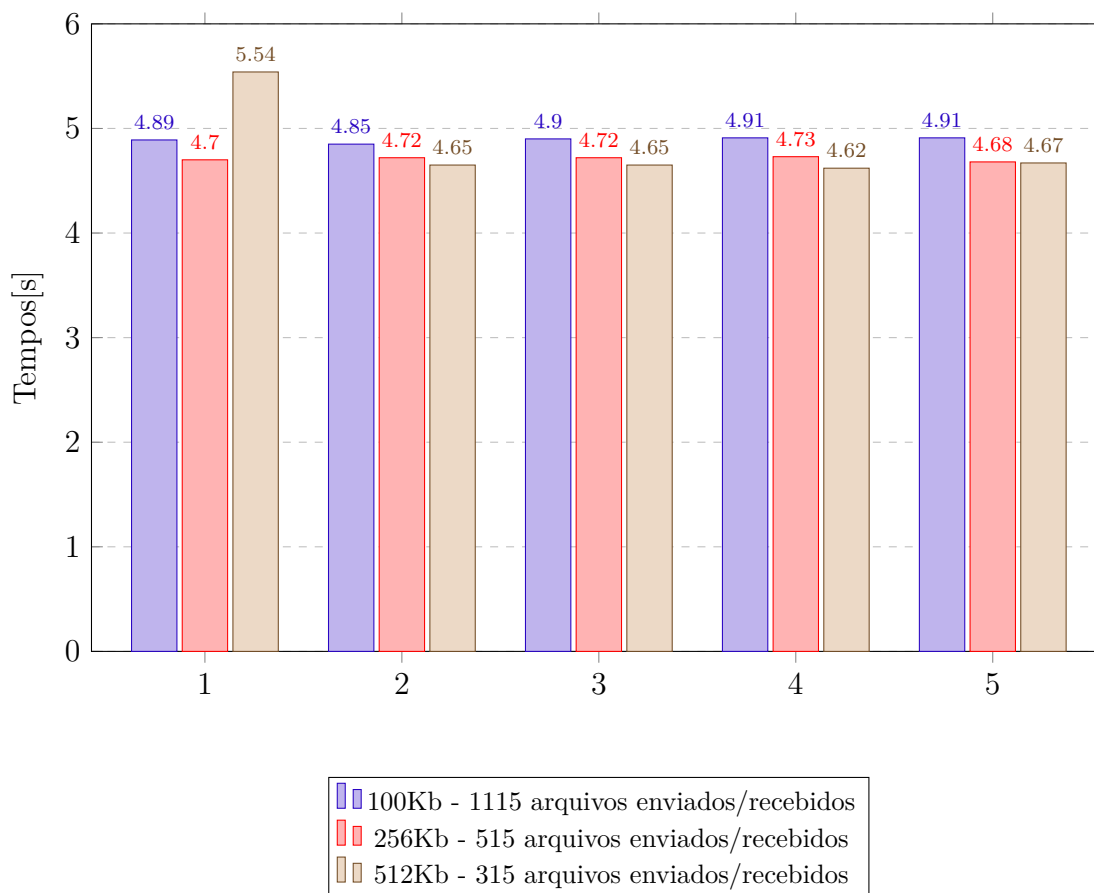
Figura 23 – Desempenho da transferência de dados com arquivos de 50 imagens e intervalos de 2 segundos



Fonte: o autor

Os dados da Figura 23 mostram poucas evidências para estabelecer uma correlação entre o tamanho do arquivo e o tempo de transmissão, com variações que podem indicar que fatores externos influem na eficiência da transmissão. Principalmente, para os arquivos de 512Kb, não houve uma tendência observada de tempos de transmissão mais longos. Esse resultado indica que a configuração da rede, a eficiência do protocolo e as condições do sistema durante os testes podem afetar os resultados.

Figura 24 – Desempenho da transferência de dados com arquivos de 100 imagens e intervalos de 2 segundos

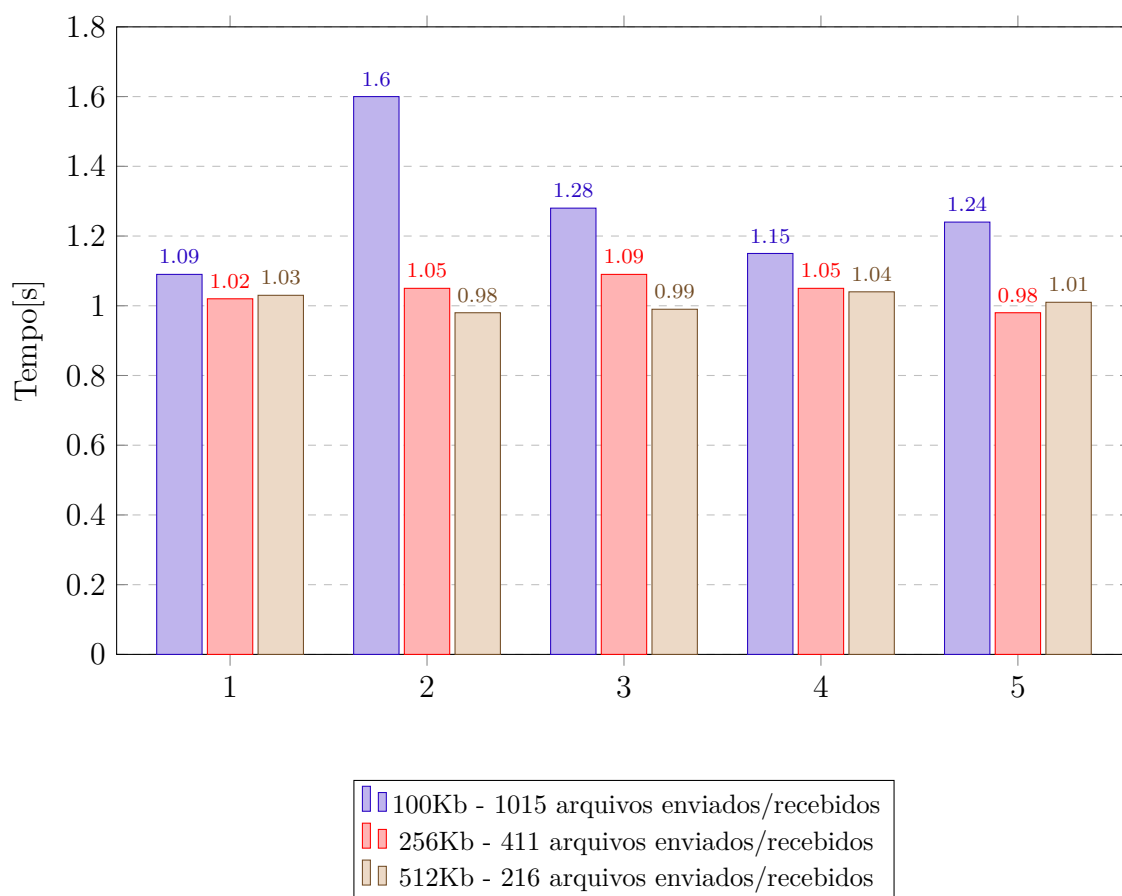


Fonte: o autor

O tempo de processamento dos arquivos tende a aumentar com o tamanho dos mesmos (Figura 24). Fragmentos de dados de 100Kb, mesmo com um grande número de arquivos (1115), tiveram tempos de processamento mais prolongados, sugerindo que transferir numerosos arquivos menores pode ser menos adequado em termos de tempo (mais overhead) do que transferir menores quantidades de arquivos maiores (menos overhead). Por outro lado, arquivos maiores (512Kb), apesar de estarem em menores quantidades (315 arquivos), inicialmente tiveram tempos de processamento altos, que melhoraram nas tentativas subsequentes. Isso é possivelmente um efeito direto da otimização no processo de transferência ou variações nas condições da rede.

Esses resultados mostram que tanto o tamanho quanto a quantidade de arquivos impactam significativamente a eficiência dos processos de transferência de dados no sistema Pub/Sub e têm efeitos na gestão de dados e otimização da rede.

Figura 25 – Desempenho da transferência de dados com arquivos de 20 imagens e intervalos de 1 segundos

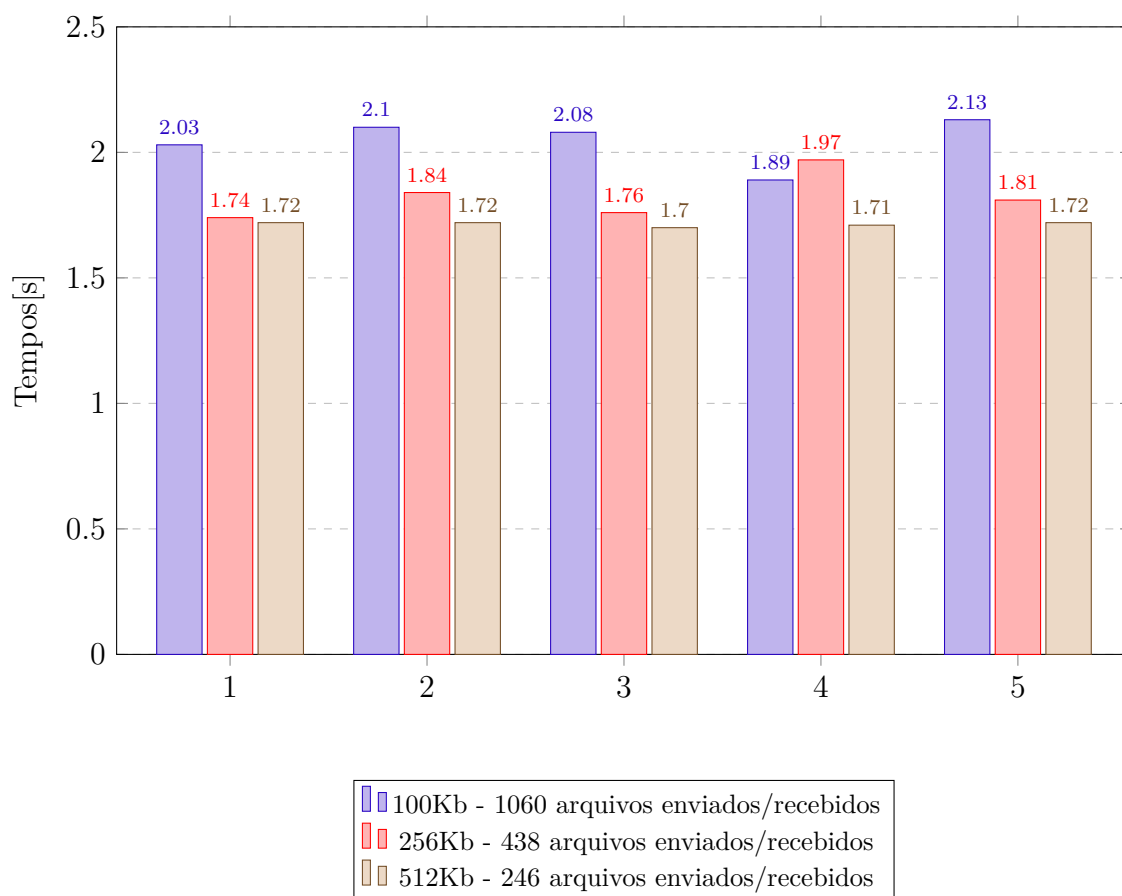


Fonte: o autor

Observamos variações nos tempos de processamento para diferentes tamanhos de arquivo (Figura 25). Curiosamente, os arquivos de 512Kb consistentemente têm tempos de processamento mais curtos do que os arquivos de 100Kb e 256Kb. Isso sugere que o manuseio de arquivos maiores pode ser mais indicado. Os dados indicam que o maior tempo de processamento foi para arquivos de 100Kb no segundo grupo de dados, enquanto o mais curto foi para arquivos de 512Kb.

O gráfico mostra como o tamanho do arquivo afeta o tempo de processamento médio em um conjunto específico de dados. A análise sugere que, para este conjunto de dados, arquivos maiores podem ser processados de forma mais adequada em termos de tempo por arquivo, apesar de haver menos arquivos no total, o que pode ser relevante para otimizar o processamento de dados em aplicações semelhantes.

Figura 26 – Desempenho da transferência de dados com arquivos de 50 imagens e intervalos de 1 segundos

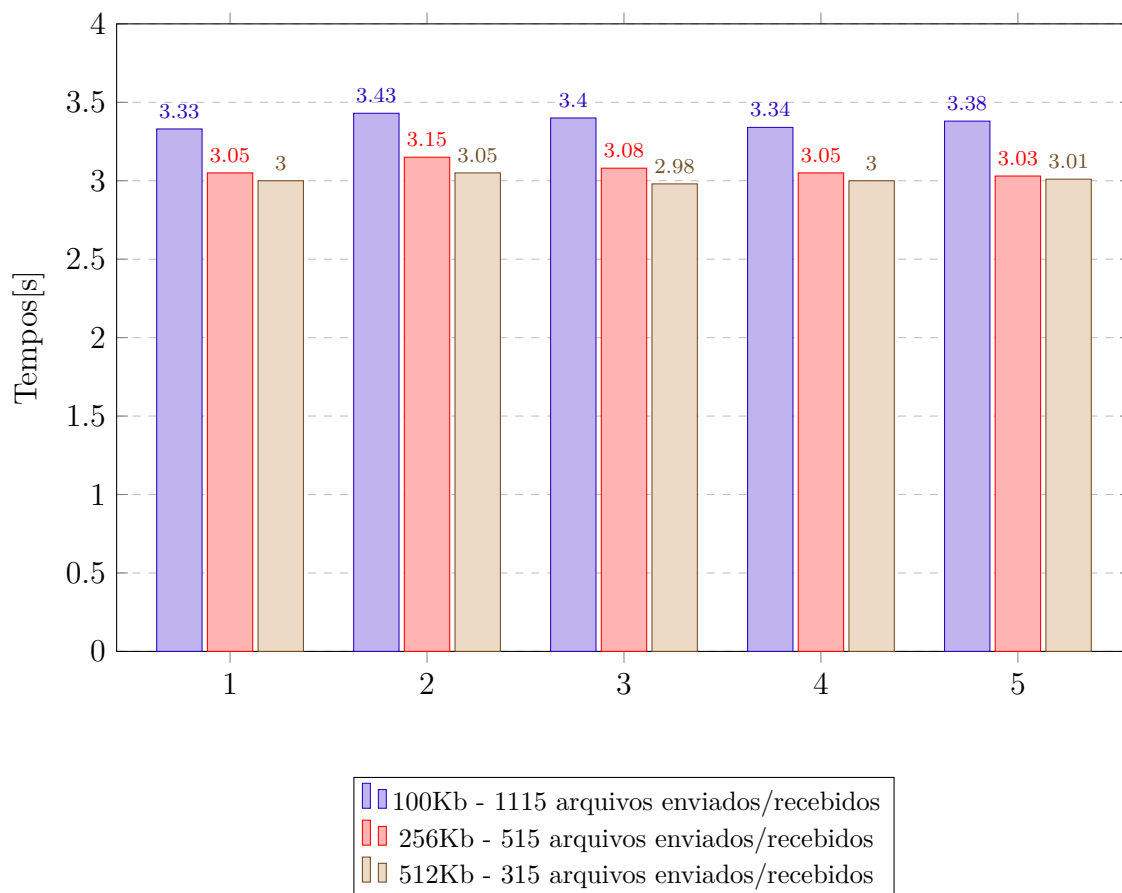


Fonte: o autor

A Figura 26 mostra que os arquivos de 100Kb levam mais tempo para serem transferidos, entre 1,89s e 2,13s, enquanto os arquivos de 256Kb e 512Kb são transferidos em menos de 2 segundos. Apesar de serem maiores, os arquivos de 512Kb têm tempos de transferência semelhantes aos de 256Kb, indicando uma gestão mais segura de dados para tamanhos maiores de arquivo.

Os tempos de transferência para arquivos de 512Kb estão em torno de 1,72 segundos. Os tempos de transferência para arquivos de 256Kb variam de 1,76s a 1,97s, e os de 100Kb mostram a maior variação. Esta análise sugere que arquivos maiores são transferidos de maneira mais efetiva.

Figura 27 – Desempenho da transferência de dados com arquivos de 100 imagens e intervalos de 1 segundos

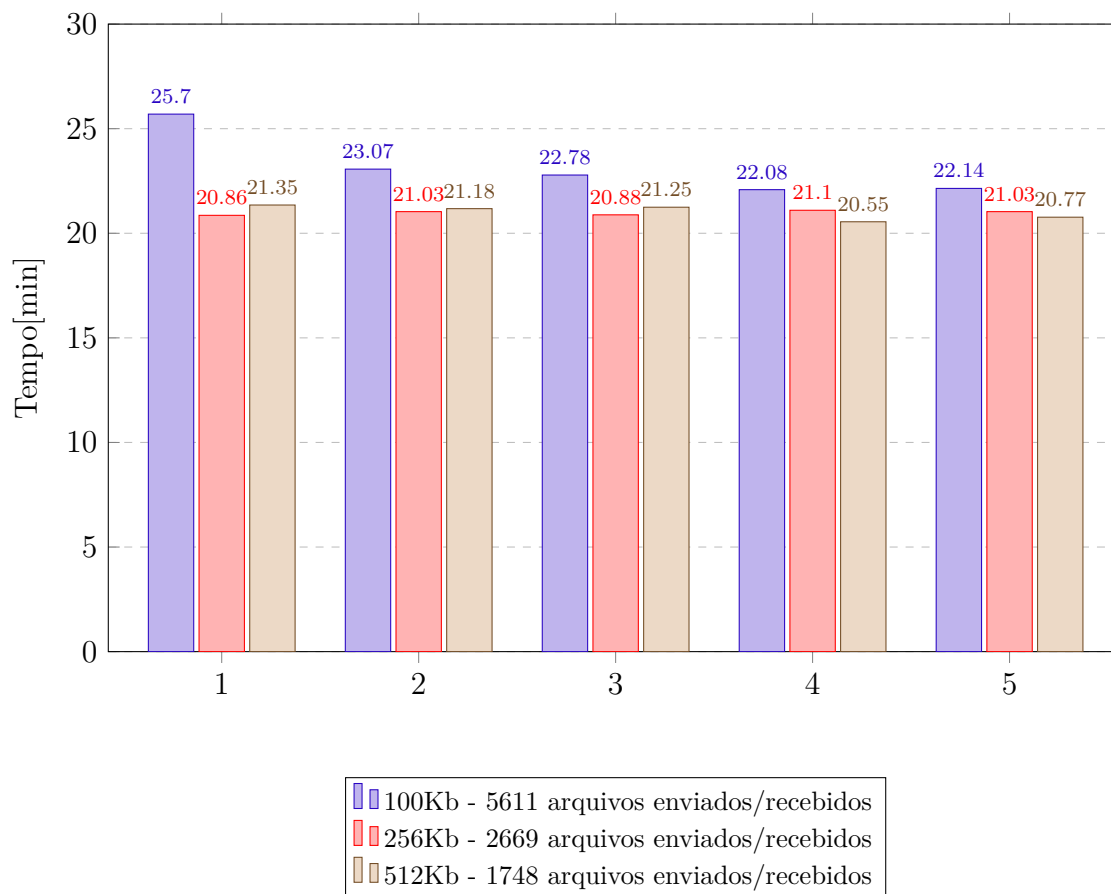


Fonte: o autor

Observamos uma transferência de dados estável para todos os tamanhos de arquivo ao longo de conjunto de dados, (Figura 27). Arquivos de 100Kb levaram mais tempo (3,33 a 3,43s) devido ao gerenciamento de múltiplos arquivos de tamanho pequeno. Por outro lado, arquivos de 256Kb e 512Kb tiveram tempos mais curtos (3,03 a 3,15s e 2,98 a 3,05s, respectivamente), indicando um gerenciamento mais ajustado com uma entrada de menos arquivos. Isso destaca novamente a eficácia do uso de arquivos maiores para a transferência de dados em sistemas otimizados para tais tamanhos.

Dataset de 500MB

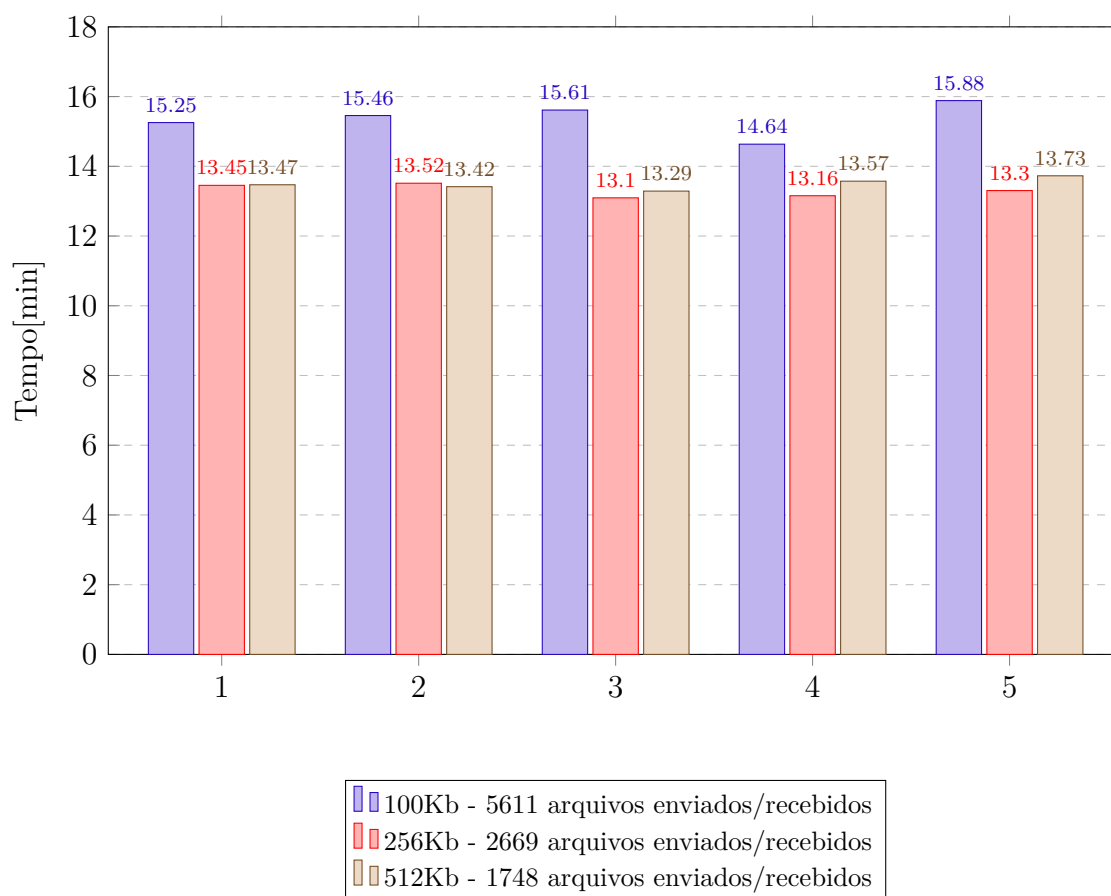
Figura 28 – Desempenho da transferência de dados com arquivos de 445 imagens e intervalos de 2 segundos



Fonte: o autor

A Figura 28 sugere que os tempos de processamento são contra-intuitivos, com arquivos de 100 Kb levando mais tempo (25,7 a 22,1 minutos) do que arquivos de 256 Kb e 512 Kb, que mostram tempos mais consistentes e ligeiramente menores (20,5 a 21,3 minutos). A eficiência não é observada ao aumentar linearmente o tamanho do arquivo, demonstrando oportunidades de otimizações para o manuseio de arquivos menores e ajustes na agrupação ou transmissão de arquivos dentro do sistema.

Figura 29 – Desempenho da transferência de dados com arquivos de 445 imagens e intervalos de 1 segundos



Fonte: o autor

Na Figura 29, são comparados os tempos de processamento para diferentes tamanhos de arquivo. Arquivos menores de 100Kb levaram de 14,6 a 15,9 minutos, enquanto arquivos de 256Kb e 512Kb levaram de 13,1 a 13,7 minutos, apesar de serem maiores.

Estudo das perdas

A Tabela 8 apresenta a frequência das transmissões de dados que impacta diretamente a taxa de sucesso e o tempo de transferência durante interrupções intencionais da conectividade à Internet. O método V1 proporcionou uma taxa de sucesso média de 91%.

Para arquivos de 100 Kb, reduzir o intervalo entre transmissões de 2 segundos para 1 segundo permitiu o aumento da taxa de sucesso. No entanto, arquivos maiores de 512 Kb mostraram taxas de sucesso reduzidas com maior frequência de envio. Os tempos de transferência variam de 13 a 25 minutos, com intervalos mais curtos

Tabela 8 – Análise de eficiência na transferência de dados: relação entre tamanho das partes, taxa de sucesso e tempo de transferência

Tamanho das partes [Kb]	Qtd. de imagens	Tempo entre fotos [s]	Partes enviadas	Partes recebidas	Sucesso [%]	Tempo de transferência [min]
100	445	2	5611	4774	85,08	25,2215
		1		5277	94,04	17,2133
256	445	1		2503	93,78	20,8812
		1	2669	2505	93,85	21,0342
		2		2492	93,36	13,0958
		2		2462	92,24	13,1563
512	445	2	1748	1670	93,53	20,55
		1		1499	85,75	13,5732

Fonte: o autor

geralmente resultando em tempos de transferência reduzidos, apesar de taxas de sucesso menores para arquivos maiores. Equilibrar a frequência de envio e a tolerância a falhas da rede é importante para otimizar a confiabilidade da transmissão de dados em condições instáveis de rede. A seguir, uma análise dos resultados é apresentada:

Tamanho e taxa de sucesso:

- Para arquivos de **100 Kb**, observa-se uma variação na taxa de sucesso de 85,08% a 94,04%. Curiosamente, a taxa de sucesso é maior (94,04%) quando o tempo entre imagens é reduzido para 1 segundo, indicando melhor desempenho sob maior frequência de envio, apesar das interrupções de conexão.
- Para arquivos de **256 Kb**, as taxas de sucesso são bastante consistentes, variando de 92,24% a 93,85%. Isso sugere que a transmissão de arquivos maiores não foi tão impactada pela qualidade da conexão, comparando com o cenário de arquivos menores.
- Para arquivos de **512 Kb**, a taxa de sucesso varia de 85,75% a 93,53%. A taxa é menor quando a frequência de envio é maior (1s), o que pode indicar que arquivos maiores enfrentam mais dificuldades com interrupções frequentes.

Tempo de transferência:

- O tempo de transferência varia de 13min 6s a 25min 13s, dependendo do tamanho do arquivo e da frequência de envio. Em geral, percebe-se que intervalos

menores entre envios (1s) tendem a reduzir o tempo total de transferência, mesmo com taxas de sucesso ligeiramente menores para arquivos maiores.

Diferença entre pacotes enviados e recebidos:

- Para arquivos de 100 Kb com intervalo de 2 segundos, foram enviados 5611 pacotes e recebidos 4774, resultando em uma perda de 837 pacotes.

Observações gerais:

- A redução no tempo entre envios parece ter um efeito positivo sobre o tempo total de transferência, mas pode aumentar a taxa de falha em arquivos maiores. Isso sugere que um equilíbrio precisa ser encontrado entre a frequência de envio e a tolerância a falhas de rede para otimizar a eficiência da transferência.
- O uso de uma taxa alta de particionamento pode ocasionar quantidade de iterações indesejadas e diminuir eficiência da rede.
- O experimento replica ambiente de condições de rede instáveis para entender como diferentes configurações de tamanho e frequência de envio afetam a confiabilidade e a eficiência da transferência de dados.

Esta análise pode ajudar a melhorar as estratégias de transmissão de dados em redes instáveis, especialmente em aplicações que precisam de transmissão contínua, como monitoramento remoto ou *streaming* de mídia em condições adversas.

Análise com a correção das perdas (V2), 100% de restauração

A Tabela 9 mostra que o sistema de transmissão de imagens é adequada, com integridade de dados consistente mesmo sob interrupções de conexão. O tempo de transmissão aumenta com o número de imagens, provavelmente devido a fatores como largura de banda ou complexidade dos dados. A maioria dos processos foi concluída sem a necessidade de retransmissão de pacotes, demonstrando um controle de erros adequado. Apesar de um caso de retransmissão (11 pacotes de um total de 1812), o desempenho geral permaneceu satisfatório.

Consistência de dados:

- As colunas mostram que o número de imagens no início e no fim do processo é o mesmo em todas as entradas, o que indica que não houve perda de dados.

Tabela 9 – Resultados das coletas: relação entre quantidade de imagens, tempo total e reenvio de partes

Qtd. de imagens	Tempo total	Partes enviadas	Reenvio de partes
20	3min 29s	235	0
20	4min 8s	235	0
50	10min 57s	292	0
50	11min 5s	293	1
100	19min 29s	315	0
100	19min 26s	315	0
445	1h 20min 5s	1801	0
445	20min 55s	1812	11

Fonte: o autor

Isso sugere que o sistema possui mecanismos para garantir a integridade dos dados mesmo em situações de interrupção da conexão.

Duração e quantidade de imagens:

- A duração do processo aumenta proporcionalmente com a quantidade de imagens enviadas. O envio de 20 imagens levou cerca de 3 a 4 minutos, enquanto o envio de 445 imagens variou entre aproximadamente 21 minutos a 1 hora e 20 minutos, indicando uma variação dependente de outros fatores, como possivelmente a largura de banda disponível ou a complexidade dos dados das imagens.

Pacotes enviados vs. reenviados:

- Para a maioria dos casos, o número de pacotes "Reenviados" é 0, o que indica que o processo de envio foi completado sem a necessidade de retransmissões, mesmo com interrupções de conexão. Isso mostra a robustez do mecanismo de restauração e a eficácia do controle de erro. No entanto, na última entrada (445 imagens, 20min 55s), houve 11 pacotes que precisaram ser reenviados. Isso ainda é um número muito baixo em comparação ao total de pacotes enviados (1812), o que continua indicando um alto grau de eficiência.

O sistema de envio é resiliente e capaz de lidar com interrupções sem perda de informações. A presença de algumas retransmissões em um dos experimentos não compromete a avaliação geral da eficácia do sistema. Essa robustez é crucial para

aplicações que demandam alta confiabilidade e integridade de dados em ambientes com conexões instáveis.

Os resultados indicam que o tamanho dos arquivos afeta o tempo de transferência de dados. Arquivos maiores de 512Kb apresentam desempenho superior em termos de eficiência e consistência comparativamente aos arquivos menores de 100Kb e 256Kb. Observou-se que arquivos de 512Kb mantêm tempos de processamento reduzidos mesmo com intervalos de envio de apenas 1 segundo, demonstrando robustez na manipulação de grandes volumes de dados.

Em suma, pode-se confirmar dentro do contexto dos experimentos realizados que o uso de arquivos maiores é uma estratégia adequada para melhorar a eficiência da transferência de dados, proporcionando uma base para otimização contínua do desempenho no envio de mensagens em um sistema *Publish/Subscribe* com protocolo MQTT. A análise também revelou que a confiabilidade do sistema é mantida mesmo sob condições adversas, como interrupções intencionais na conexão à Internet. Isso demonstra que a alteração realizada obteve o sucesso desejado, garantindo a restauração de 100% das imagens (Tabela 9).

5.3 Resultados da validação em ambiente de rede controlada

Para melhorar o entendimento dos limites da solução, adotou-se uma abordagem baseada em cenários, simulando problemas e avaliando o desempenho da transmissão de imagens por meio do protocolo MQTT em condições especificadas a priori. Os métodos propostos foram avaliados por meio de testes de tráfego realizados em um ambiente de *testbed* controlado, representando uma rede IoT voltada para a transmissão de imagens.

Esta seção apresenta os resultados em diferentes cenários, considerando o papel de cada nó integrante à rede — *NodeProductor*, *NodeOrquestrator* e *NodeHybrid*. A avaliação considera variáveis como o tamanho dos arquivos e a frequência de transferência, permitindo uma análise detalhada da eficácia e resiliência do sistema de comunicação.

O experimento também investigou o impacto do uso de software configurável, que possibilita o controle e o aumento do tráfego, no comportamento da transmissão de imagens. As configurações utilizadas estão detalhadas na Seção 4.4.

Cenário 1

Este cenário foi projetado para a transmissão contínua de pacotes de imagem, sem interrupções de Internet ou interferências de tráfego artificial gerado por aplica-

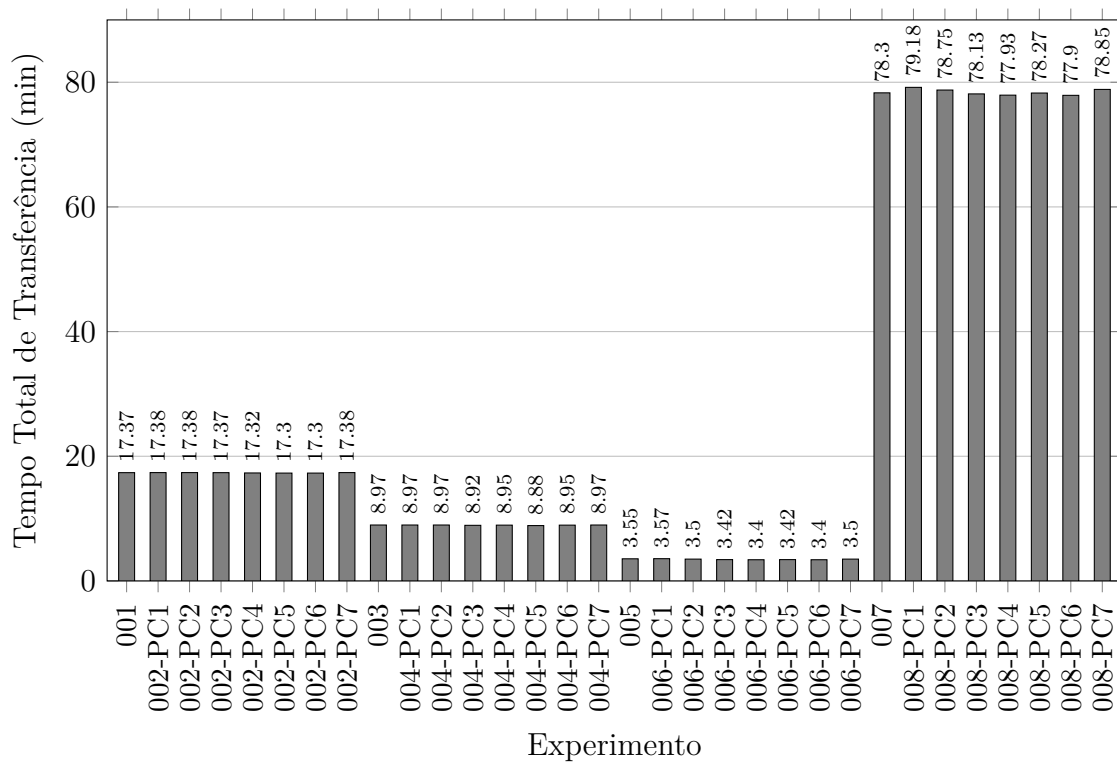
ções adicionais. A Tabela 10 apresenta a estrutura do Cenário 1 dividida em oito experimentos.

Tabela 10 – Parâmetros utilizados nos experimentos 001 a 008 no cenário 1

Exp.	PCs	Pacote (MB)	Número de Imagens	Tamanho da Imagem (MB)
001	1	100	100	1
002	7	100	100	1
003	1	100	50	2
004	7	100	50	2
005	1	100	20	5
006	7	100	20	5
007	1	500	445	-
008	7	500	445	-

Fonte: o autor

Figura 30 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, sem interrupções de Internet e sem injeção artificial de pacotes



Fonte: o autor

Na Figura 30, a variação no tempo de transferência dentro de cada grupo é mínima, indicando resultados consistentes entre as configurações. Os grupos 001 e

002 (PC1 a PC7) apresentaram tempos de transferência entre 17,3 e 17,38 minutos, com diferenças mínimas entre os experimentos, refletindo condições de transferência estáveis. De forma similar, os grupos 003 e 004 (PC1 a PC7) mostraram variação leve, com tempos entre 8,85 e 8,97 minutos, sugerindo uniformidade nas configurações e no ambiente experimental. No grupo 006 (PC1 a PC7), os tempos de transferência variaram entre 3,4 e 3,57 minutos, mantendo eficiência de transferência consistente. Por fim, os grupos 007 e 008 (PC1 a PC7) registraram os maiores tempos de transferência, variando de 78,13 a 79,18 minutos, mas com pequenas variações, indicando condições estáveis. Essas observações confirmam a consistência dos tempos de transferência em cada configuração experimental.

Tabela 11 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 001 a 008

Exp.	Arquivos Enviados	Arquivos Recebidos	Imagens Restauradas
001	315	315	100
002	315	315	100
003	292	292	50
004	292	292	50
005	235	235	20
006	235	235	20
007	1801	1801	445
008	1801	1801	445

Fonte: o autor

A Tabela 11 apresenta os resultados dos experimentos. Nos experimentos 001 e 002, todos os 315 arquivos enviados foram recebidos com sucesso, alcançando 100% de restauração das imagens. Nos experimentos 003 e 004, 292 arquivos foram enviados e recebidos, restaurando 50 imagens em cada caso. Os experimentos 005 e 006 envolveram a transferência de 235 arquivos, resultando na restauração de 20 imagens. Por fim, nos experimentos 007 e 008, 1801 arquivos foram enviados e recebidos, levando à restauração de 445 imagens em cada experimento. A análise da Figura 30 e da Tabela 11 indica que a aplicação apresentou desempenho consistente em um ambiente controlado, com variação mínima nos tempos de transferência e altas taxas de restauração de imagens em todos os experimentos.

Cenário 2

No Cenário 2, a conexão com a Internet é interrompida desconectando fisicamente o cabo Ethernet dos computadores. Essa configuração permite avaliar o impacto da interrupção da rede no fluxo de dados entre os componentes do sistema

e analisar a resiliência do protocolo de comunicação utilizado. A Tabela 12 apresenta a divisão do Cenário 2 em quatro experimentos distintos.

Tabela 12 – Parâmetros utilizados nos experimentos 009 a 012 no cenário 2

Exp.	PCs	Pacote (MB)	Número de Imagens	Tamanho da Imagem (MB)
009	1	100	100	1
010	1	100	100	1
011	1	100	100	1
012	7	100	100	1

Fonte: o autor

No experimento 009, o cabo Ethernet é desconectado do computador configurado como *NodeProductor*. No experimento 010, o cabo Ethernet é desconectado do broker. No experimento 011, a desconexão ocorre no *NodeOrquestrator*. Finalmente, no experimento 012, a desconexão é feita no *NodeOrquestrator* enquanto todos os sete *NodeProductor* estão enviando imagens simultaneamente.

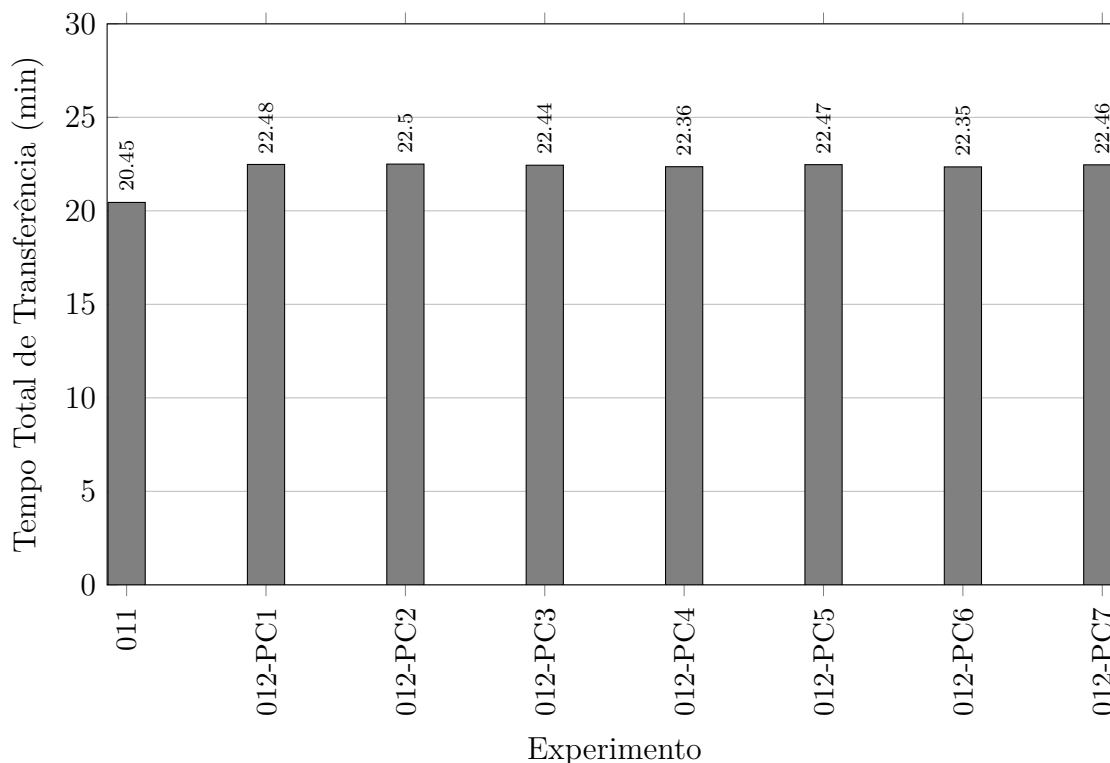
Tabela 13 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 009 a 012

Exp.	Arquivos Enviados	Arquivos Recebidos	Imagens Restauradas
009	150	69	46
010	150	75	50
011	315	315	100
012	315	315	100

Fonte: o autor

A Tabela 13 apresenta os resultados dos experimentos 009 e 010, nos quais 150 arquivos foram enviados antes do cabo Ethernet ser desconectado do *NodeProductor* (experimento 009) e do broker (experimento 010). Nesses experimentos, entre 69 e 75 arquivos foram recebidos com sucesso, respectivamente, levando à restauração parcial das imagens: 46 no experimento 009 e 50 no experimento 010. Após a reconexão do cabo Ethernet, a transmissão de imagens não foi retomada nesses casos. Em contraste, nos experimentos 011 e 012, onde o cabo Ethernet foi desconectado e reconectado no *NodeOrquestrator*, todos os 315 arquivos enviados foram recebidos com sucesso, resultando na restauração completa das 100 imagens em cada caso.

Figura 31 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com interrupções de Internet e sem injeção artificial de pacotes



Fonte: o autor

A Figura 31 mostra que o desempenho da aplicação em um ambiente controlado permanece consistente quando o cabo Ethernet é desconectado e reconectado no *NodeOrquestrator*. No entanto, quando a desconexão ocorre no *NodeProductor* ou no broker, a transmissão de imagens é interrompida, levando à restauração parcial das imagens, conforme observado nos experimentos 009 e 010 (Tabela 13). Esses resultados mostram a funcionalidade de buferização do *NodeOrquestrator*, garantindo entrega de dados em cenários com perda total de conectividade (desconexão do cabo ethernet).

Cenário 3

No Cenário 3, são realizados experimentos com alterações nos nomes de pastas, nomes de arquivos e com repetição de imagens. A Tabela 14 apresenta a divisão do Cenário 3 em três experimentos distintos.

No experimento 013, dois *NodeProductor* enviam imagens para o *NodeOrquestrator*, mas os nomes das imagens e dos arquivos são idênticos. A principal distinção está no nome do pacote, que diferencia as fontes, como "Amostra PC1" e "Amostra

Tabela 14 – Parâmetros utilizados nos experimentos 013 a 015 no cenário 3

Exp.	PCs	Pacote (MB)	Imagens Número	Tamanho da Imagem (MB)
013	2	100	50	2
014	2	100	50	2
015	2	100	50	2

Fonte: o autor

PC2". No experimento 014, o experimento 013 é repetido, com a observação de que os arquivos no *NodeOrquestrator* não foram duplicados, nem nas pastas nem no conteúdo dentro delas. No experimento 015, o experimento 014 é repetido, mas desta vez, as pastas de envio nos *NodeProductor* têm o mesmo nome, como ambas sendo rotuladas como "Amstre PC1." Foi observado que as pastas não foram duplicadas no *NodeOrquestrator*.

Tabela 15 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 013 a 015

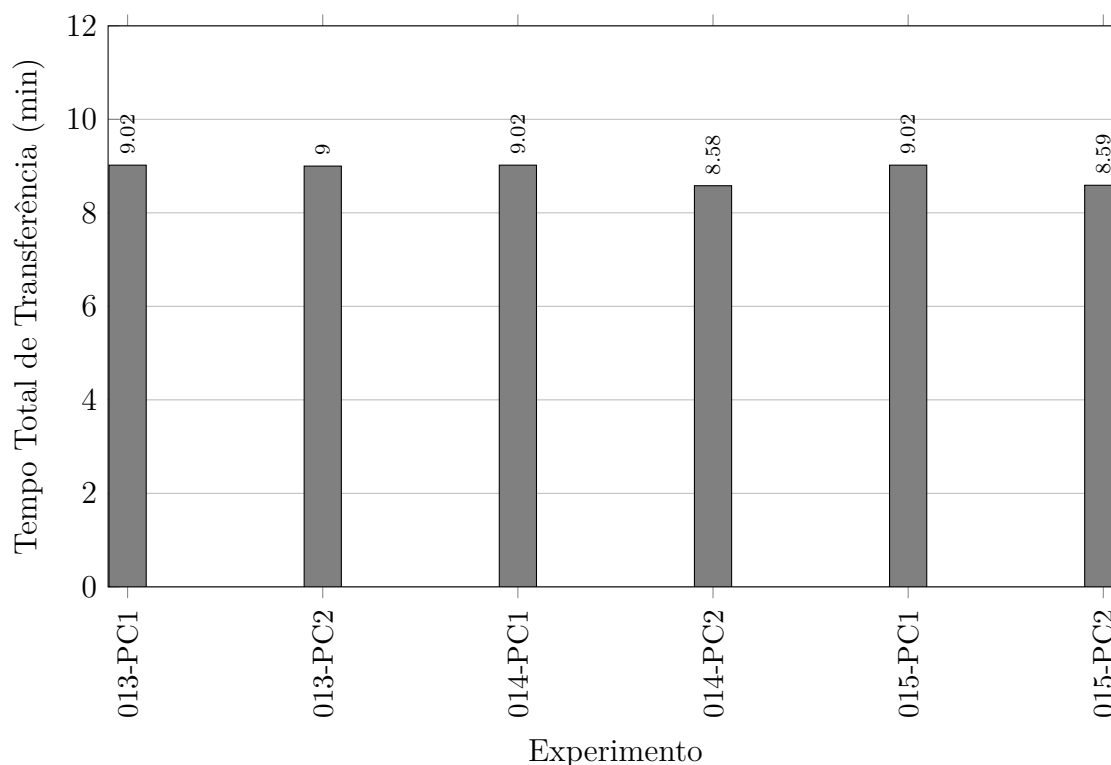
Exp.	Arquivos Enviados	Arquivos Recebidos	Imagens Restauradas
013	292	292	50
014	292	292	50
015	292	292	50

Fonte: o autor

A Tabela 15 mostra que, em todos os experimentos, todos os arquivos enviados foram recebidos, resultando em uma restauração completa de todas as imagens, sem duplicação no armazenamento.

A análise da Figura 32 indica que o desempenho da aplicação em um ambiente controlado permanece estável, com variações mínimas nos tempos de transferência de imagens em todos os experimentos.

Figura 32 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com alteração de nome de arquivo, nome de pasta e com repetição



Fonte: o autor

Cenário 4

No Cenário 4, são realizados experimentos de transmissão de pacotes utilizando o software *iPerf3*. Os parâmetros utilizados estão detalhados na Seção 4.4.6 (Implementação e Configuração em um Ambiente de Rede Controlada / Parâmetros Utilizados). Alguns experimentos também incluem a desconexão do cabo Ethernet. A Tabela 16 apresenta a divisão do Cenário 4 em quatro experimentos distintos.

Tabela 16 – Parâmetros utilizados nos experimentos 016 a 019 no cenário 4

Exp.	PCs	Pacote (MB)	Imagens Número	Tamanho da Imagem (MB)
016	7	100	50	2
017	7	100	50	2
018	7	100	50	2
019	7	100	50	2

Fonte: o autor

A Tabela 17 apresenta o mapa da rede para transmissão de pacotes entre os

oito computadores na configuração experimental. Ela descreve as interações entre pares de dispositivos, onde cada dispositivo funciona tanto como cliente quanto servidor em diferentes conexões. Por exemplo, o PC1 (*NodeProductor*) envia pacotes para o *NodeOrquestrator* (IP 10.0.0.14), enquanto o PC4 (*NodeProductor*) estabelece uma conexão com o PC1 (*NodeProductor*), demonstrando um modelo de comunicação bidirecional. Este mapa ajuda a compreender o fluxo de dados e a simulação de carga na rede em um ambiente controlado.

Tabela 17 – Mapa da rede de transmissão de pacotes entre cliente/servidor configurado com o uso de *iPerf3*

Cliente	Servidor	Cliente	Servidor
PC 1 (10.0.0.10)	Orquestrador (10.0.0.14)	Orquestrador (10.0.0.14)	PC 4 (10.0.0.16)
PC 2 (10.0.0.11)	PC 6 (10.0.0.15)	PC 6 (10.0.0.15)	PC 3 (10.0.0.12)
PC 3 (10.0.0.12)	Broker (10.0.0.6)	Broker (10.0.0.6)	PC 3 (10.0.0.11)
PC 4 (10.0.0.16)	PC 7 (10.0.0.13)	PC 7 (10.0.0.13)	PC 1 (10.0.0.10)

Fonte: o autor

A Tabela 18 apresenta os resultados dos experimentos 016 a 019, que abordam a transmissão, recepção, retransmissão de arquivos e restauração de imagens. No experimento 016, sete *NodeProductor* enviam imagens para o nó orquestrador, juntamente com pacotes adicionais para aumentar o tráfego na rede, sem interrupção do cabo Ethernet. No experimento 017, os sete *NodeProductor* enviam imagens para o *NodeOrquestrator*, com pacotes adicionais para intensificar o tráfego, mas o cabo Ethernet do *NodeOrquestrator* é desconectado por cinco minutos. O experimento 018 replica o experimento 017, porém com um intervalo de desconexão de dez minutos do cabo Ethernet do *NodeOrquestrator*. O experimento 019 repete as condições do experimento 016 para confirmar os dados coletados.

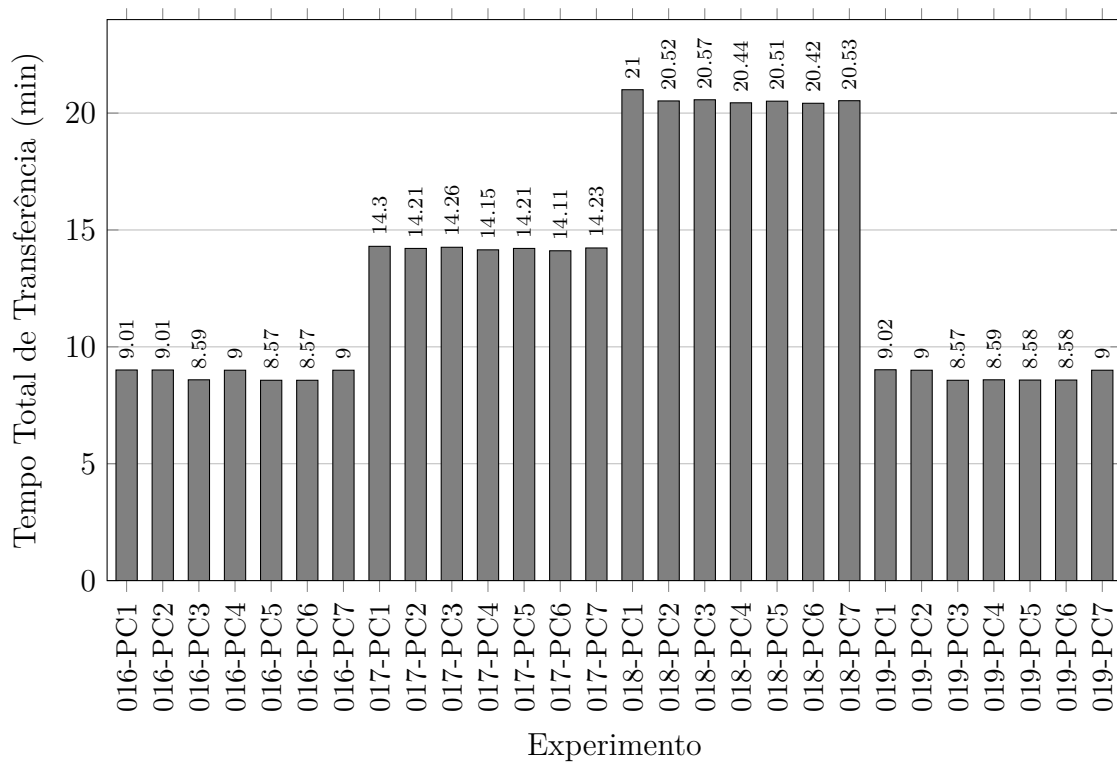
O gráfico na Figura 33 apresenta os tempos totais de transferência para os experimentos 016, 017, 018 e 019. Observa-se que, apesar da transmissão de pacotes e do congestionamento na rede, os tempos de transferência permaneceram dentro da média dos experimentos anteriores, exceto quando o cabo Ethernet foi desconectado, ocasião em que o tempo de transferência aumentou proporcionalmente à duração da desconexão. Mesmo no experimento 017, onde houve retransmissão de dados, o tempo de transferência manteve-se dentro da média.

Tabela 18 – Resultados da transmissão de arquivos e restauração de imagens nos experimentos 016 a 018 com desconexão

Exp.	Arquivos Enviados	Arquivos Recebidos	Arquivos Retrans.	Imagens Restauradas	Desconexão (min)
016	292	292	-	50	0
017	292	325	PC1 (33)	50	5
018	292	292	-	50	10
019	292	292	-	50	0

Fonte: o autor

Figura 33 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com interrupções de Internet e com injeção artificial de pacotes



Fonte: o autor.

Cenário 5

No Cenário 5, o *NodeHybrid* entra em ação. Neste ambiente experimental, ele solicita pacotes de imagens enviados pelos *NodeProductor* ao *NodeOrquestrator*. A Tabela 19 apresenta o fluxo de solicitações de imagens do *NodeHybrid* para o *NodeOrquestrator*.

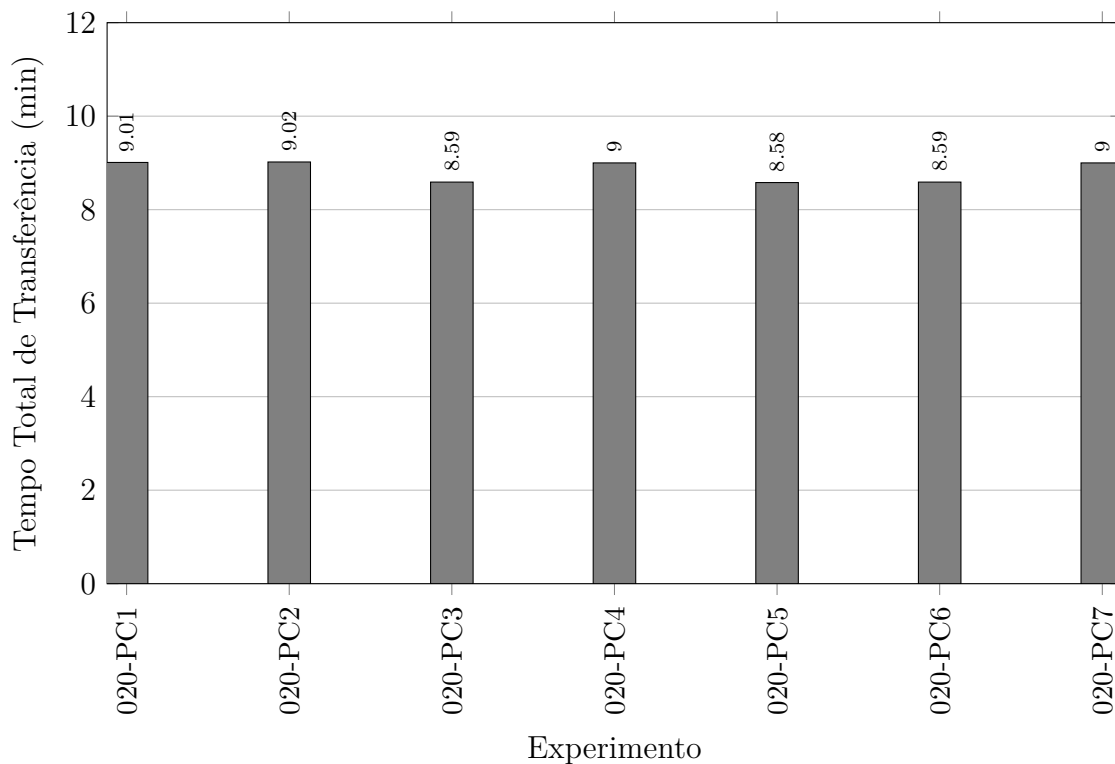
No experimento 020, todos os sete computadores são configurados como

Tabela 19 – Descrição do fluxo de solicitação de imagens com uso do *NodeHybrid* ao *NodeOrquestrator* no experimento 020

NodeHybrid Solicita	NodeOrquestrator Amostra	Arq. Env.	Arq. Rec.	Imag. Rest.
PC 1	PC 7	292	292	50
PC 2	PC 6	292	292	50
PC 3	PC 5	292	292	50
PC 4	PC 1	292	292	50
PC 5	PC 2	292	292	50
PC 6	PC 3	292	292	50
PC 7	PC 4	292	292	50

Fonte: o autor

NodeHybrid. Cada um solicita imagens que já foram enviadas e armazenadas no *NodeOrquestrator*, permitindo que qualquer *NodeProductor* acesse amostras enviadas por outros *NodeProductor* quando necessário.

Figura 34 – Análise comparativa dos tempos totais de transferência em ambiente de rede controlado com tráfego de imagens via MQTT, com solicitações de pacotes de imagens realizadas por *NodeHybrid* ao *NodeProductor*

Fonte: o autor.

A análise da Figura 34 mostra que o desempenho da aplicação, durante as

solicitações de imagens pelo *NodeHybrid* no ambiente controlado, mantém tempos médios comparáveis aos das transmissões de imagens dos *NodeProducer* para o *NodeOrchestrator*, com variações mínimas nos tempos de recebimento entre todos os *NodeHybrid* no experimento 020.

Os resultados evidenciam melhorias significativas na captura e transmissão de imagens em redes IoT baseadas no modelo Pub/Sub. A configuração óptica do PlanktoScope apresentou uma relação adequada entre detalhamento e precisão, enquanto os testes de transmissão indicaram maior eficiência com envio de partes de maior tamanho. O sistema demonstrou resiliência em cenários adversos, com estratégias que asseguraram a integridade dos dados, mesmo em condições de alta frequência de envio.

6 Considerações finais

O presente trabalho desenvolveu e avaliou um sistema para a transmissão confiável de imagens de microplásticos em alta resolução. O estudo focou em cenários com conectividade limitada, propondo uma estrutura para o monitoramento de microplásticos baseada em uma arquitetura de rede com agentes e nós distribuídos que compartilham dados e informações via protocolo MQTT e paradigma de comunicação Pub/Sub. A plataforma de baixo custo e *open-hardware* PlanktoScope foi adaptada para atender às especificidades do monitoramento, permitindo a análise de partículas a partir de 20 μm . O estudo incluiu uma discussão sobre melhores práticas, desafios e resultados da adaptação do sistema óptico. Além disso, foi desenvolvida uma solução robusta para a transmissão de imagens em redes IoT utilizando o protocolo MQTT. A abordagem demonstrou desempenho no compartilhamento de imagens de alta resolução, mesmo em condições adversas de conectividade, empregando um sistema baseado na arquitetura *Publish/Subscribe* [1] e [3].

A solução apresentada lida com a gestão de dados em escala permitindo seu compartilhamento entre nós distribuídos. A modularidade facilita a manutenção e a escalabilidade, assegurando que cada parte do processo seja executada de forma satisfatória, garantindo a integridade e a disponibilidade das imagens processadas [2].

Os agentes do sistema possuem funções específicas que contribuem para o tratamento adequado das imagens, desde a divisão e o envio das partes até a restauração e o armazenamento das imagens completas. A utilização de tópicos MQTT específicos para publicação e requisição de reenvio de imagens, proporciona uma gestão de tráfego adequada. Isso otimiza a largura de banda e garante que as imagens sejam entregues apenas aos inscritos que realmente necessitam delas, aumentando a segurança e a privacidade dos dados transmitidos [2].

Os testes realizados em ambiente controlado *testbed* permitiram analisar o impacto de variáveis como tamanho de arquivo, frequência de envio, e condições de interrupção na rede. Os resultados indicaram que o uso de arquivos maiores e técnicas de particionamento de dados otimizadas contribuem significativamente para a eficiência e a confiabilidade do sistema, reduzindo perdas e maximizando o desempenho [3] [4]. Além disso, foi comprovado que a abordagem garante a restauração integral das imagens enviadas, mesmo diante de desconexões momentâneas nos *NodeOrquestrator*. Por fim, este estudo contribui para o avanço tecnológico em sistemas de monitoramento baseados em redes IoT, oferecendo um *benchmark* para

auxiliar a comunidade a lidar com alguns dos principais desafios no monitoramento de microplásticos: o compartilhamento de dados e conhecimento entre pesquisadores de diferentes áreas geográficas [2].

6.1 Sugestões para trabalhos futuros

Para a continuidade do trabalho, as seguintes áreas podem ser desenvolvidas para aprimorar o sistema:

- Investigação do não reestabelecimento da comunicação nos experimentos 9 e 10 do cenário 2;
- Estudo dos impactos energéticos associados à transmissão de imagens em redes IoT que utilizam o modelo Pub/Sub, com foco em dispositivos de borda e no processamento local de dados;
- Desenvolvimento de uma funcionalidade que notifique os assinantes quando novas imagens forem postadas no tópico ao qual estão inscritos;
- Implementação de modelos de IA que atuem como *Publish/Subscriber* no *Hybrid Node*, recebendo mensagens para simulação, previsão e evolução de cenários por meio do sistema integrado.

Referências

- ABBAS, N. et al. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, IEEE, v. 5, n. 1, p. 450–465, 2017. Citado 2 vezes nas páginas 28 e 123.
- ABREHA, H. G.; HAYAJNEH, M.; SERHANI, M. A. Federated learning in edge computing: a systematic survey. *Sensors*, MDPI, v. 22, n. 2, p. 450, 2022. Citado 4 vezes nas páginas 110, 111, 115 e 116.
- ALAM, K. M.; AKRAM, A. A survey on mqtt protocol for the internet of things. *Khulna University, Dept. of Computer Science and Engineering (CSE)*, 2016. Citado 3 vezes nas páginas 18, 19 e 29.
- BAIOCCHI, O. et al. A performance evaluation of raspberry pi zero w based gateway running mqtt broker for iot. 2019. Citado na página 31.
- BARRICELLI, B. R.; CASIRAGHI, E.; FOGLI, D. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE access*, IEEE, v. 7, p. 167653–167671, 2019. Citado 2 vezes nas páginas 23 e 24.
- BAUER, P. et al. The digital revolution of earth-system science. *Nature Computational Science*, Nature Publishing Group, v. 1, n. 2, p. 104–113, 2021. Citado na página 23.
- BELLINI, P.; NESI, P.; PANTALEO, G. Iot-enabled smart cities: A review of concepts, frameworks and key technologies. *Applied Sciences*, MDPI, v. 12, n. 3, p. 1607, 2022. Citado 2 vezes nas páginas 30 e 33.
- BLACKBURN, K.; GREEN, D. The potential effects of microplastics on human health: What is known and what is unknown. *Ambio*, Springer, p. 1–13, 2021. Citado na página 23.
- BOTÍN-SANABRIA, D. M. et al. Digital twin technology challenges and applications: A comprehensive review. *Remote Sensing*, MDPI, v. 14, n. 6, p. 1335, 2022. Citado na página 29.
- BUSKEY, E. J.; HYATT, C. J. Use of the flowcam for semi-automated recognition and enumeration of red tide cells (*karenia brevis*) in natural plankton samples. *Harmful Algae*, Elsevier, v. 5, n. 6, p. 685–692, 2006. Citado na página 17.
- CAO, K. et al. An overview on edge computing research. *IEEE access*, IEEE, v. 8, p. 85714–85728, 2020. Citado 4 vezes nas páginas 19, 27, 28 e 119.
- CERA, A.; CESARINI, G.; SCALICI, M. Microplastics in freshwater: what is the news from the world? *Diversity*, MDPI, v. 12, n. 7, p. 276, 2020. Citado na página 21.
- CHATTERJEE, S.; SHARMA, S. Microplastics in our oceans and marine health. *Field Actions Science Reports. The Journal of Field Actions*, Institut Veolia, n. Special Issue 19, p. 54–61, 2019. Citado na página 23.

CHENG, Y. et al. Cyber-physical integration for moving digital factories forward towards smart manufacturing: a survey. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 97, n. 1, p. 1209–1221, 2018. Citado na página 23.

CONNAN, M. et al. The indian ocean ‘garbage patch’: Empirical evidence from floating macro-litter. *Marine Pollution Bulletin*, Elsevier, v. 169, p. 112559, 2021. Citado na página 17.

COSTA, B. et al. Orchestration in fog computing: A comprehensive survey. *ACM Computing Surveys (CSUR)*, ACM New York, NY, v. 55, n. 2, p. 1–34, 2022. Citado na página 27.

EGGER, M.; SULU-GAMBARI, F.; LEBRETON, L. First evidence of plastic fallout from the north pacific garbage patch. *Scientific reports*, Nature Publishing Group, v. 10, n. 1, p. 1–10, 2020. Citado na página 17.

EUROPE, P. *Plastics - the Facts 2021*. 2021. Available at <<https://plasticseurope.org/knowledge-hub/plastics-the-facts-2021/>> (2022/10/10). Citado na página 17.

FENOGLIO, E.; PITHADIA, H.; TRELEAVEN, P. Federated computing. *Available at SSRN*, 2023. Citado 7 vezes nas páginas 110, 112, 117, 120, 121, 123 e 124.

FERNANDEZ, G.; ZHAO, L.; WIJEGUNARATNE, I. Patterns for federated architecture. *Journal of Object Technology*, Citeseer, v. 2, n. 1, p. 135–149, 2003. Citado 5 vezes nas páginas 114, 115, 117, 118 e 123.

FU, G.; ZHANG, Y.; YU, G. A fair comparison of message queuing systems. *IEEE Access*, IEEE, v. 9, p. 421–432, 2020. Citado na página 28.

FULLER, A. et al. Digital twin: Enabling technologies, challenges and open research. *IEEE access*, IEEE, v. 8, p. 108952–108971, 2020. Citado na página 25.

GAMBI, E. et al. A home automation architecture based on lora technology and message queue telemetry transfer protocol. *International Journal of Distributed Sensor Networks*, SAGE Publications Sage UK: London, England, v. 14, n. 10, p. 1550147718806837, 2018. Citado na página 30.

GEYER, R.; JAMBECK, J. R.; LAW, K. L. Production, use, and fate of all plastics ever made. *Science advances*, American Association for the Advancement of Science, v. 3, n. 7, p. e1700782, 2017. Citado na página 17.

GIGAULT, J. et al. Current opinion: what is a nanoplastic? *Environmental pollution*, Elsevier, v. 235, p. 1030–1034, 2018. Citado na página 21.

GLAESSGEN, E.; STARGEL, D. The digital twin paradigm for future nasa and us air force vehicles. In: *53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA*. [S.l.: s.n.], 2012. p. 1818. Citado na página 24.

GOMATHI, R. M. et al. A survey on iot technologies, evolution and architecture. In: *IEEE. 2018 International Conference on Computer, Communication, and Signal Processing (ICCCSP)*. [S.l.], 2018. p. 1–5. Citado 2 vezes nas páginas 33 e 34.

- HALBA, K. *Toward a Framework for the Composition and Assessment of IoT and CPS Capabilities: Smart Cities Applications*. Tese (Doutorado) — Université Grenoble Alpes [2020-....], 2023. Citado 2 vezes nas páginas 43 e 46.
- HALE, R. C. et al. A global perspective on microplastics. *Journal of Geophysical Research: Oceans*, Wiley Online Library, v. 125, n. 1, p. e2018JC014719, 2020. Citado 2 vezes nas páginas 17 e 22.
- HEIMBIGNER, D.; MCLEOD, D. A federated architecture for information management. *ACM Transactions on Information Systems (TOIS)*, ACM New York, NY, USA, v. 3, n. 3, p. 253–278, 1985. Citado 6 vezes nas páginas 114, 117, 119, 120, 123 e 124.
- HOELLEIN, T. J. et al. Microplastic deposition velocity in streams follows patterns for naturally occurring allochthonous particles. *Scientific Reports*, Nature Publishing Group, v. 9, n. 1, p. 1–11, 2019. Citado na página 23.
- HORTON, A. A. et al. Microplastics in freshwater and terrestrial environments: evaluating the current understanding to identify the knowledge gaps and future research priorities. *Science of the total environment*, Elsevier, v. 586, p. 127–141, 2017. Citado na página 21.
- HUANG, Y. et al. A high-precision method for 100-day-old classification of chickens in edge computing scenarios based on federated computing. *Animals*, MDPI, v. 12, n. 24, p. 3450, 2022. Citado 3 vezes nas páginas 28, 111 e 121.
- HWANG, H. C.; PARK, J.; SHON, J. G. Design and implementation of a reliable message transmission system based on mqtt protocol in iot. *Wireless Personal Communications*, Springer, v. 91, p. 1765–1777, 2016. Citado 3 vezes nas páginas 18, 19 e 31.
- HWANG, H. C. et al. An enhanced reliable message transmission system based on mqtt protocol in iot environment. In: SPRINGER. *Advances in Computer Science and Ubiquitous Computing: CSA-CUTE2016 8*. [S.l.], 2017. p. 982–987. Citado na página 32.
- IMTEAJ, A. et al. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, IEEE, v. 9, n. 1, p. 1–24, 2021. Citado 2 vezes nas páginas 33 e 34.
- JAMIL, S.; RAHMAN, M.; FAWAD. A comprehensive survey of digital twins and federated learning for industrial internet of things (iiot), internet of vehicles (ioV) and internet of drones (iod). *Applied System Innovation*, MDPI, v. 5, n. 3, p. 56, 2022. Citado 2 vezes nas páginas 110 e 111.
- JIA, B. et al. Double-matching resource allocation strategy in fog computing networks based on cost efficiency. *Journal of Communications and Networks*, KICS, v. 20, n. 3, p. 237–246, 2018. Citado na página 27.
- KAILE, N. et al. Preliminary results from detection of microplastics in liquid samples using flow cytometry. *Frontiers in Marine Science*, Frontiers, p. 856, 2020. Citado 2 vezes nas páginas 17 e 19.

- KHAN, W. Z. et al. Edge computing: A survey. *Future Generation Computer Systems*, Elsevier, v. 97, p. 219–235, 2019. Citado 4 vezes nas páginas 27, 28, 31 e 123.
- KHANNA, A.; KAUR, S. Internet of things (iot), applications and challenges: a comprehensive review. *Wireless Personal Communications*, Springer, v. 114, p. 1687–1762, 2020. Citado 2 vezes nas páginas 33 e 34.
- KOELMANS, A. A. et al. Risk assessment of microplastic particles. *Nature Reviews Materials*, Nature Publishing Group, v. 7, n. 2, p. 138–152, 2022. Citado na página 22.
- KONEČNÝ, J. et al. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016. Citado 2 vezes nas páginas 110 e 121.
- KOOHANG, A. et al. Internet of things (iot): From awareness to continued use. *International Journal of Information Management*, Elsevier, v. 62, p. 102442, 2022. Citado 2 vezes nas páginas 33 e 34.
- LAGHARI, A. A. et al. A review and state of art of internet of things (iot). *Archives of Computational Methods in Engineering*, Springer, p. 1–19, 2021. Citado na página 33.
- LEO, M. et al. A federated architecture approach for internet of things security. In: IEEE. *2014 Euro Med Telco Conference (EMTC)*. [S.l.], 2014. p. 1–5. Citado 6 vezes nas páginas 18, 29, 30, 31, 114 e 115.
- LIMA, S.; ROCHA, A.; ROQUE, L. An overview of openstack architecture: a message queuing services node. *Cluster Computing*, Springer, v. 22, p. 7087–7098, 2019. Citado 3 vezes nas páginas 25, 29 e 31.
- LIU, Z.; MEYENDORF, N.; MRAD, N. The role of data fusion in predictive maintenance using digital twin. In: AIP PUBLISHING LLC. *AIP conference proceedings*. [S.l.], 2018. v. 1949, n. 1, p. 020023. Citado na página 23.
- LU, Y. et al. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks. *IEEE Internet of Things Journal*, IEEE, v. 8, n. 4, p. 2276–2288, 2020. Citado 2 vezes nas páginas 111 e 116.
- LUSHER, A. L. et al. Moving forward in microplastic research: a norwegian perspective. *Environment International*, Elsevier, v. 157, p. 106794, 2021. Citado na página 38.
- MADNI, A.; MADNI, C.; LUCERO, S. Leveraging digital twin technology in model-based systems engineering. *systems* 7 (1): 7. *Article-Number*, v. 7, 2019. Citado 2 vezes nas páginas 23 e 25.
- MERENDA, M.; PORCARO, C.; IERO, D. Edge machine learning for ai-enabled iot devices: A review. *Sensors*, MDPI, v. 20, n. 9, p. 2533, 2020. Citado na página 33.
- MÉRIGUET, Z. et al. Basin-scale underway quantitative survey of surface microplankton using affordable collection and imaging tools deployed from tara. *Frontiers in Marine Science*, Frontiers, p. 1125, 2022. Citado na página 37.

- MOSQUITTO, E. Eclipse mosquito. *Eclipse Mosquito*, 2018. Citado 3 vezes nas páginas 35, 37 e 64.
- MOURTZIS, D.; ANGELOPOULOS, J.; PANOPOULOS, N. A literature review of the challenges and opportunities of the transition from industry 4.0 to society 5.0. *Energies*, MDPI, v. 15, n. 17, p. 6276, 2022. Citado na página 25.
- NATIVI, S.; CRAGLIA, M. Destination earth. 2020. Citado na página 18.
- NATIVI, S.; MAZZETTI, P.; CRAGLIA, M. Digital ecosystems for developing digital twins of the earth: The destination earth case. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 13, n. 11, p. 2119, 2021. Citado 4 vezes nas páginas 26, 30, 42 e 45.
- NAZIR, S.; KALEEM, M. Reliable image notifications for smart home security with mqtt. In: IEEE. *2019 International Conference on Information Science and Communication Technology (ICISCT)*. [S.l.], 2019. p. 1–5. Citado 7 vezes nas páginas 18, 19, 29, 30, 31, 41 e 45.
- NGUYEN, D. C. et al. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, IEEE, v. 8, n. 16, p. 12806–12825, 2021. Citado 3 vezes nas páginas 111, 121 e 123.
- NUGRAHA, I. R. et al. A comparative study of http and mqtt for iot applications in hydroponics. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, v. 8, n. 1, p. 119–126, 2024. Citado na página 32.
- NYADJRO, E. S. et al. The noaa ncei marine microplastics database. *Scientific Data*, Nature Publishing Group UK London, v. 10, n. 1, p. 726, 2023. Citado na página 21.
- OBERMAISSER, R. et al. From a federated to an integrated automotive architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 28, n. 7, p. 956–965, 2009. Citado 3 vezes nas páginas 115, 117 e 123.
- OLSON, R. J.; SOSIK, H. M. A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging flowcytobot. *Limnology and Oceanography: Methods*, Wiley Online Library, v. 5, n. 6, p. 195–203, 2007. Citado na página 17.
- PAN, Z. et al. Ultra-stable phonon laser through closed-loop feedback control for optomechanical sensing. *Laser & Photonics Reviews*, Wiley Online Library, p. 2400593, 2024. Citado na página 18.
- PARK, C.; BALASUBRAMANIAN, A. A look into the performance of mmwave 5g, 3g, 4g/lte using pc application, iperf3 and dash. *Journal of Student Research*, v. 12, n. 1, 2023. Citado na página 72.
- PARK, Y. et al. Imaging flow cytometry protocols for examining phagocytosis of microplastics and bioparticles by immune cells of aquatic animals. *Frontiers in Immunology*, Frontiers, p. 203, 2020. Citado na página 17.
- PEREIRA, F. et al. Challenges in resource-constrained iot devices: Energy and communication as critical success factors for future iot deployment. *Sensors*, MDPI, v. 20, n. 22, p. 6420, 2020. Citado na página 34.

- PIAS, M. et al. *On the Scaling of Digital Twins by Aggregation*. 2023. 1-13 p. Citado 2 vezes nas páginas 25 e 26.
- PLANKTOSCOPE.ORG. *What is a PlanktoScope?* 2020. Disponível em: <<https://www.planktoscope.org/>>. Acesso em: 15 Mai. 2022. Citado na página 38.
- POLLINA, T. et al. Planktoscope: affordable modular imaging platform for citizen oceanography. *BioRxiv*, Cold Spring Harbor Laboratory, 2020. Citado 2 vezes nas páginas 17 e 37.
- RAMU, S. P. et al. Federated learning enabled digital twins for smart cities: Concepts, recent advances, and future directions. *Sustainable Cities and Society*, Elsevier, v. 79, p. 103663, 2022. Citado 2 vezes nas páginas 118 e 119.
- RÍOS, J. et al. Product avatar as digital counterpart of a physical individual product: Literature review and implications in an aircraft. *Transdisciplinary Lifecycle Analysis of Systems*, IOS Press, p. 657–666, 2015. Citado na página 24.
- SÁ, L. C. D. et al. Studies of the effects of microplastics on aquatic organisms: what do we know and where should we focus our efforts in the future? *Science of the total environment*, Elsevier, v. 645, p. 1029–1039, 2018. Citado na página 21.
- SADEQ, A. S. et al. A qos approach for internet of things (iot) environment using mqtt protocol. In: IEEE. *2019 International Conference on Cybersecurity (ICoCSec)*. [S.l.], 2019. p. 59–63. Citado 2 vezes nas páginas 40 e 45.
- SATYANARAYANAN, M. The emergence of edge computing. *Computer*, IEEE, v. 50, n. 1, p. 30–39, 2017. Citado 2 vezes nas páginas 27 e 123.
- SCHLUSE, M.; ROSSMANN, J. From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems. In: IEEE. *2016 IEEE International Symposium on Systems Engineering (ISSE)*. [S.l.], 2016. p. 1–6. Citado na página 25.
- SHIM, W. J.; HONG, S. H.; EO, S. E. Identification methods in microplastic analysis: a review. *Analytical methods*, Royal Society of Chemistry, v. 9, n. 9, p. 1384–1391, 2017. Citado na página 37.
- SHIM, W. J.; THOMPSON, R. C. Microplastics in the ocean. *Archives of environmental contamination and toxicology*, Springer, v. 69, n. 3, p. 265–268, 2015. Citado na página 23.
- SHUVO, M. M. H. et al. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, IEEE, v. 111, n. 1, p. 42–91, 2022. Citado 2 vezes nas páginas 120 e 123.
- SILVA, A. C. F. D. et al. Internet of things out of the box: using toasca for automating the deployment of iot environments. In: SCITEPRESS. *International Conference on Cloud Computing and Services Science*. [S.l.], 2017. v. 2, p. 358–367. Citado na página 35.
- SOCIETY, T. R. Digital technology and the planet: Harnessing computing to achieve net zero. The Royal Society, 2020. Citado 6 vezes nas páginas 18, 19, 23, 24, 28 e 124.

- SOLANKI, V. K.; DHALL, R. An iot based predictive connected car maintenance approach. *International Journal of Interactive Multimedia and Artificial Intelligence* . . . , 2017. Citado na página 35.
- SONI, D.; MAKWANA, A. A survey on mqtt: a protocol of internet of things (iot). In: *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*. [S.l.: s.n.], 2017. v. 20, p. 173–177. Citado na página 31.
- SUN, W. et al. Adaptive federated learning and digital twin for industrial internet of things. *IEEE Transactions on Industrial Informatics*, IEEE, v. 17, n. 8, p. 5605–5614, 2020. Citado 5 vezes nas páginas 110, 111, 116, 119 e 121.
- SUN, W. et al. Dynamic digital twin and federated learning with incentives for air-ground networks. *IEEE Transactions on Network Science and Engineering*, IEEE, v. 9, n. 1, p. 321–333, 2020. Citado 2 vezes nas páginas 118 e 119.
- TEIXEIRA, R. et al. Planetary digital twin: a case study in aquaculture. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. [S.l.: s.n.], 2022. p. 191–197. Citado 2 vezes nas páginas 18 e 26.
- TONG, Y. et al. Federated computing: Query, learning, and beyond. *IEEE Data Eng. Bull.*, v. 46, n. 1, p. 9–26, 2023. Citado 6 vezes nas páginas 112, 118, 119, 120, 122 e 125.
- TSUKADA, M.; KONDO, M.; MATSUTANI, H. A neural network-based on-device learning anomaly detector for edge devices. *IEEE Transactions on Computers*, IEEE, v. 69, n. 7, p. 1027–1044, 2020. Citado na página 120.
- TURINA, V. et al. Combining split and federated architectures for efficiency and privacy in deep learning. In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*. [S.l.: s.n.], 2020. p. 562–563. Citado na página 115.
- VARGHESE, B. et al. Challenges and opportunities in edge computing. In: *IEEE. 2016 IEEE international conference on smart cloud (SmartCloud)*. [S.l.], 2016. p. 20–26. Citado 3 vezes nas páginas 27, 28 e 123.
- WANG, S. et al. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, IEEE, v. 37, n. 6, p. 1205–1221, 2019. Citado na página 110.
- WANG, X.-S. et al. Quantitatively analyzing the variation of micrometer-sized microplastic during water treatment with the flow cytometry-fluorescent beads method. *ACS ES&T Engineering*, ACS Publications, v. 1, n. 12, p. 1668–1677, 2021. Citado na página 17.
- WARD, C. P. et al. Sunlight converts polystyrene to carbon dioxide and dissolved organic carbon. *Environmental science & technology letters*, ACS Publications, v. 6, n. 11, p. 669–674, 2019. Citado na página 22.

XIA, Q. et al. A survey of federated learning for edge computing: Research problems and solutions. *High-Confidence Computing*, Elsevier, v. 1, n. 1, p. 100008, 2021. Citado 2 vezes nas páginas 112 e 123.

YASSEIN, M. B. et al. Internet of things: Survey and open issues of mqtt protocol. In: IEEE. *2017 international conference on engineering & MIS (ICEMIS)*. [S.l.], 2017. p. 1–6. Citado 6 vezes nas páginas 18, 19, 31, 32, 43 e 45.

YE, Y. et al. Edgefed: Optimized federated learning based on edge computing. *IEEE Access*, IEEE, v. 8, p. 209191–209198, 2020. Citado 4 vezes nas páginas 116, 117, 121 e 123.

YELAMARTHI, K.; AMAN, M. S.; ABDELGAWAD, A. An application-driven modular iot architecture. *Wireless Communications and Mobile Computing*, Wiley Online Library, v. 2017, n. 1, p. 1350929, 2017. Citado 2 vezes nas páginas 30 e 33.

YOKOGAWA. *FlowCam*. 2023. <<https://www.fluidimaging.com/>> (2022/10/10). Citado na página 17.

YOKOTANI, T.; SASAKI, Y. Comparison with http and mqtt on required network resources for iot. In: IEEE. *2016 international conference on control, electronics, renewable energy and communications (ICCEREC)*. [S.l.], 2016. p. 1–6. Citado 2 vezes nas páginas 31 e 32.

ZHANG, C.; LIU, X. Hbasemq: a distributed message queuing system on clouds with hbase. In: IEEE. *2013 Proceedings IEEE INFOCOM*. [S.l.], 2013. p. 40–44. Citado na página 28.

ZHENG, Y.; YANG, S.; CHENG, H. An application framework of digital twin and its case study. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 10, n. 3, p. 1141–1153, 2019. Citado na página 23.

APÊNDICE A – Computação Federada

A Computação Federada, ou "*Federated Computing*", refere-se à distribuição e execução de processos computacionais em uma rede de sistemas independentes, ou que tenham dispositivos que não necessariamente compartilham uma infraestrutura comum [Wang et al. 2019]. Onde diferentes sistemas ou organizações colaboram e compartilham recursos computacionais, mantendo ao mesmo tempo a autonomia sobre seus próprios dados e infraestrutura. Esse modelo de computação é baseado na ideia de uma "federação" [Abreha, Hayajneh e Serhani 2022]. Otimiza a transferência de dados dentro de *frameworks* de computação descentralizada, reduzindo a necessidade de grandes transferências de dados. Isso não apenas economiza largura de banda da rede, mas também minimiza potenciais riscos de segurança associados ao movimento de dados [Fenoglio, Pithadia e Treleaven 2023].

É abordada principalmente por meio da lente do aprendizado federado, que é uma forma de computação distribuída onde vários dispositivos ou nós colaboram para realizar o treinamento de modelos de aprendizado de máquina sem compartilhar diretamente seus dados. Portanto, a computação federada, refere-se a essa capacidade de realizar computação distribuída de maneira adequada e privada em um ambiente como o da IoT, utilizando aprendizado federado como a metodologia central [Sun et al. 2020].

Segundo [Jamil, Rahman e Fawad 2022] é um sistema de ciência de dados destinado a dados dispersos e, portanto, não centralizados, permitindo que empresas utilizem seus dados conjuntamente para treinar modelos de aprendizado de máquina de forma colaborativa sem a necessidade de compartilhar ou centralizar esses dados explicitamente.

Surge como uma resposta à necessidade de treinar modelos de alta qualidade em dados distribuídos de maneira desigual por um grande número de nós, minimizando a quantidade de comunicação necessária. Esta abordagem é particularmente relevante quando os dados de treinamento são mantidos localmente nos dispositivos móveis dos usuários, em vez de serem centralizados em um *data center*, com o objetivo de realizar computações nos dados locais para atualizar um modelo global [Konečný et al. 2016].

A.1 Aplicações

A computação federada está sendo aplicada em diversas áreas, incluindo saúde, cidades inteligentes, redes de próxima geração, IoT, indústria automotiva, prognósticos e gestão de saúde, e Indústria 4.0 como sistemas ciber-físicos. Especificamente, há menções à sua aplicação na *Internet of Vehicles* (IoV) e *Internet of Drones* (IoD) [Jamil, Rahman e Fawad 2022].

Na área da IIoT, aparece com foco em melhorar a eficiência e a privacidade do aprendizado de máquina em ambientes industriais distribuídos [Sun et al. 2020]. Ainda dentro do contexto da IIoT a aplicação para [Lu et al. 2020], esta focada na melhoria da eficiência da comunicação em redes de borda de DT. Este método permite construir modelos de DT a partir dos dados operacionais dos dispositivos IoT, sem a necessidade de transmitir os dados brutos, o que melhora a privacidade dos dados e reduz o custo de comunicação, utilizando para isso o aprendizado federado.

Na área de redes de borda "*edge networks*", em cenários de computação de borda em redes IoT, para melhorar a segurança e eficiência, integrando tecnologias de *blockchain* e aprendizado federado "*federated learning*" para criar uma arquitetura segura que respeita a privacidade dos dados em computação distribuída de borda [Lu et al. 2020]. Em [Nguyen et al. 2021] é aplicada como uma solução colaborativa para treinamento de dados, coordenando múltiplos dispositivos móveis para treinar um modelo de IA compartilhado sem expor diretamente seus dados subjacentes. Ainda na área da computação de borda em [Huang et al. 2022] é aplicada na área de classificação de idade de frangos. Implementado em dispositivos com capacidades computacionais limitadas, comum nestes cenários, e visa melhorar a precisão do reconhecimento da idade dos frangos, o que é crucial para a alimentação de precisão e gestão na agricultura inteligente.

No contexto da Indústria 4.0, a computação federada é aplicada combinando ativos de dados distribuídos, aprendizado de máquina federado (FML), e tecnologia de ledger distribuído (DLT) para garantir privacidade de dados, análises descentralizadas e segurança. Esta abordagem é importante para a gestão de dados e colaboração em várias configurações, incluindo organizações e entre robôs e máquinas "inteligentes" colaborativas, demonstrando a importância da computação federada na promoção da quarta revolução industrial [Abreha, Hayaajneh e Serhani 2022]

A.2 Vantagens e Desvantagens da Computação Federada

A computação federada também enfrenta desafios consideráveis, incluindo a complexidade na gestão de recursos heterogêneos, a necessidade de mecanismos

de incentivo para a participação dos usuários, questões de latência e eficiência na comunicação, e a dificuldade em garantir a segurança e a confiabilidade em redes descentralizadas. À medida que exploramos o potencial dessa tecnologia disruptiva, é crucial abordar essas vantagens e desvantagens para maximizar seu impacto positivo em diversos setores.

- **Vantagens da Computação Federada**

1. **Preservação da Privacidade:** A computação federada permite que os dados permaneçam localizados, preservando a privacidade dos dados ao evitar a necessidade de transferi-los para um servidor central [Fenoglio, Pithadia e Treleaven 2023];
2. **Redução do Tempo de Treinamento:** Utiliza múltiplos dispositivos para calcular gradientes em paralelo, acelerando significativamente o processo de treinamento [Xia et al. 2021];
3. **Facilitação da Aprendizagem Colaborativa:** A aprendizagem federada permite uma espécie de "*crowdsourcing*" para coleta e rotulação de dados, simplificando esses processos em termos de tempo e esforço [Xia et al. 2021];
4. **Eficiência de Recursos:** Otimiza a alocação de recursos em redes descentralizadas, promovendo escalabilidade e custo-efetividade [Fenoglio, Pithadia e Treleaven 2023].

- **Desvantagens da Computação Federada**

1. **Desafios de Privacidade:** Apesar de melhorar a privacidade, a computação federada introduz novas restrições de privacidade, como a necessidade de manter os dados brutos localmente, o que pode complicar a computação colaborativa [Tong et al. 2023];
2. **Desafios de Eficiência:** As restrições de privacidade e segurança frequentemente introduzem comunicação e computação extras, sendo importante cumprir essas restrições enquanto se permite uma computação de alta eficiência [Tong et al. 2023];
3. **Desafios de Eficácia:** A heterogeneidade dos dados entre os proprietários de dados pode afetar a precisão da análise, especialmente quando os dados não são independentes e identicamente distribuídos (não-IID) [Tong et al. 2023].

A Tabela 20 apresenta um comparativo detalhado das principais vantagens e desvantagens da Computação Federada, evidenciando como este, pode otimizar

processos e gestão de dados enquanto enfrenta desafios significativos em termos de eficácia e gerenciamento de recursos. Esta tabela tem como objetivo oferecer uma visão clara dos efeitos e aspectos importantes relacionados ao uso desta tecnologia em situações reais e práticas.

Tabela 20 – Vantagens e Desvantagens da Computação Federada

Categoria	Vantagens	Desvantagens
Privacidade de Dados	Mantém dados localizados, evitando transferências desnecessárias e protegendo a privacidade dos usuários.	Introduz vulnerabilidades de segurança ao exigir manutenção local dos dados brutos.
Eficiência do Processo	Acelera o aprendizado de máquina por meio do uso paralelo da capacidade computacional de vários dispositivos.	Aumenta o consumo de recursos de comunicação e computacionais devido a medidas de segurança e privacidade.
Tempo de Resposta	Permite respostas rápidas e independentes ao distribuir modelos por vários dispositivos.	Compromete a precisão dos modelos devido à variação nos conjuntos de dados entre participantes (dados não-IID).
Colaboração e Escalabilidade	Economiza tempo e recursos por meio da coleta e análise colaborativa e distribuída de dados.	Desafios na integração e sincronização entre diferentes plataformas e dispositivos.
Alocação de Recursos	Melhora a escalabilidade e reduz custos operacionais ao maximizar o uso de recursos em rede descentralizada.	Necessita adaptações frequentes para otimizar a alocação de recursos à medida que novos dispositivos são adicionados.

Fonte: O autor

Concluindo, a Computação Federada é uma tecnologia promissora que permite o gerenciamento e a análise de dados em ambientes descentralizados, valorizando a privacidade e a colaboração. Apesar dos desafios de eficácia e gerenciamento de recursos, suas várias aplicações mostram seu potencial de transformação. À medida que a tecnologia evolui, será importante continuar desenvolvendo e melhorando esses sistemas para aproveitar ao máximo seus benefícios em um mundo cada vez mais baseado em dados.

APÊNDICE B – Arquitetura Federada

A arquitetura federada é um método que facilita o compartilhamento e a troca de informações entre bancos de dados independentes, permitindo que eles colaborem de forma controlada e parcial. Essa abordagem reúne vários sistemas de banco de dados numa estrutura chamada federação, que é flexível em sua formação. Cada sistema dentro dessa federação mantém o controle sobre o que compartilha e o que importa de outros sistemas por meio de esquemas de exportação e importação. Além disso, um componente central, conhecido como dicionário federal, ajuda a organizar a estrutura da federação e monitorar a inclusão de novos sistemas. Essa arquitetura não só permite o compartilhamento de dados e transações, mas também a combinação de informações de diferentes fontes e a coordenação entre elas [Heimbigner e McLeod 1985].

A arquitetura federada refere-se a um modelo de design que permite a troca de dados e serviços em cenários de Internet das Coisas, focando na implantação e gerenciamento de um ambiente federado para mecanismos de delegação de autoridade, capacidade baseada em identidade e informações de contexto dinâmico. Este modelo é projetado para gerenciar a comunicação segura, tanto para nós intraSMGW quanto para interSMGW, utilizando uma abordagem de Arquitetura Orientada a Serviços (SOA) baseada em Serviços *Web*, suportando os princípios SOAP e REST de acordo com as características e natureza dos dispositivos e serviços [Leo et al. 2014].

Arquitetura Federada é uma forma arquitetônica distribuída e orientada a eventos, projetada para espelhar a estrutura de uma organização, agrupando módulos e aplicações altamente interdependentes em domínios majoritariamente independentes. Essa arquitetura visa refletir os padrões de trabalho de grupos autônomos dentro de uma organização, onde cada domínio é isolado administrativa e processualmente de outros domínios, possuindo todas as capacidades necessárias para suportar suas próprias aplicações. A separação de processamento e administração dos domínios ajuda a reduzir a complexidade das conexões interdomínios. Um módulo de software situado na fronteira de cada domínio atua como uma interface ou facilitador, gerenciando e coordenando o tráfego de informações do domínio com o restante da federação. Por meio desta interface, cada domínio de aplicação é conectado a uma via federal orientada a mensagens, atuando como veículo para comunicação interdomínios e garantindo a entrega de mensagens [Fernandez, Zhao e Wijegunaratne 2003].

Uma arquitetura federada é aquela em que cada função principal é implementada em uma unidade de hardware dedicada. Esta abordagem de hardware federado

tem vantagens notáveis do ponto de vista de gestão de complexidade, isolamento de falhas e contenção de erros. Em uma arquitetura federada, as unidades de contenção de falhas, ou seja, as Unidades de Controle Eletrônico (ECUs), são claramente definidas, dificultando que dependências implícitas causem a propagação de erros de uma ECU defeituosa para outra ECU fisicamente distante que não está diretamente afetada pelo defeito [Obermaisser et al. 2009].

Aplicada ao aprendizado de máquina, é uma técnica de treinamento de redes neurais onde os dados permanecem privados e não são compartilhados entre os participantes. Neste modelo, o algoritmo de aprendizado de máquina é enviado até os dados, que estão distribuídos entre diversos participantes, cada um com seu próprio conjunto de dados. Durante o treinamento, os modelos locais são ajustados separadamente e apenas as atualizações de pesos das redes neurais são compartilhadas entre os participantes. Isso permite que a rede neural seja treinada sem a necessidade de expor os dados originais, protegendo a privacidade dos dados envolvidos. [Turina et al. 2020].

A Arquitetura Federada, como descrita em diversos estudos, apresenta uma estrutura que reflete a maneira como unidades ou departamentos organizacionais interagem entre si. Segundo [Fernandez, Zhao e Wijegunaratne 2003], esta arquitetura baseia-se em comunicação interdomínio por um mecanismo de publicar-assinar, visando espelhar a estrutura organizacional e permitindo a troca de informações. Ela suporta melhor a computação distribuída para aplicações novas e legadas, fornecendo uma infraestrutura integrada para a organização.

[Leo et al. 2014] expande esse conceito ao detalhar como a Arquitetura Federada facilita o compartilhamento e a troca controlados de informações entre bancos de dados autônomos. Descreve como um conjunto de sistemas de banco de dados independentes pode formar uma federação pouco acoplada, onde cada componente mantém controle sobre suas interações com outros por meio de esquemas de exportação e importação.

B.1 Aplicações

No setor automotivo, [Obermaisser et al. 2009] ilustra a aplicação da Arquitetura Federada, onde cada função principal é implementada em uma unidade de hardware dedicada. Esta abordagem oferece vantagens significativas em termos de gestão de complexidade, isolamento de falhas e contenção de erros, com unidades claramente definidas para contenção de falhas.

Em [Abreha, Hayajneh e Serhani 2022], é descrita a arquitetura de aprendizado

federado tanto nos dispositivos participantes quanto no servidor de aprendizado federado, detalhando os componentes essenciais que compõem essas arquiteturas. Assim, a descrição em [Abreha, Hayajneh e Serhani 2022] oferece uma visão mais técnica de como o aprendizado federado pode ser efetivamente implementado para maximizar tanto o desempenho quanto a segurança, respeitando a privacidade dos dados dos usuários.

Já [Sun et al. 2020] menciona a colaboração de dispositivos de maquinário baseada em aprendizado federado para tomada de decisões inteligentes, o que implica a aplicação de uma arquitetura federada para a inicialização de tarefas e atualização de modelos locais. Essa abordagem não só melhora a privacidade e a segurança dos dados, mas também otimiza a utilização de recursos ao reduzir a necessidade de transferência de grandes volumes de dados entre os dispositivos e o servidor central.

[Lu et al. 2020] propõe a arquitetura de Redes de Borda de Gêmeos Digitais (DITEN), integrando gêmeos digitais com computação de borda e adotando aprendizado federado para construir modelos em DITEN, o que é uma aplicação direta de arquitetura federada. Portanto esta estratégia não só melhora a escalabilidade e a eficiência dos sistemas distribuídos, mas também reforça a capacidade de cada nó em tomar decisões baseadas em dados atualizados e contextuais, potencializando a aplicação prática da arquitetura federada na construção de soluções de inteligência artificial distribuída.

[Ye et al. 2020] discute uma arquitetura hierárquica de aprendizado federado otimizada, que reduz a latência de comunicação e o consumo de energia dos dispositivos finais, mostrando outra aplicação de arquitetura federada. Esta técnica mostra a versatilidade da arquitetura federada ao adaptá-la para melhorar o desempenho e a eficiência em cenários de aprendizado de máquina distribuído, destacando seu potencial em aplicativos modernos de tecnologia.

B.2 Vantagens e Desvantagens da Arquitetura Federada

A arquitetura federada oferece várias vantagens para organizações, como melhorar a segurança dos dados, otimizar o uso de recursos e adaptar-se bem à estrutura organizacional existente. Contudo, essa abordagem também traz desafios, como a complexidade na implementação, potenciais problemas de comunicação e o risco de causar fragmentação dentro da organização.

- **Vantagens da Arquitetura Federada**

1. **Privacidade e Segurança dos Dados:** Assegura que os dados permaneçam locais, evitando riscos de vazamentos em larga escala e aumentando a

conformidade com regulamentações de privacidade [Heimbigner e McLeod 1985, Fenoglio, Pithadia e Treleaven 2023];

2. **Eficiência na Utilização de Recursos:** Otimiza a utilização de recursos computacionais ao distribuir processamentos mais próximos aos usuários finais, reduzindo latências e custos operacionais [Heimbigner e McLeod 1985, Ye et al. 2020];
3. **Descentralização:** Facilita a colaboração inter organizacional sem a necessidade de compartilhar dados diretamente, promovendo a inovação e a integração de novas tecnologias em ambientes descentralizados [Heimbigner e McLeod 1985, Fenoglio, Pithadia e Treleaven 2023];
4. **Reflexo da Estrutura Organizacional:** A arquitetura espelha as interações entre departamentos, permitindo uma melhor integração de software que alinha as tecnologias às necessidades de cada unidade [Fernandez, Zhao e Wijegunaratne 2003];
5. **Autonomia dos Domínios:** Promove módulos autônomos dentro da organização, simplificando a gestão de complexidade e reduzindo as interdependências diretas entre diferentes áreas [Heimbigner e McLeod 1985];
6. **Gestão de Complexidade e Isolamento de Falhas:** Em setores como o automotivo, separa funções críticas em unidades de hardware específicas, reduzindo o impacto de falhas e melhorando a robustez geral do sistema [Obermaisser et al. 2009].

- **Desvantagens da Arquitetura Federada**

1. **Complexidade na Gestão de Recursos:** A coordenação de diversos dispositivos e sistemas pode adicionar uma camada de complexidade na gestão de recursos [Heimbigner e McLeod 1985, Ye et al. 2020];
2. **Desafios de Comunicação e Latência:** Ainda que a arquitetura busque reduzir a latência, desafios de comunicação podem persistir, especialmente em redes heterogêneas [Heimbigner e McLeod 1985, Ye et al. 2020];
3. **Complexidade de Implementação:** Refletir a estrutura organizacional na tecnologia pode levar a complexidades significativas tanto na fase de implementação quanto de manutenção [Obermaisser et al. 2009];
4. **Desafios na Integração:** A autonomia dos domínios, apesar de útil, pode dificultar a integração de sistemas e a comunicação entre diferentes áreas da organização [Heimbigner e McLeod 1985];

5. **Risco de Fragmentação:** A ênfase na modularidade e autonomia pode resultar em uma fragmentação excessiva, complicando a coordenação e a visão estratégica organizacional [Fernandez, Zhao e Wijegunaratne 2003];
6. **Heterogeneidade e Escalabilidade:** A diversidade de dispositivos e a necessidade de escalabilidade em ambientes federados podem complicar a implementação e a manutenção de sistemas federados, exigindo soluções robustas para gerenciar a heterogeneidade [Tong et al. 2023].

A Tabela 21 apresenta as vantagens e desvantagens da arquitetura federada de forma comparativa. Este método é cada vez mais usado para aumentar a privacidade e a segurança dos dados, melhorar o uso dos recursos e ajudar na colaboração entre sistemas de forma descentralizada. No entanto, a arquitetura federada enfrenta alguns desafios, como a complexidade na gestão de recursos e problemas na comunicação e integração entre sistemas. A tabela fornece um resumo claro desses pontos, facilitando o entendimento rápido dos benefícios e dificuldades dessa abordagem.

Tabela 21 – Vantagens e Desvantagens da Arquitetura Federada

Categoria	Vantagens	Desvantagens
Privacidade e Segurança dos Dados	Protege a privacidade, mantendo os dados locais.	Aumenta a complexidade da segurança local.
Eficiência na Utilização de Recursos	Otimiza recursos e reduz latências.	Complexidade na gestão de múltiplos dispositivos.
Descentralização	Facilita colaborações sem compartilhamento direto de dados.	Desafios em comunicação e integração.
Reflexo da Estrutura Organizacional	Alinha tecnologia com as necessidades departamentais.	Desafiador espelhar a estrutura organizacional.
Autonomia dos Domínios	Promove independência e reduz interdependências.	Risco de fragmentação e dificuldade em coordenação.

Fonte: Seção B.2

Resumidamente, a arquitetura federada é um modelo distribuído no qual várias partes independentes colaboram para alcançar um objetivo comum, compartilhando informações e recursos de maneira descentralizada [Sun et al. 2020, Ramu et al. 2022]. Trata-se de um sistema distribuído no qual múltiplos componentes ou entidades autônomas colaboram para formar uma infraestrutura ou sistema unificado. Cada componente do sistema mantém o controle local de seus dados e recursos, ao mesmo tempo em que coopera com os outros para alcançar um objetivo comum. Dessa forma,

eles podem manter sua autonomia e gerenciar suas próprias políticas e decisões internas [Sun et al. 2020, Ramu et al. 2022].

B.3 Entidades Federadas

As "Entidades Federadas" são componentes que trabalham juntos e compartilham recursos em um sistema espalhado, e que não centralizam controle ou dados. Esse método é usado quando diferentes organizações ou sistemas querem manter seu controle próprio, mas precisam colaborar para atingir metas comuns.

Na literatura, as entidades federadas não são muito citadas, é mais comum a discussão dos aspectos técnicos, desafios e aplicações da computação federada e do aprendizado federado em cenários como Edge Computing, privacidade de dados e colaboração entre dispositivos ou partes com dados distribuídos.

Por exemplo, [Tong et al. 2023] discute a estrutura geral da computação federada, onde várias tarefas de computação são realizadas localmente pelos proprietários dos dados, seguidas por uma computação segura para derivar o resultado final de maneira preservando a privacidade. Este conceito se alinha com a ideia de cooperação entre várias partes mantendo a autonomia sobre seus dados e recursos.

Da mesma forma, [Sun et al. 2020] menciona a colaboração de dispositivos de maquinário baseada em aprendizado federado para tomada de decisões inteligentes, o que ilustra a aplicabilidade da arquitetura federada em cenários industriais e de IoT.

Em [Heimbigner e McLeod 1985] é mencionado que uma arquitetura federada de banco de dados que é composta por um conjunto de sistemas de banco de dados independentes que se unem em uma federação acoplada para compartilhar e trocar informações. Cada componente nesta federação controla suas interações com outros componentes por meio de esquemas de exportação e importação, mantendo sua autonomia enquanto permite um grau de compartilhamento e troca de informações. Além disso, enfatiza-se a importância da cooperação entre as entidades federadas para o funcionamento correto da federação, onde as atividades cooperativas são essenciais para o compartilhamento negociado de dados.

Outra situação é em [Cao et al. 2020], onde a arquitetura de Edge Computing é descrita como uma estrutura de rede federada. Esta estrutura estende os serviços de nuvem até a borda da rede, introduzindo dispositivos de borda entre dispositivos terminais e a computação em nuvem. A colaboração entre nuvem e borda é dividida em camada terminal, camada de borda e camada de computação em nuvem, destacando a importância da federação entre essas camadas para a eficácia da arquitetura de Edge Computing.

[Shuvo et al. 2022] menciona o aprendizado federado como uma solução para treinamento descentralizado em dispositivos de borda, colaborativamente treinando um modelo global usando dados locais sem a necessidade de centralizar os dados. Isso é particularmente relevante para tarefas de reconhecimento de atividades e padrões

O uso de um *framework* de aprendizado federado para treinar um modelo global em dispositivos de usuário, onde cada dispositivo executa treinamento apenas com seus dados locais é discutido em [Tsukada, Kondo e Matsutani 2020]. Embora o objetivo seja criar um modelo personalizado localmente para o dispositivo de borda alvo, a concepção de utilizar dispositivos como nós computacionais para treinamento colaborativo se alinha ao conceito de entidades federadas

Portanto, uma entidade federada é uma base de dados ou um sistema autônomo que faz parte de uma federação, onde compartilha informações por meio de interfaces explícitas sem a existência de uma autoridade central ou um esquema global. Neste contexto, cada entidade federada mantém sua autonomia, controlando quais dados deseja compartilhar com outras entidades federadas e quais informações de outras entidades deseja acessar e manipular. As modificações e o compartilhamento de dados são controlados pelo proprietário dos dados, mas um mecanismo de negociação garante que as mudanças na estrutura dos dados ocorram de maneira ordenada [Heimbigner e McLeod 1985].

Esta abordagem equilibra a necessidade de manter a autonomia dos componentes com a capacidade de alcançar um grau razoável de compartilhamento de informações, sem a imposição de atividades por uma autoridade central. Em vez disso, promove a atividade cooperativa entre os mesmos. Destacando a autonomia dos componentes federados e a necessidade de mecanismos de cooperação e negociação para o compartilhamento e troca de informações [Fenoglio, Pithadia e Treleaven 2023].

As entidades federadas são descritas em vários contextos e incluem dispositivos, partes e entidades que podem ser consideradas como componentes de uma federação em diferentes domínios, principalmente em computação federada e aprendizado federado. Aqui está a lista de dispositivos, partes e entidades mencionadas na literatura como exemplos de entidades federadas:

- **Proprietários de Dados Locais:** No contexto de computação federada, os proprietários de dados locais trabalham juntos para computar resultados parciais em seus próprios *datasets* e, em seguida, colaboram para derivar o resultado final de maneira segura e preservando a privacidade [Tong et al. 2023];
- **Bancos e Supermercados em Diferentes Regiões:** No cenário de apren-

dizado federado de transferência, bancos e supermercados podem colaborar, aprendendo um modelo juntos por meio da interação segura com resultados intermediários [Huang et al. 2022];

- **Dispositivos Móveis em Edge Computing:** No contexto de aprendizado federado e *blockchain*, múltiplos dispositivos móveis colaboram para realizar o treinamento distribuído de IA, construindo um modelo global compartilhado sem *offloading* de todos os dados brutos para um *cloud* ou *data center* [Nguyen et al. 2021].
- **Dispositivos de Maquinário:** Mencionados no contexto de tomada de decisões inteligentes baseadas em aprendizado federado em cenários industriais e de IoT [Sun et al. 2020];
- **Arquitetura Federada de Banco de Dados:** Composta por um conjunto de sistemas de banco de dados independentes que se unem para compartilhar e trocar informações, mantendo autonomia individual [Fenoglio, Pithadia e Treleaven 2023];
- **Estrutura de Edge Computing:** Descrita como uma arquitetura de rede federada, com colaboração entre a nuvem e dispositivos de borda, organizada em camadas terminais, camadas de borda e camadas de nuvem [Ye et al. 2020];
- **Dispositivos de Usuário em Framework de Aprendizado Federado:** Usados para treinar modelos globais em dispositivos individuais usando dados locais, sem a necessidade de centralizar os dados [Konečný et al. 2016].

Essas entidades e dispositivos são exemplos de componentes que, embora mantenham sua autonomia, colaboram em uma federação para alcançar objetivos comuns, preservando a privacidade e a segurança dos dados. A Tabela 22 organiza os exemplos de entidades federadas, categorizando-os por tipo de entidade, exemplos específicos, e o contexto em que são utilizados.

Tabela 22 – Exemplos de Entidades Federadas em Diferentes Contextos

Categoria	Exemplos	Contexto
Proprietários de Dados	Proprietários de Dados Locais	Colaboração para computação de resultados parciais preservando privacidade.
Comércio	Supermercados, Lojas, etc.	Aprendizado federado de transferência com interação segura.
Instituições Financeiras	Bancos em diferentes regiões	Aprendizado federado de transferência com interação segura.
Dispositivos Móveis	Dispositivos móveis em Edge Computing	Treinamento distribuído de IA em blockchain.
Dispositivos Industriais	Dispositivos de maquinário	Decisões inteligentes em cenários industriais e IoT.
Sistemas de Banco de Dados	Arquitetura federada de banco de dados	Compartilhamento e troca de informações mantendo autonomia.
Estruturas de Rede	Estrutura de Edge Computing	Colaboração entre nuvem e borda organizada em camadas.
Dispositivos de Usuário	Dispositivos de usuário em framework de aprendizado federado	Treinamento de modelos globais usando dados locais.

Fonte: O autor

Como pode ser visto, diferentes setores usam o aprendizado federado e sistemas distribuídos para proteger e melhorar a eficiência. Ela inclui exemplos de vários campos, como comércio, finanças e dispositivos móveis, cada um utilizando essas tecnologias para colaboração segura e tomada de decisões inteligentes.

Ao implementar sistemas federados, detalharemos as vantagens e desvantagens que acompanham a adoção desses sistemas, fornecendo uma perspectiva equilibrada para auxiliar na tomada de decisões informadas:

- **Vantagens**
 1. **Autonomia de Dados:** As entidades mantêm controle total sobre seus dados, crucial para organizações que priorizam a privacidade e segurança dos dados [Tong et al. 2023];

2. **Colaboração sem Centralização:** Facilita a cooperação entre várias organizações sem a necessidade de centralizar o controle ou os dados [Fenoglio, Pithadia e Treleaven 2023];
3. **Eficiência em Privacidade:** Processamento local dos dados, aumentando a proteção da privacidade [Xia et al. 2021];
4. **Flexibilidade e Escalabilidade:** Sistemas federados são escaláveis, permitindo que novos participantes se juntem sem interrupções significativas [Heimbigner e McLeod 1985];
5. **Aplicabilidade em Diversos Setores:** Utilizados em uma variedade de contextos como IoT, Edge Computing e inteligência artificial [Nguyen et al. 2021, Shuvo et al. 2022, Abbas et al. 2017];
6. **Redução de Carga nos Servidores Centrais:** Distribui o processamento e armazenamento de dados, melhorando a performance do sistema [Varghese et al. 2016, Ye et al. 2020];
7. **Adaptação Localizada:** Permite que cada entidade adapte suas operações e sistemas conforme necessidades locais [Fenoglio, Pithadia e Treleaven 2023, Khan et al. 2019];
8. **Resiliência a Falhas:** Maior resistência a falhas, pois problemas em uma entidade não afetam outras [Obermaisser et al. 2009, Khan et al. 2019].

- **Desvantagens**

1. **Complexidade de Gerenciamento:** Gerenciar um sistema federado é complexo, especialmente na coordenação entre entidades autônomas [Heimbigner e McLeod 1985, Fernandez, Zhao e Wijegunaratne 2003];
2. **Latência em Comunicação:** A comunicação entre entidades geograficamente distantes pode introduzir latências [Satyanarayanan 2017];
3. **Desafios de Interoperabilidade:** Diferenças em tecnologias e padrões podem complicar a integração entre sistemas federados [Khan et al. 2019];
4. **Segurança Distribuída:** A natureza distribuída pode introduzir vulnerabilidades de segurança [Abbas et al. 2017, Khan et al. 2019];
5. **Custo de Implementação:** Estabelecer e manter uma arquitetura federada pode ser custoso [Obermaisser et al. 2009, Heimbigner e McLeod 1985];
6. **Desafios na Consistência de Dados:** Manter a consistência dos dados entre entidades pode ser desafiador [Heimbigner e McLeod 1985];

- 7. **Complexidade de Compliance:** Conformidade com regulamentações variadas pode ser complicada [Society 2020, Fenoglio, Pithadia e Treleaven 2023];
- 8. **Dificuldades na Governança de Dados:** Estabelecer governança sobre dados compartilhados pode ser difícil [Society 2020].

As entidades federadas são importantes na gestão de sistemas distribuídos, onde colaboração e autonomia são essenciais para a eficiência. Esse modelo permite que diferentes entidades, como organizações ou sistemas, trabalhem juntas sem centralizar o controle ou os dados. Apesar das vantagens, como autonomia sobre os dados e colaboração satisfatória, há desafios na gestão de um sistema tão complexo [Heimbigner e McLeod 1985].

A autonomia sobre os próprios dados é uma grande vantagem, permitindo que as entidades federadas mantenham controle total sobre suas informações. No entanto, isso pode criar desafios na consistência dos dados, exigindo esforços para garantir precisão e atualizações constantes. Além disso, a flexibilidade dos sistemas federados facilita a adesão de novos participantes e a adaptação a novas necessidades. Porém, essa flexibilidade também traz complexidade no cumprimento de regulamentos, o que pode ser um desafio de gestão. Os altos custos de implementação e manutenção são outro obstáculo, podendo dificultar a adoção de sistemas federados por organizações com recursos limitados [Heimbigner e McLeod 1985].

A tabela 23 resume as principais vantagens e desvantagens das entidades federadas, destacando os pontos críticos deste modelo de operação.

Tabela 23 – Vantagens e Desvantagens das Entidades Federadas

Categoria	Vantagens	Desvantagens
Controle de Dados	Autonomia total sobre dados próprios.	Desafios na consistência de dados.
Colaboração	Permite cooperação sem centralizar controle.	Complexidade de gerenciamento.
Privacidade	Eficiência em proteger a privacidade.	Vulnerabilidades de segurança distribuída.
Flexibilidade	Facilita a adesão de novos participantes.	Complexidade de compliance com regulamentações.
Eficiência	Reduz carga em servidores centrais.	Latência em comunicação entre entidades.

Fonte: Seção B.3

Em resumo, os sistemas federados equilibram benefícios e desafios. As vantagens incluem maior controle de dados e eficiência na colaboração entre entidades, o que é importante para a segurança e privacidade. Contudo, esses sistemas também apresentam dificuldades como gerenciamento complexo, latência na comunicação e desafios de interoperabilidade, que podem complicar a sua implementação e manutenção. Portanto, embora ofereçam soluções para trabalhar com várias organizações, é necessário considerar cuidadosamente seu gerenciamento e segurança [Tong et al. 2023].

Relembrando, a arquitetura federada é ideal para situações que exigem cooperação entre várias partes, mas onde centralizar tudo não é desejável ou possível. As entidades federadas têm seus próprios dados e recursos que são importantes para fazer previsões, por exemplo, do tempo e impactos ambientais. Cada uma delas cuida dos seus dados e recursos. Aqui estão alguns exemplos de entidades que participam ou podem participar desse tipo de sistema:

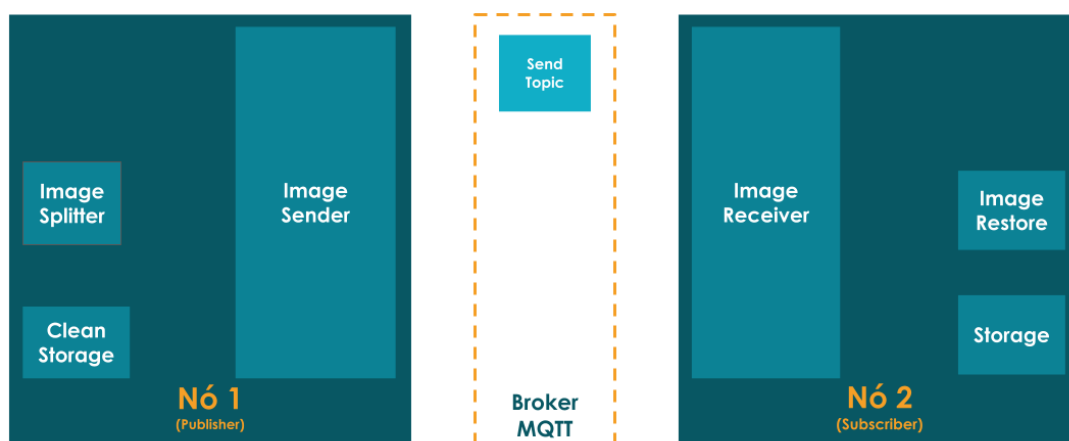
- União Européia(UE);
- Governos;
- Instituições de ensino;
- Organizações não governamentais(ONGs).

APÊNDICE C – Fluxograma de comunicação

C.1 Fluxogramas do método V1

No método V1 o sistema possui apenas um tópico (*Send Topic*), este tópico não se repete no método V2 e os componentes de cada um dos agentes está ilustrado na Figura 35.

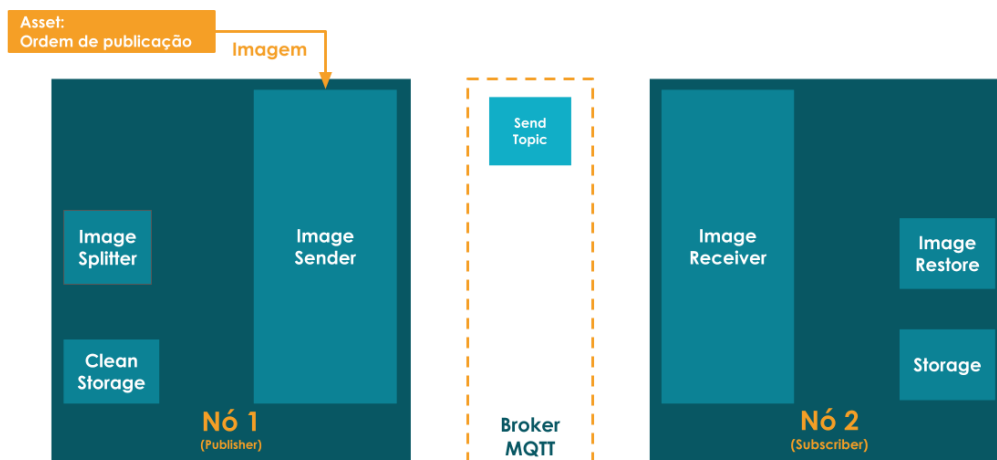
Figura 35 – Fluxograma dos módulos de comunicação e suas interações: componentes do sistema



Fonte: o autor

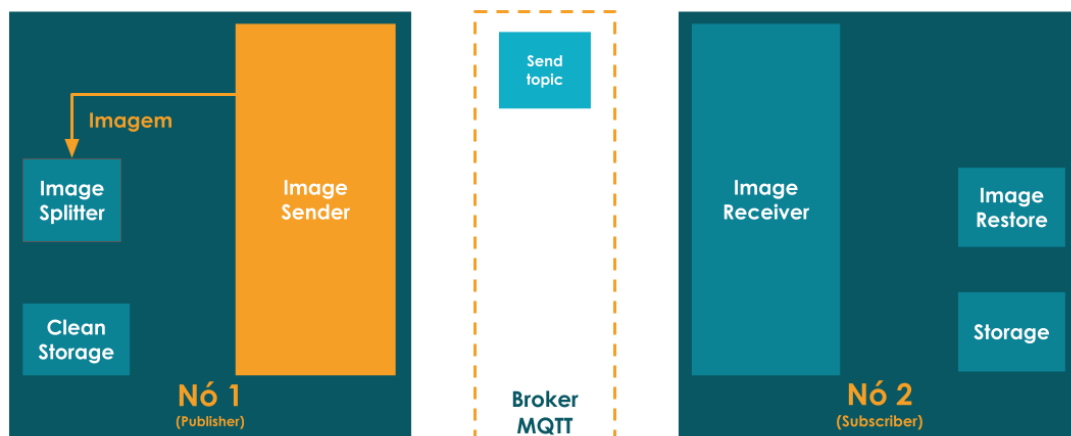
O momento 1 se inicia com uma ordem de publicação de uma imagem para o *ImageSender*, como visto na Figura 36.

Figura 36 – Fluxograma dos módulos de comunicação e suas interações: Momento 1



Fonte: o autor

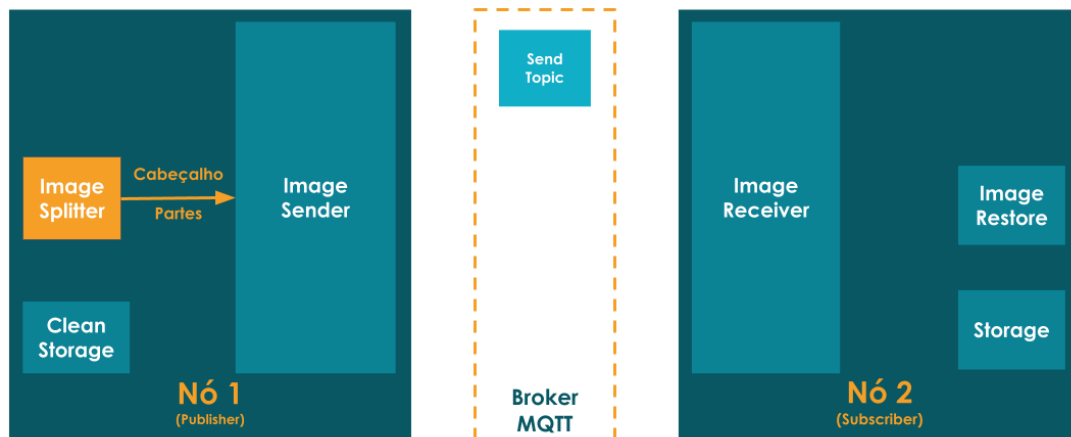
Na Figura 37, o *ImageSender* encaminha essa imagem para o *ImageSplitter*.

Figura 37 – Fluxograma dos módulos de comunicação e suas interações: *ImageSplitter*

Fonte: o autor

O *ImageSplitter* em posse dessa imagem gera um *hash* único para ela, salva esse *hash* em um arquivo de cabeçalho e neste cabeçalho também salva outras informações relativas à imagem a ser trafegada na rede. Entre essas informações estão nome, tamanho e número de partes a serem geradas. E então ele fraciona as imagens nessa quantidade e as devolve para o *ImageSender* juntamente do cabeçalho, como ilustrado na Figura 38.

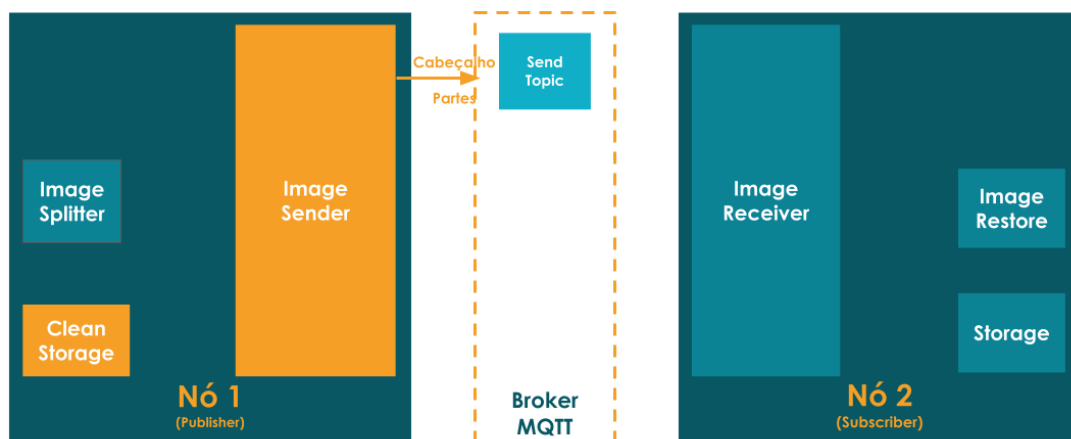
Figura 38 – Fluxograma dos módulos de comunicação e suas interações: Cabeçalho e fragmentos de imagem



Fonte: o autor

O *ImageSender* envia o cabeçalho e as partes em sequência por meio do *SendTopic*. E o *CleanStorage* limpa os dados relativos à imagem enviada, conforme 39.

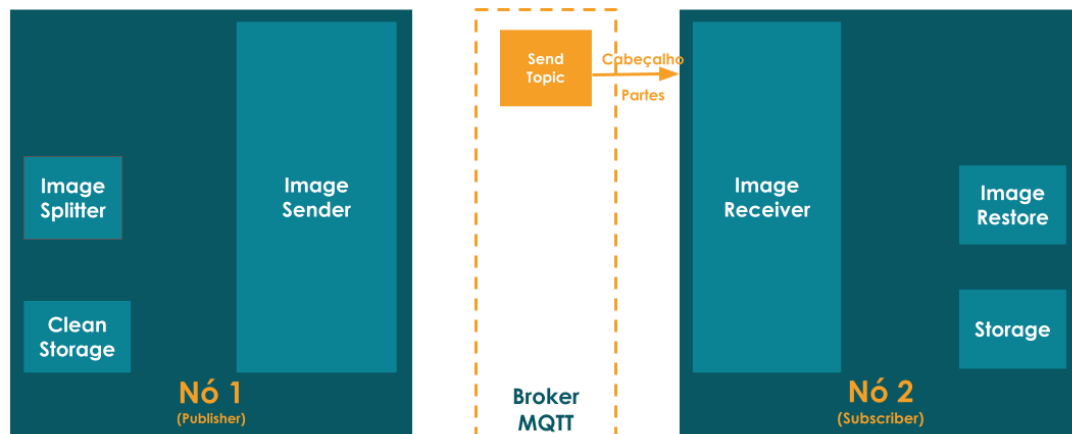
Figura 39 – Fluxograma dos módulos de comunicação e suas interações: Envio de imagens



Fonte: o autor

Isso nos leva ao momento 2 que começa com a chegada dessas partes por meio do *ImageReceiver*, ilustrado na Figura 40.

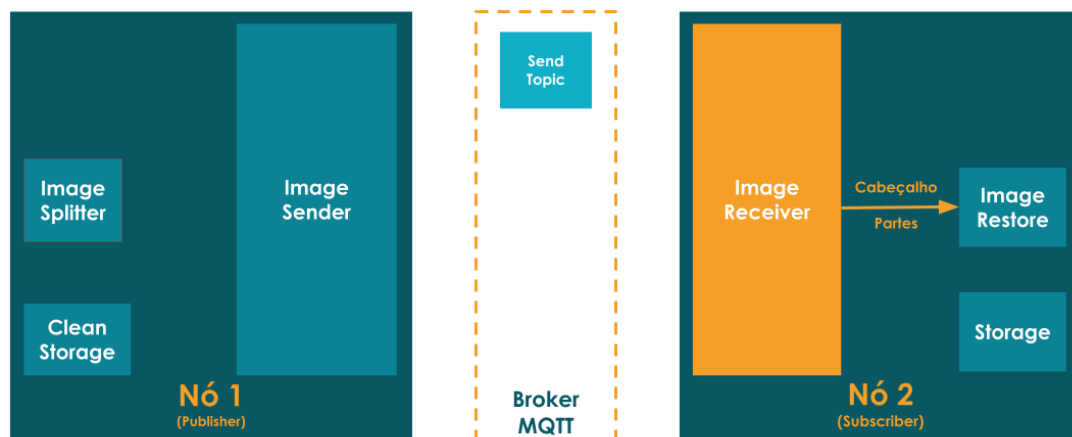
Figura 40 – Fluxograma dos módulos de comunicação e suas interações: Momento 2



Fonte: o autor

ImageReceiver avisa o *ImageRestore* da chegada de uma nova imagem, como pode ser visto na Figura 41.

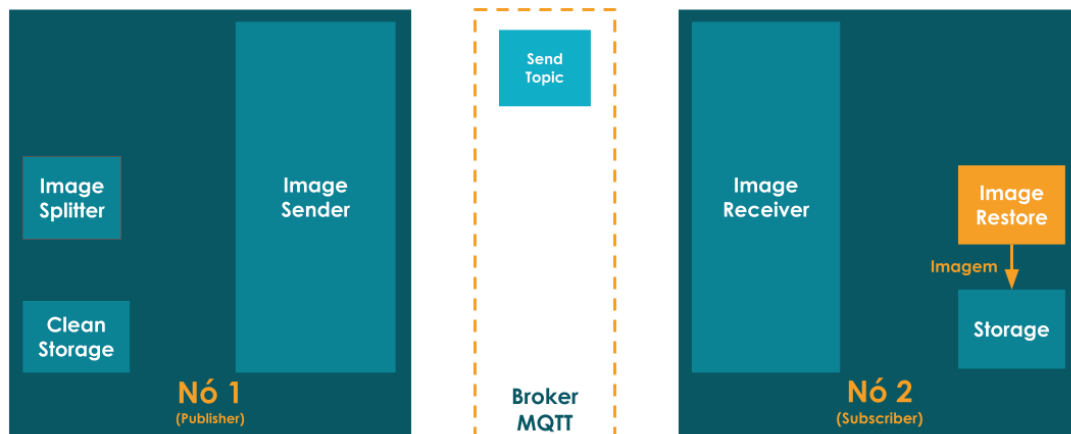
Figura 41 – Fluxograma dos módulos de comunicação e suas interações: Chegada das partes



Fonte: o autor

O *ImageRestore* remonta a imagem, conforme Figura 42, levando-nos ao momento 3.

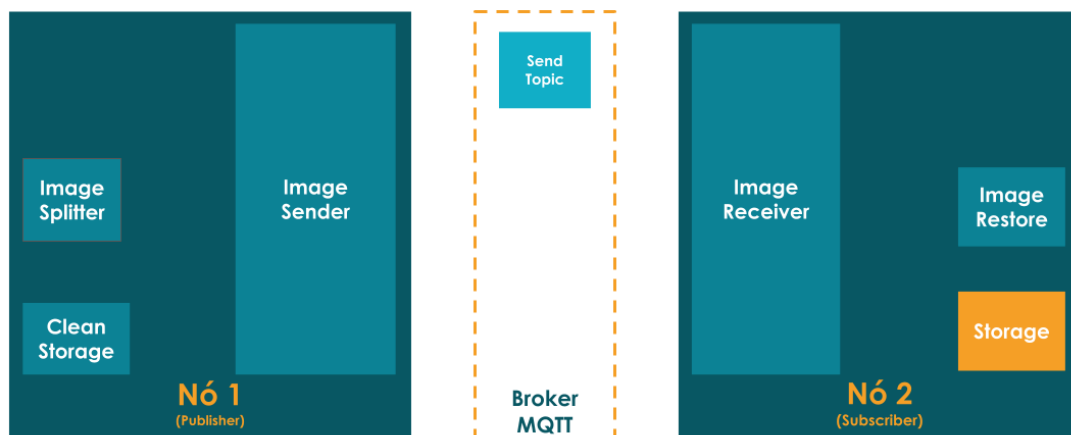
Figura 42 – Fluxograma dos módulos de comunicação e suas interações: Restauração



Fonte: o autor

O momento 3 aguarda que todas as imagens recebidas tenham sido restauradas e salvas por meio do *Storage*. E assim o processo termina. Ilustrado na Figura 43.

Figura 43 – Fluxograma dos módulos de comunicação e suas interações: Momento 3



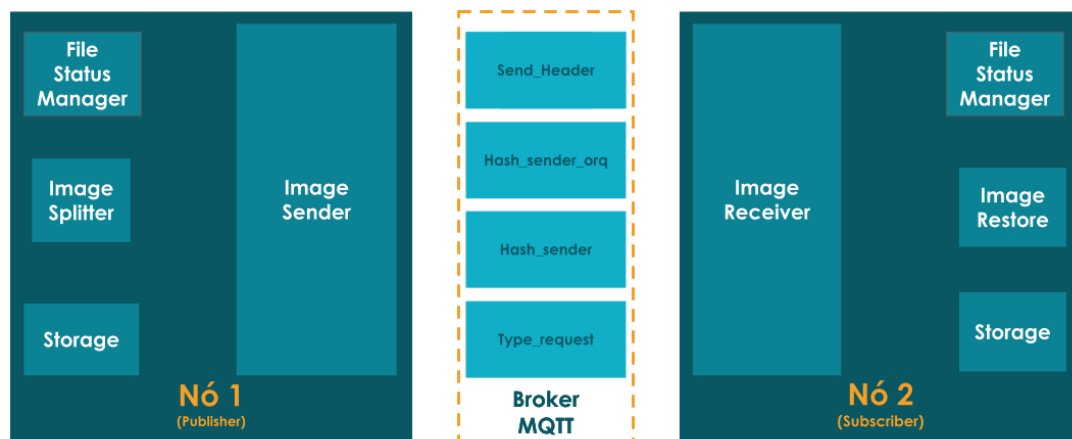
Fonte: o autor

Fim do processo, até que uma nova ordem de publicação de uma imagem para o *ImageSender* seja feita.

C.2 Método V2

O sistema possui inicialmente um tópico e os componentes de cada um dos agentes está ilustrado na imagem. Destaque para o novo componente, o file status manager, visto na Figura 44.

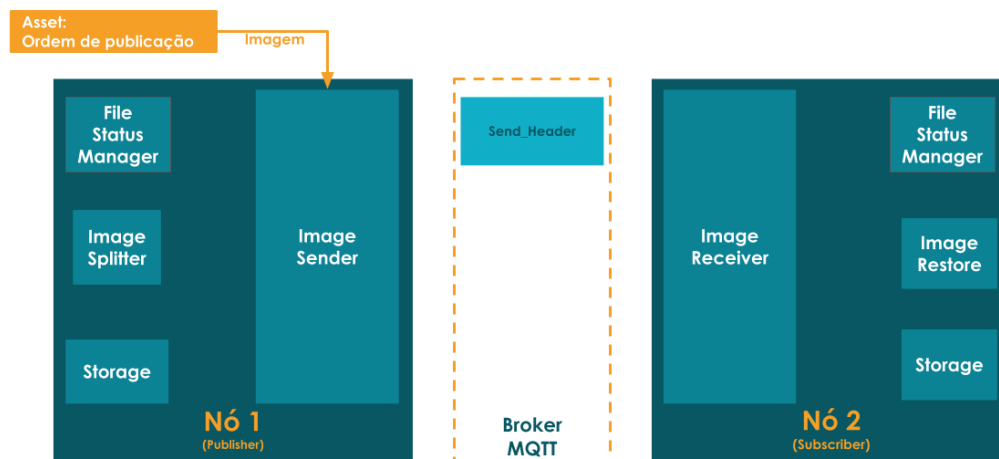
Figura 44 – Fluxograma dos módulos de comunicação e suas interações (V2): componentes do método V2



Fonte: o autor

O momento 1 se inicia com uma ordem de publicação de uma imagem para o *ImageSender*, como visto na Figura 45.

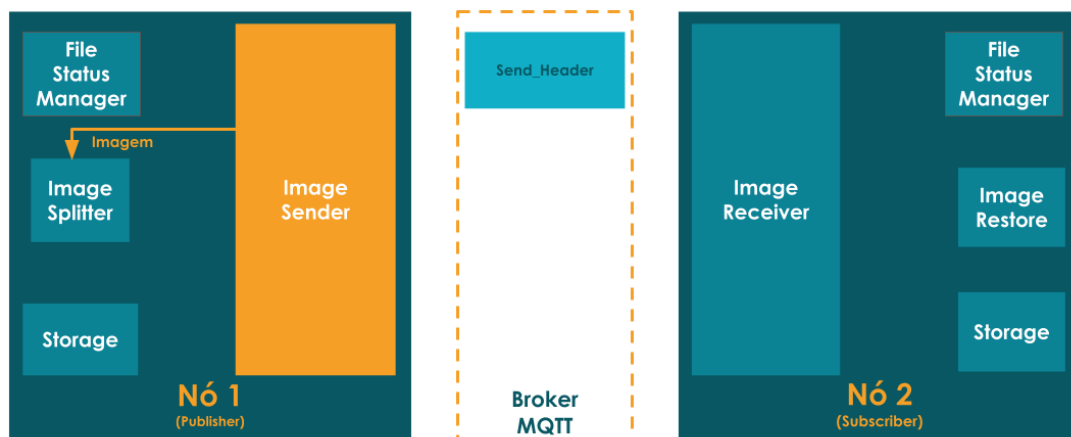
Figura 45 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 1



Fonte: o autor

O *ImageSender* encaminha essa imagem para o *ImageSplitter*, como ilustrado na Figura 46.

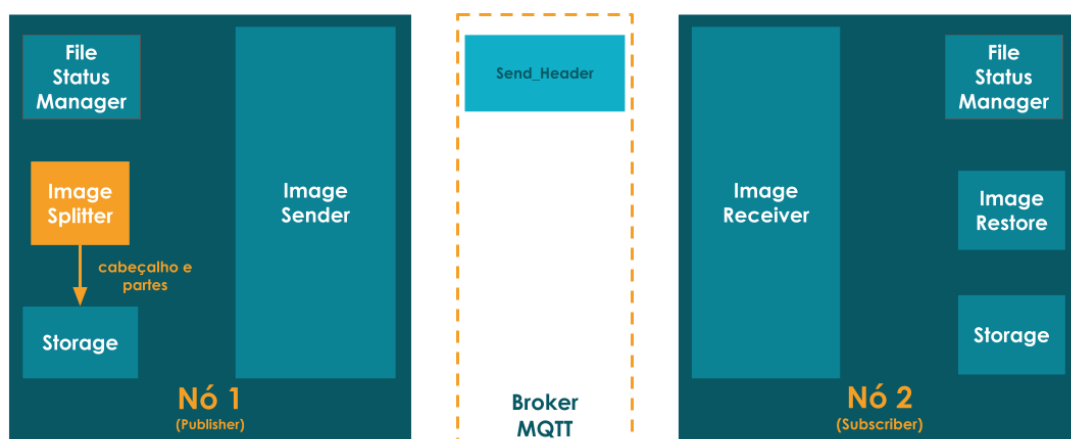
Figura 46 – Fluxograma dos módulos de comunicação e suas interações (V2): *ImageSplitter*



Fonte: o autor

O *ImageSplitter* em posse dessa imagem gera um *hash_file* único para ela, salva esse *hash_file* em um arquivo de cabeçalho (*header*). A Figura 47 ilustra o processo.

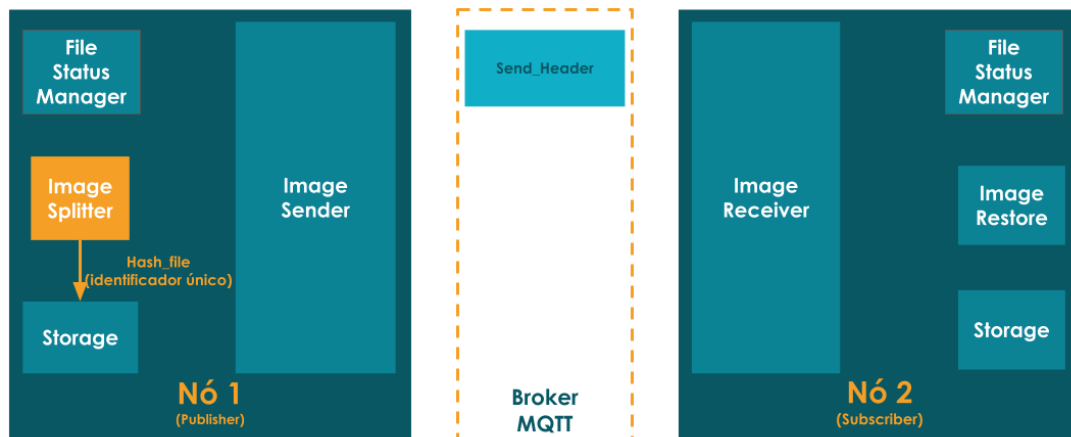
Figura 47 – Fluxograma dos módulos de comunicação e suas interações (V2): Cabeçalho e fragmentos de imagem



Fonte: o autor

Neste cabeçalho também são salvas outras informações relativas à imagem a ser trafegada na rede. Entre essas informações estão nome, tamanho e número de partes a serem geradas. E então ele fraciona as imagens nessa quantidade. Conforme Figura 48.

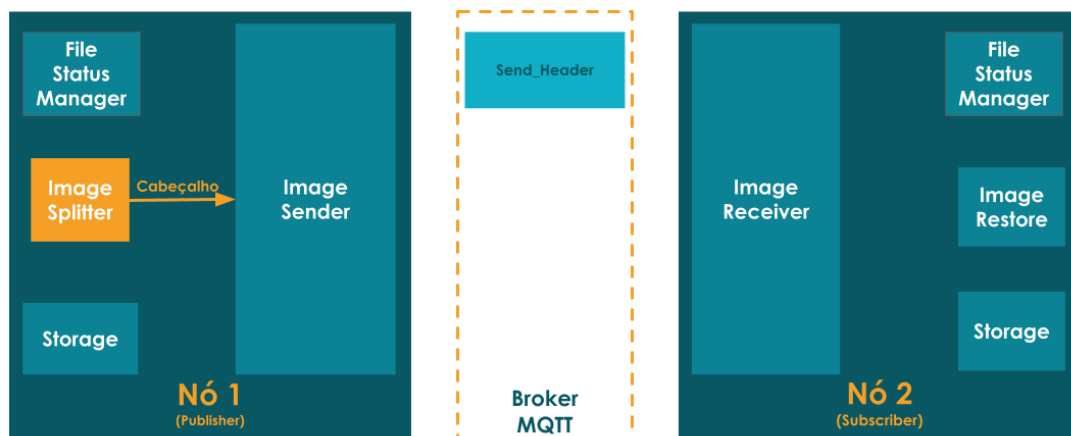
Figura 48 – Fluxograma dos módulos de comunicação e suas interações (V2): Identificador único



Fonte: o autor

Slide 16 Na Figura 49, o cabeçalho com as informações da imagem é direcionado ao *ImageSender*.

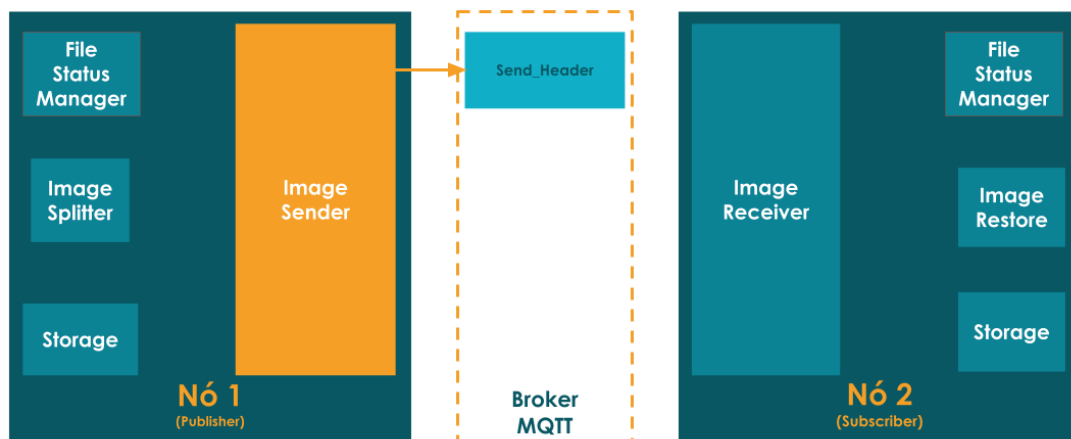
Figura 49 – Fluxograma dos módulos de comunicação e suas interações (V2): Cabeçalho



Fonte: o autor

Ele então é responsável por enviar esse cabeçalho por meio do tópico *Send_header*, como visto na Figura 50.

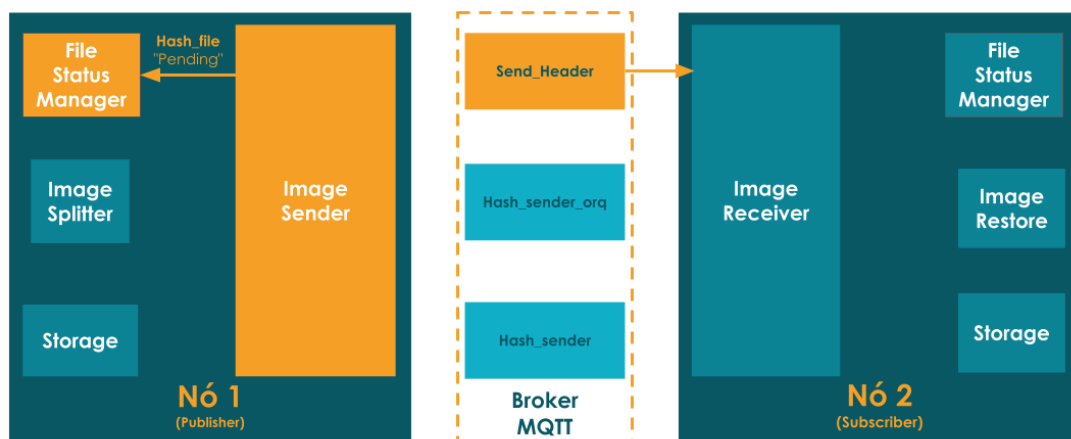
Figura 50 – Fluxograma dos módulos de comunicação e suas interações (V2): Envio do cabeçalho



Fonte: o autor

No momento 2, o status "*pending*" é atribuído ao *hash_file* gerado por meio do *FileStatusManager*, indicando que o *ImageSender* está aguardando confirmação do início da recepção. Caso ocorra um *timeout* sem a confirmação, o *header* é reenviado automaticamente para o *broker*. O processo é ilustrado na Figura 51.

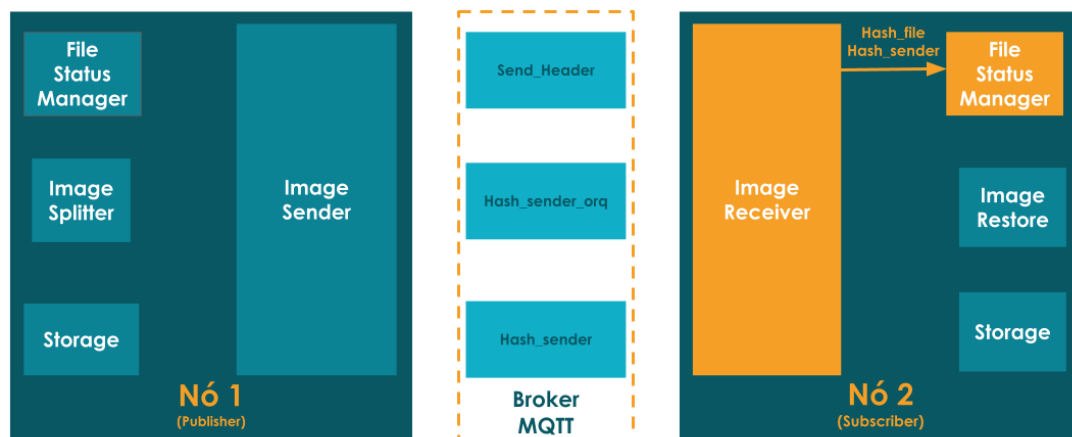
Figura 51 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 2



Fonte: o autor

Ao receber o *header* pelo *Send_Header*, o módulo *ImageReceiver* abre e extrai o *hash_file* e o *hash_sender* contidos nele, conforme Figura 52.

Figura 52 – Fluxograma dos módulos de comunicação e suas interações (V2): Extração de *hash*

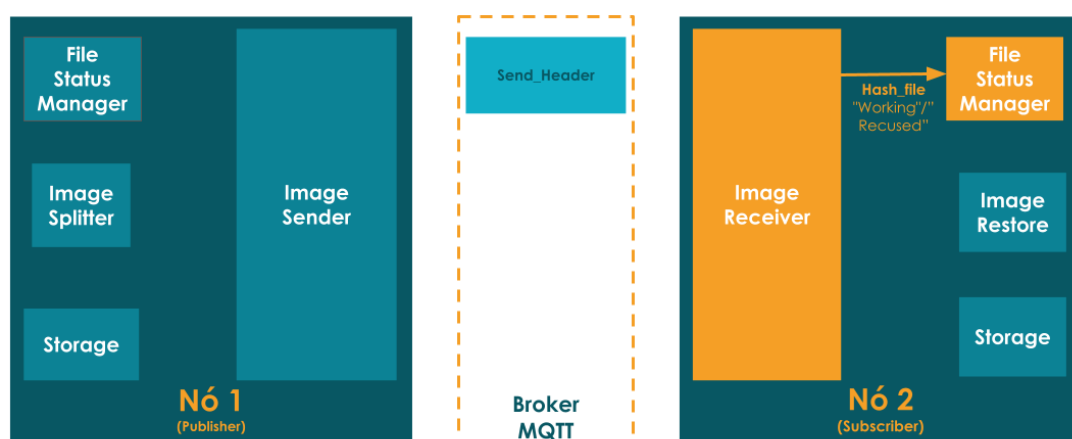


Fonte: o autor

Utilizando o *FileStatusManager*, o status do *hash_file* é atualizado para "*working*", indicando que a recepção da imagem está em andamento. Isto ocorre caso já não exista essa imagem com o status "*completed*" na base de dados do *ImageReceiver*.

Em seguida, o *hash_file* é devolvido pelo tópico *hash_sender* para o *broker*, sinalizando que a recepção foi iniciada com sucesso, caso a imagem já exista, o *hash_file* é acrescido de "*recused*". O processo é ilustrado na Figura 53.

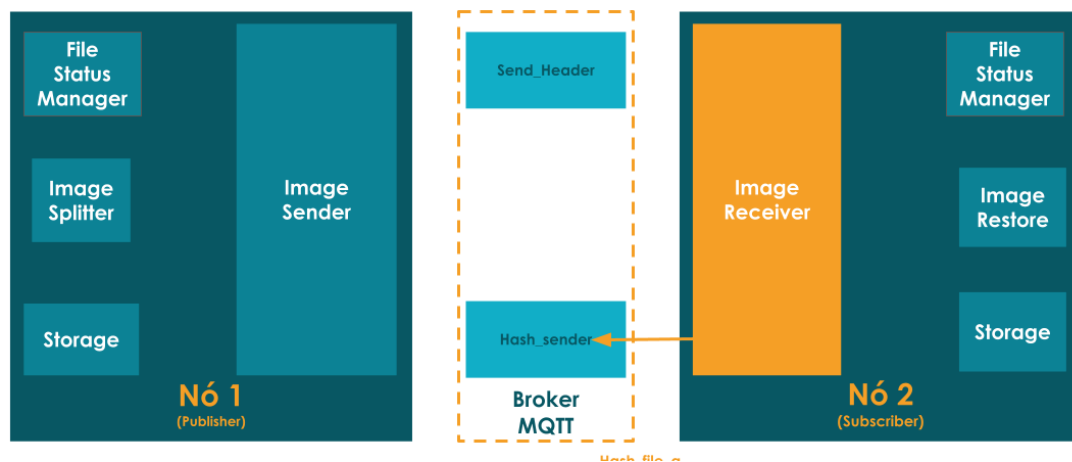
Figura 53 – Fluxograma dos módulos de comunicação e suas interações (V2): *File Status*



Fonte: o autor

O tópico utilizado para devolução de informação ao *ImageReceiver* é o *hash_sender*, ele recebe essa mensagem, e assim termina o momento 2. Ilustrado na Figura 54.

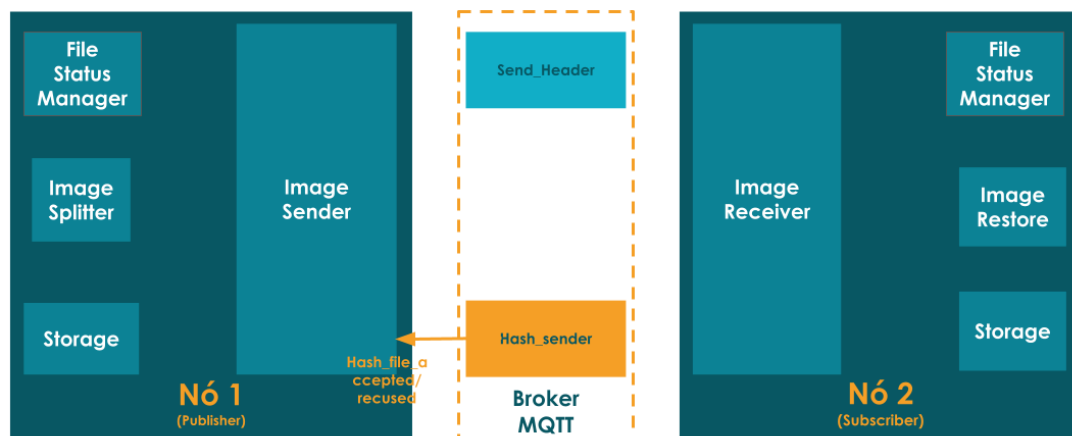
Figura 54 – Fluxograma dos módulos de comunicação e suas interações (V2): Fim do momento 2



Fonte: o autor

O momento 3 se inicia com o *ImageSender*, ao receber o *hash_file* de confirmação, conforme Figura 55.

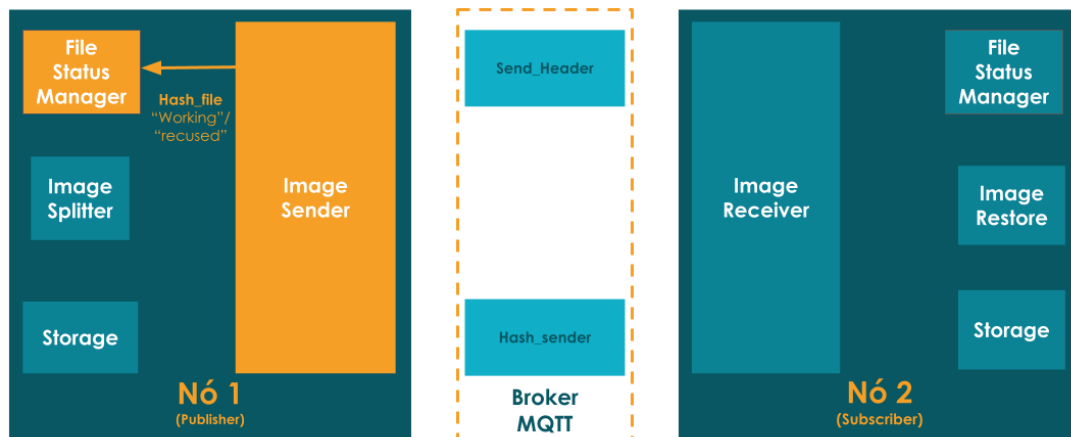
Figura 55 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 3



Fonte: o autor

O *ImageSender*, ao receber o *hash_file* de confirmação, atualiza seu status para "working" através do *FileStatusManager*, porém se a imagem foi recusada pelo *ImageReceiver*, ele atribui "recused" e não envia partes de imagem. O processo é ilustrado na Figura 56.

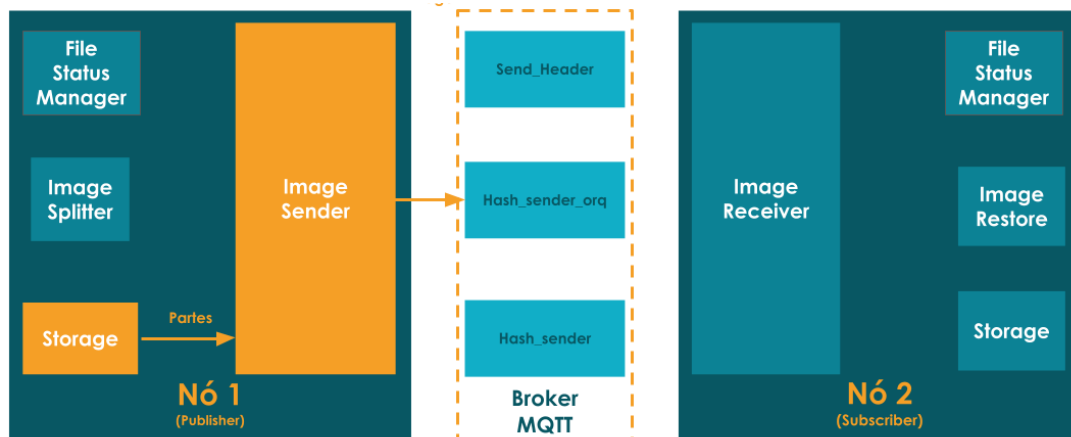
Figura 56 – Fluxograma dos módulos de comunicação e suas interações (V2): Atualização do "file status"



Fonte: o autor

Com esta confirmação, o *ImageSender* busca as partes que estavam guardadas. O *ImageSender* prossegue para enviar todas as partes da imagem pelo tópico *hash_sender_orq*, exclusivo para comunicação com o orquestrador. O processo é ilustrado na Figura 57.

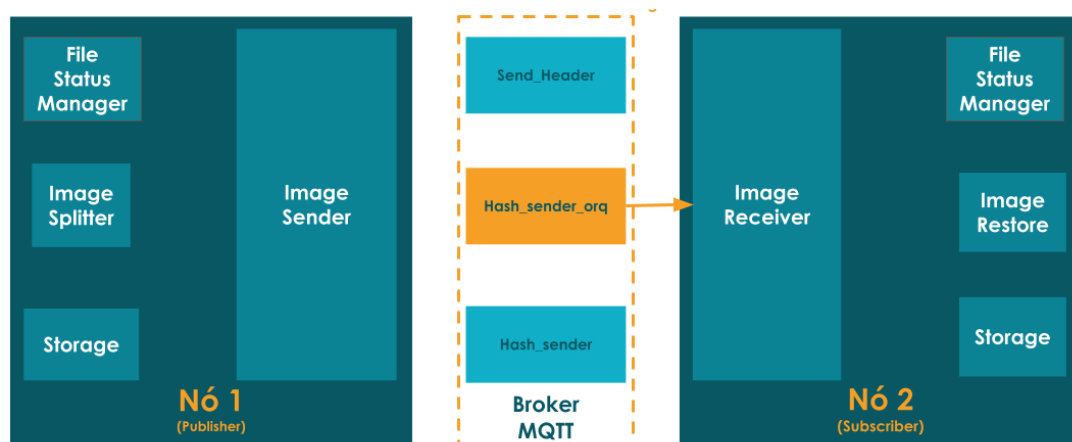
Figura 57 – Fluxograma dos módulos de comunicação e suas interações (V2): Envio dos fragmentos de imagem



Fonte: o autor

Na Figura 58, o momento 4 começa com a chegada dessas partes da imagem.

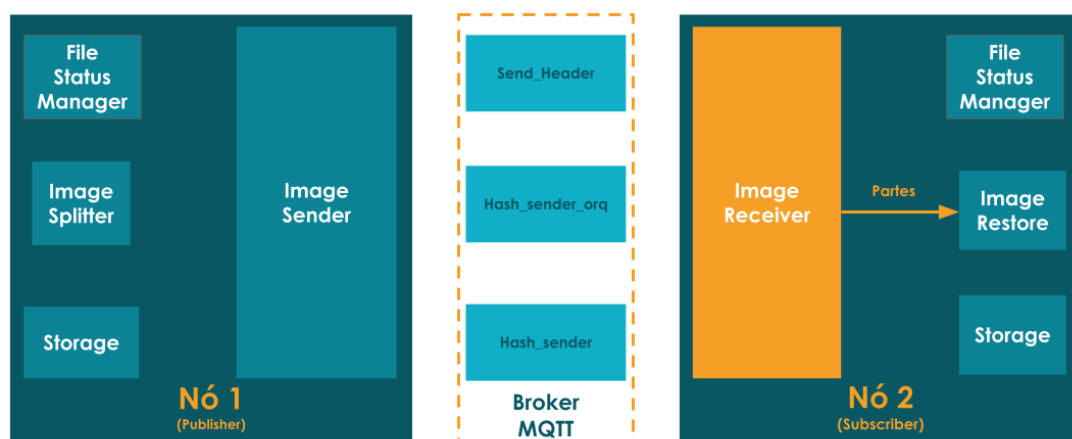
Figura 58 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 4



Fonte: o autor

O *ImageReceiver* encaminha elas para o *ImageRestore*. O *ImageReceiver* recebe as partes da imagem e tenta montar a imagem original utilizando o módulo *ImageRestore*, conforme Figura 59.

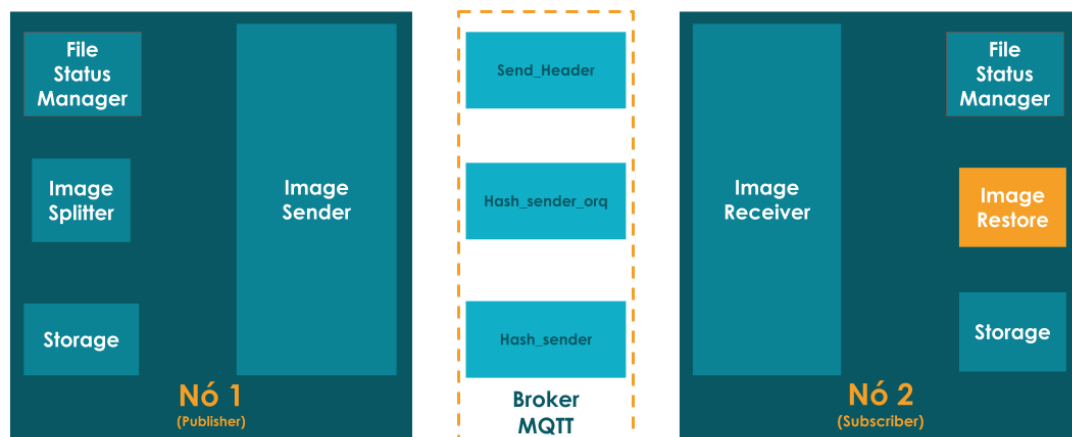
Figura 59 – Fluxograma dos módulos de comunicação e suas interações (V2): *ImageRestore*



Fonte: o autor

O *ImageRestore* tenta montar a imagem. Conforme Figura 60.

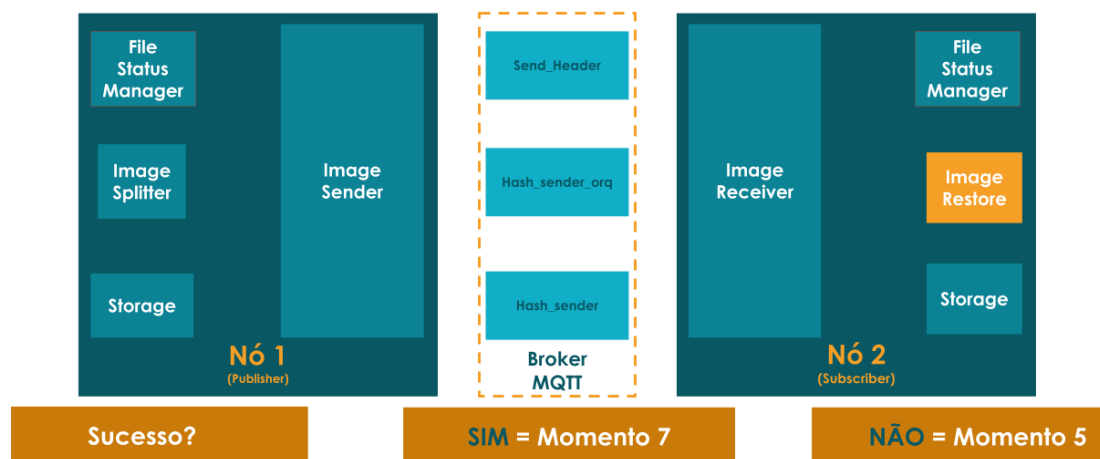
Figura 60 – Fluxograma dos módulos de comunicação e suas interações (V2): Restauração



Fonte: o autor

Chegado o fim do momento 4, caso a montagem seja bem sucedida, avança para o momento 7, caso contrário para o momento 5.

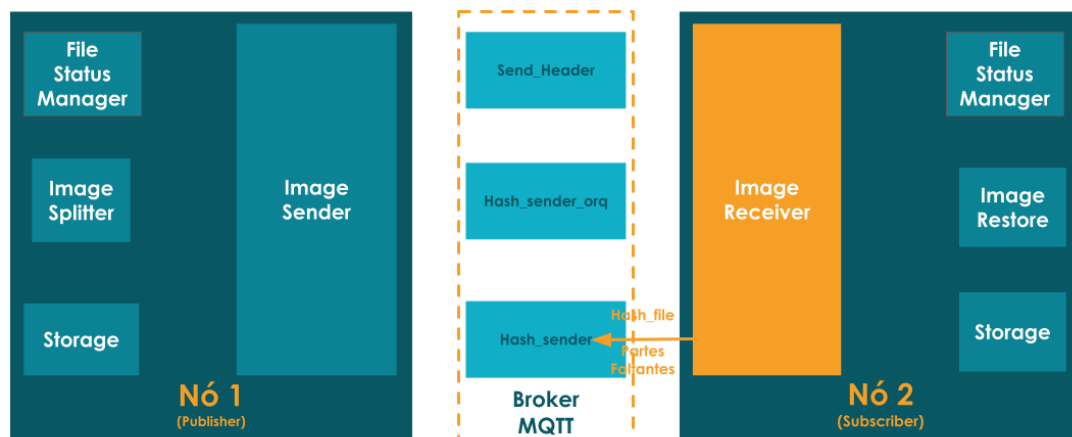
Figura 61 – Fluxograma dos módulos de comunicação e suas interações (V2): Fim do momento 4



Fonte: o autor

No caso do momento 5, quando existem partes faltantes, o *ImageReceiver* solicita essas partes específicas pelo tópico *hash_sender*. Esta solicitação inclui o *hash_file* da imagem e as identificações das partes que não foram recebidas. O processo é ilustrado na Figura 62.

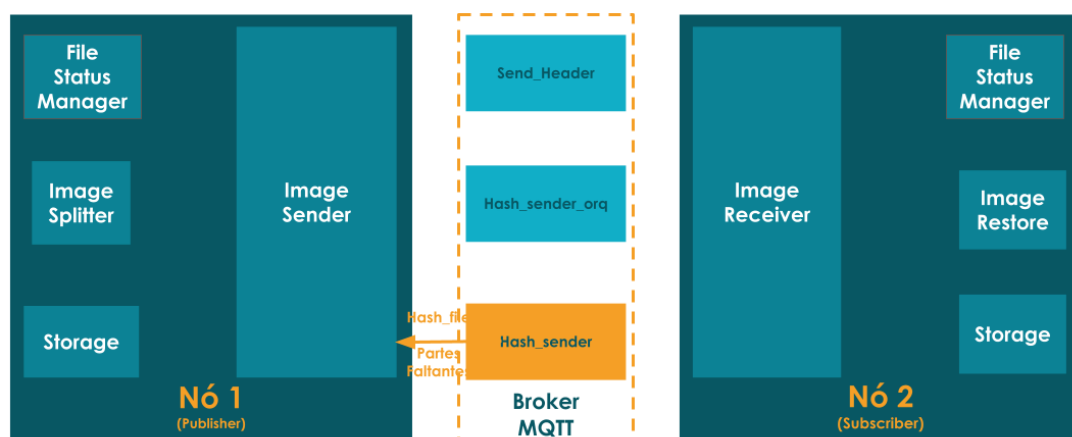
Figura 62 – Fluxograma dos módulos de comunicação e suas interações (V2): Solicitações de reenvios



Fonte: o autor

O *ImageSender* recebe essas informações, iniciando o momento 6. O processo ilustrado na Figura 63.

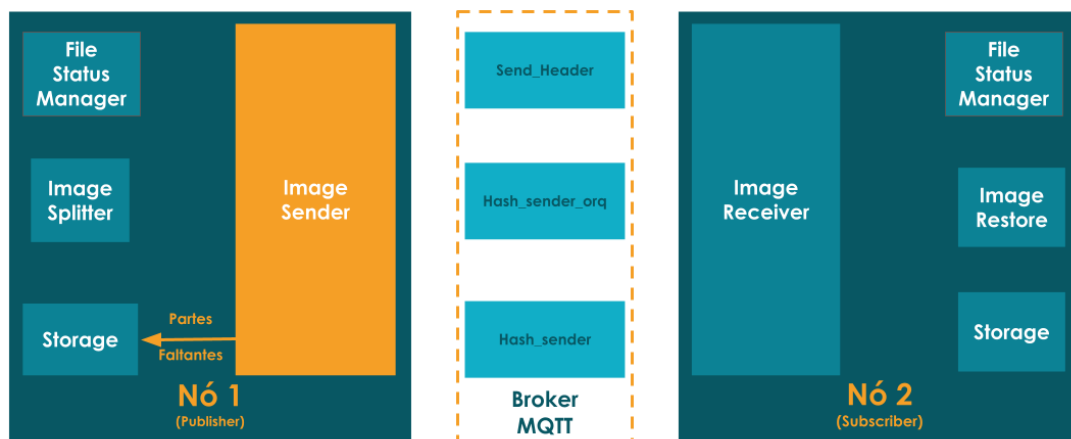
Figura 63 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 6



Fonte: o autor

Como visto na Figura 64, ao receber a informação sobre as partes faltantes, ele busca essas partes.

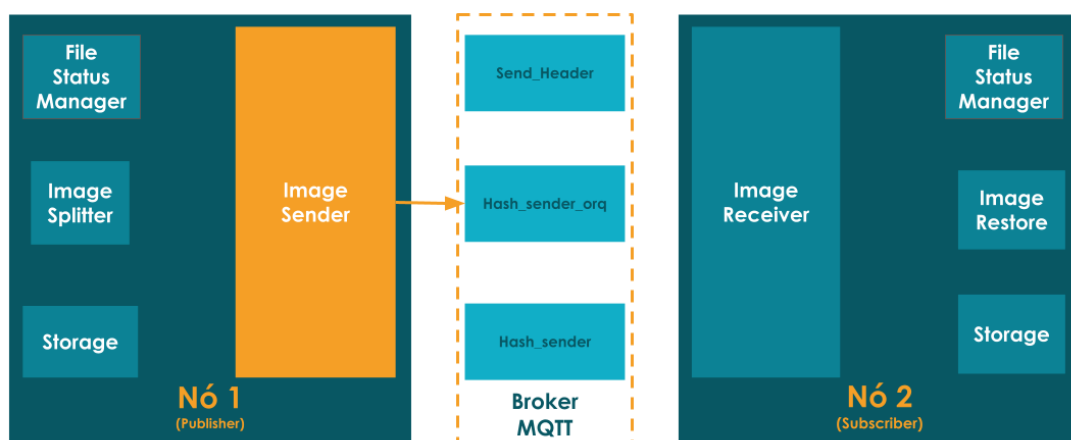
Figura 64 – Fluxograma dos módulos de comunicação e suas interações (V2): Busca das partes



Fonte: o autor

Com o envio, o momento 6 chega ao seu fim, como ilustrado na Figura 65.

Figura 65 – Fluxograma dos módulos de comunicação e suas interações (V2): Fim do momento 6



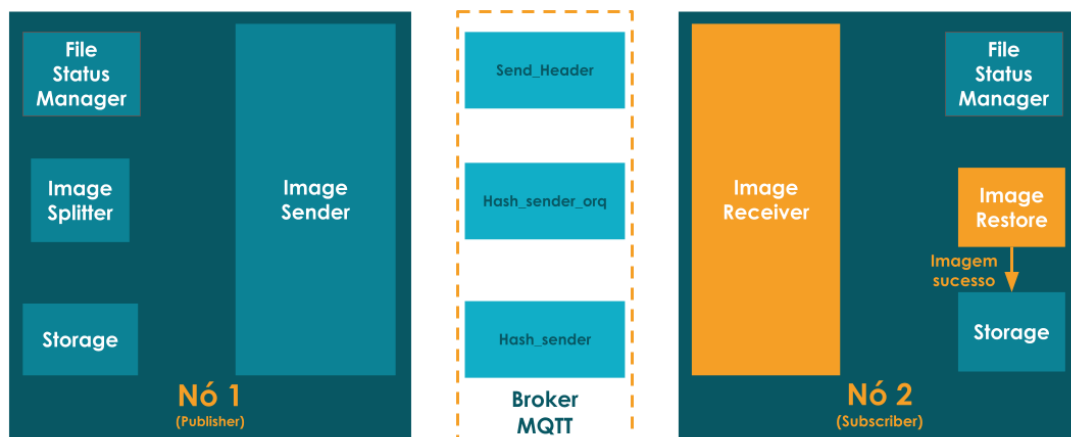
Fonte: o autor

Isso leva o sistema de volta ao Momento 4, que é onde o nó vai tentar montar a imagem.

Este processo de solicitação e reenvio (Momentos 4, 5 e 6) repete-se até que todas as partes da imagem tenham sido recebidas corretamente e a imagem possa ser montada sem qualquer tipo de erro.

Com o sucesso na montagem da imagem, ela é armazenada. O sistema então está no momento 7. O processo é ilustrado na Figura 66.

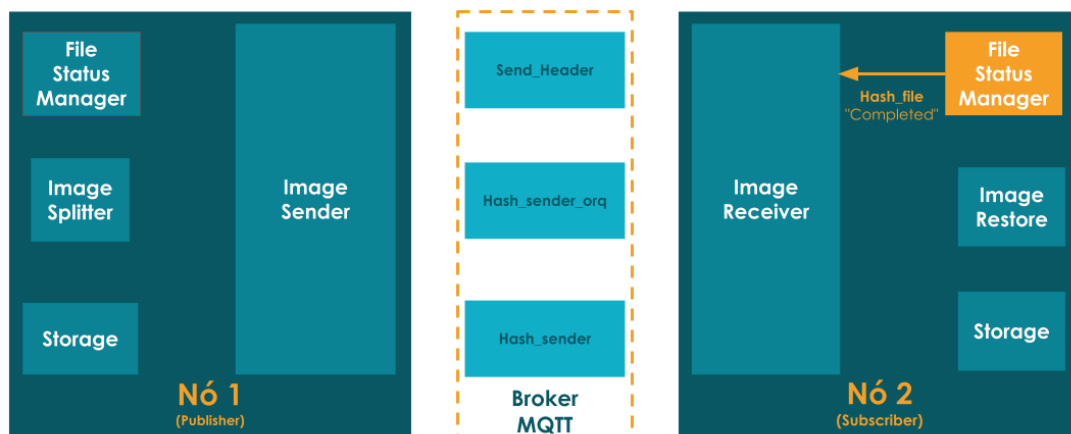
Figura 66 – Fluxograma dos módulos de comunicação e suas interações (V2): Momento 7



Fonte: o autor

O *FilestatusManager* atualiza o status dessa imagem para *completed*, conforme visto na Figura 67.

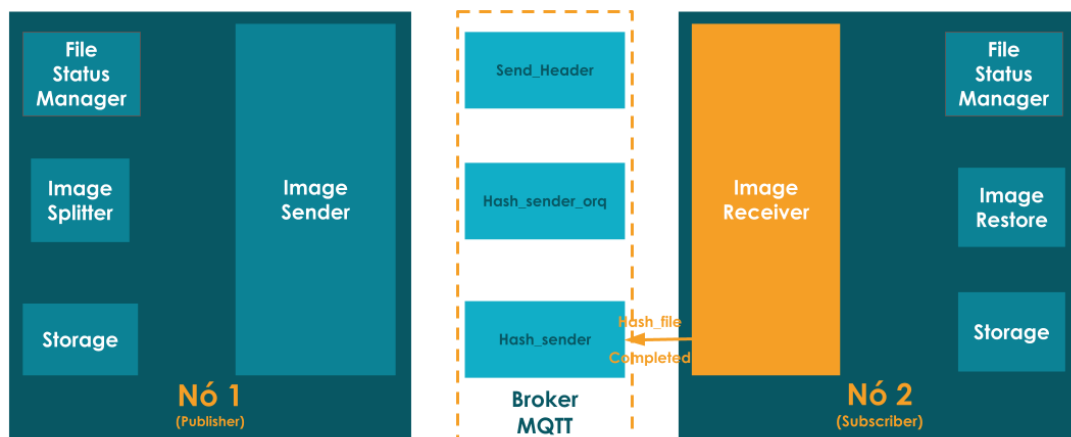
Figura 67 – Fluxograma dos módulos de comunicação e suas interações (V2): Atualização de status



Fonte: o autor

O *ImageReceiver* é avisado do sucesso da operação. Então ele avisa por meio do *hash_sender* o sucesso da transmissão. Chegando ao fim do momento 7. O processo é ilustrado na Figura 68.

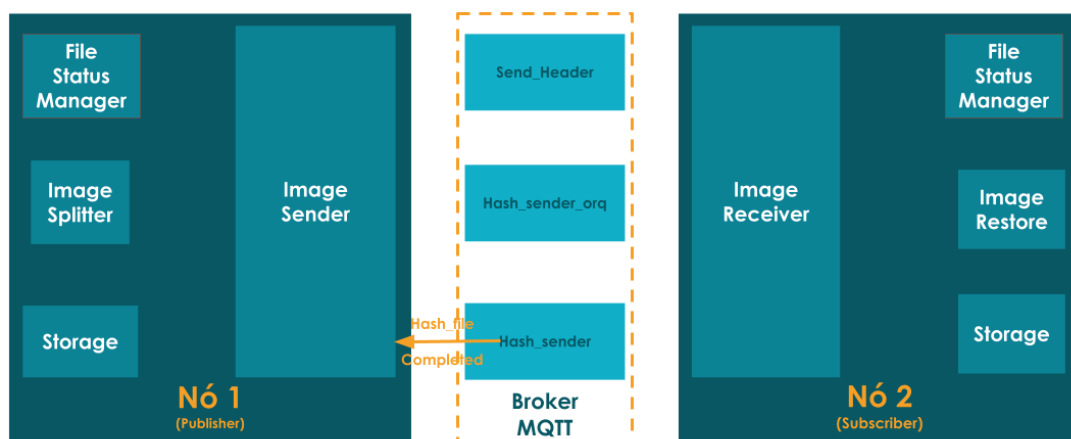
Figura 68 – Fluxograma dos módulos de comunicação e suas interações (V2): Fim do momento 7



Fonte: o autor

No momento 8, essa informação chega no *ImageSender*. Como visto na Figura 69.

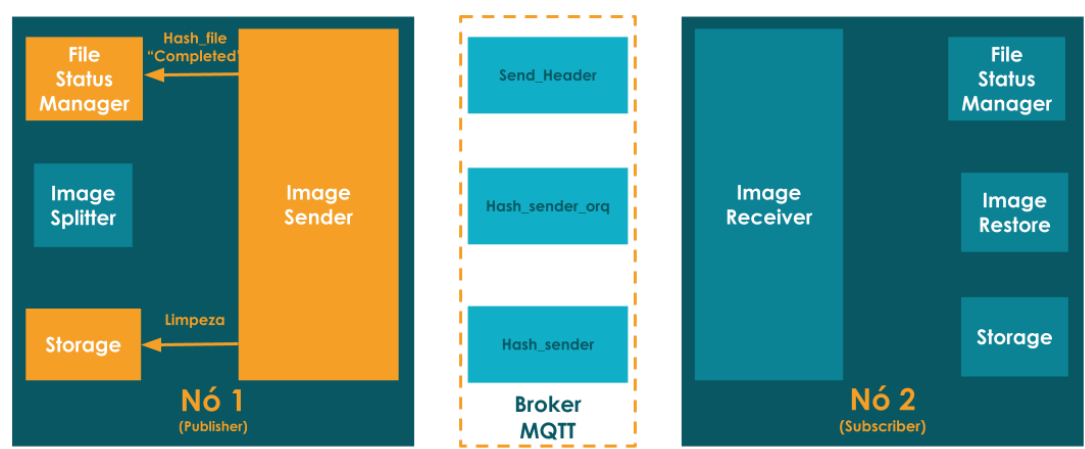
Figura 69 – Fluxograma dos módulos de comunicação e suas interações (V2): Recebimento de sucesso



Fonte: o autor

Ele atribui o status do *hash_file* para "*completed*" por meio do *FileStatusManager* e apaga todos os arquivos temporários gerados, concluindo assim o ciclo de transmissão e recepção da imagem. O processo é ilustrado na Figura 70.

Figura 70 – Fluxograma dos módulos de comunicação e suas interações (V2): Atualização de status e limpeza



Fonte: o autor