



CKP7500 - Sistemas Distribuídos e Redes De Comunicação
SMD - Sistemas Distribuídos
Prof. Windson Viana de Carvalho
Prof. Fernando Antonio Mota Trinta
2025

Relatório AT03 - Paradigmas de Comunicação em Sistemas Distribuídos

Aluno : Leonardo Quezado - 584270

1. Introdução

Este relatório apresenta a implementação prática e a análise comparativa de dois paradigmas fundamentais de comunicação em sistemas distribuídos: **o modelo Publish–Subscribe**, representado pelo protocolo **MQTT**, e o modelo **Requisição–Resposta**, implementado com **REST/HTTP**.

Foram desenvolvidos dois sistemas completos:

1. **Sistema MQTT**: monitoramento de temperatura de uma caldeira industrial por meio de sensores simulados e um sensor físico real.
2. **Sistema REST/HTTP**: calculadora distribuída com tolerância a falhas por meio de política de *retry*.

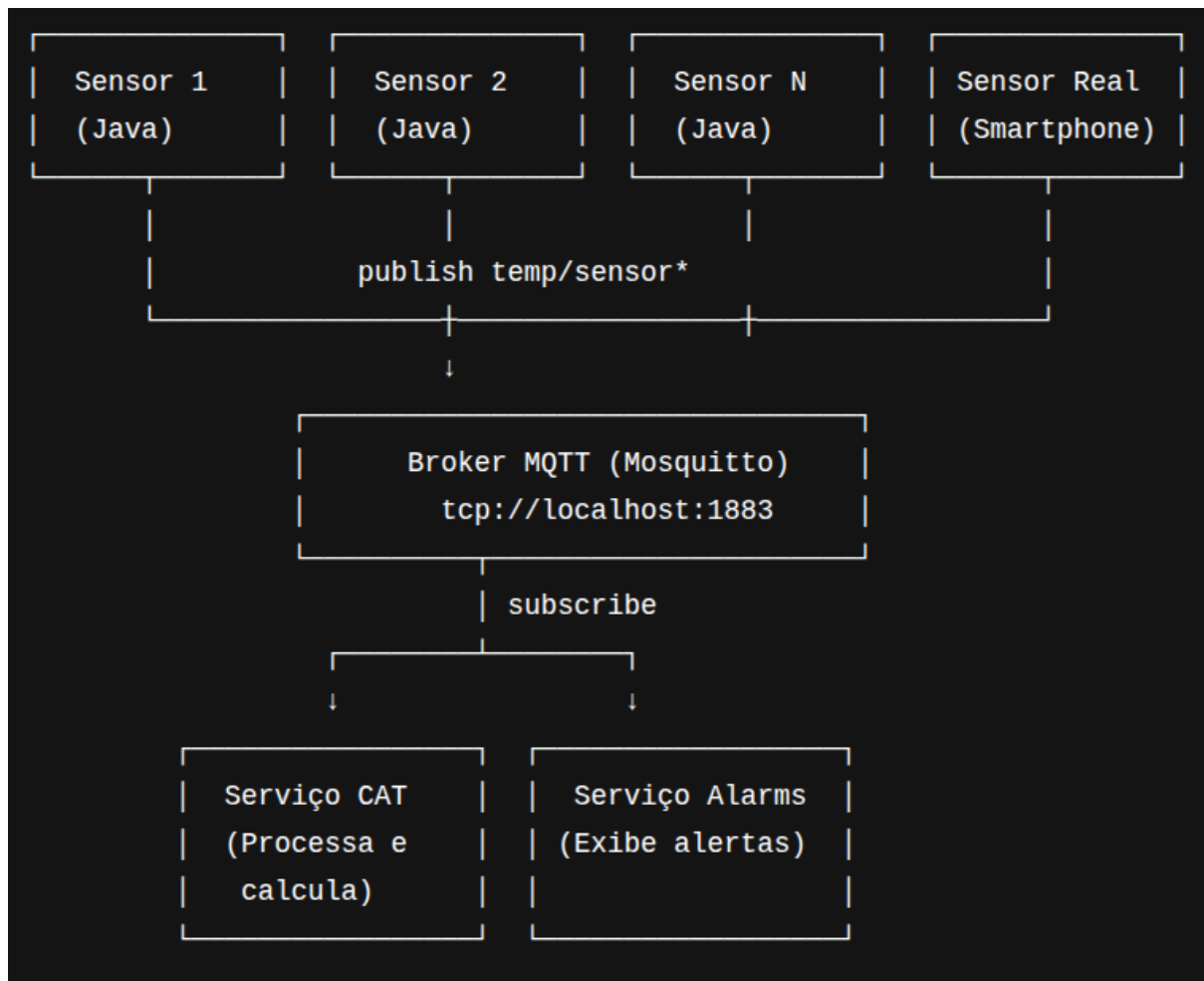
O objetivo é demonstrar, por experimentação prática, diferenças entre os modelos quanto a acoplamento, escalabilidade, sobrecarga de protocolo (*overhead*), desempenho e adequação a cenários específicos. A análise inclui capturas de tráfego obtidas via **Wireshark**, essenciais para compreender o comportamento real das comunicações.

2. Arquiteturas Implementadas

2.1 Arquitetura MQTT – Modelo Publish–Subscribe

O sistema MQTT foi desenvolvido para monitoramento de temperatura em uma caldeira industrial. Múltiplos sensores publicam periodicamente valores de temperatura, enquanto serviços independentes consomem e processam esses dados.

Arquitetura Geral



Características Técnicas Principais

- Modelo: Publish–Subscribe (1:N)
- Acoplamento espacial: **desacoplado**, intermediado por broker
- Acoplamento temporal: **desacoplado** (QoS 1 garante entrega mínima)
- Protocolo: MQTT 3.1.1 sobre TCP

- *Overhead*: ~23 bytes por mensagem (muito baixo)
- Porta padrão: 1883

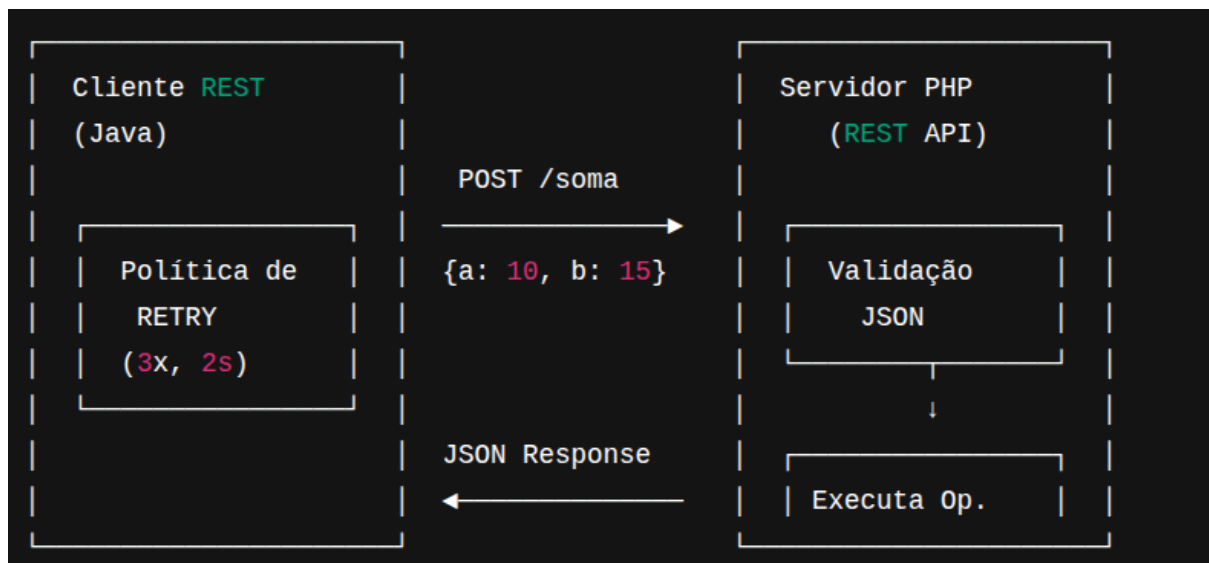
Tópicos MQTT Utilizados

- `temp/sensor1`, `temp/sensor2`, `temp/sensor_real` – leituras
- `alerts/high_temp` – temperatura média > 200 °C
- `alerts/temp_spike` – variação súbita > 5 °C

2.2 Arquitetura REST/HTTP – Modelo Requisição–Resposta

O sistema REST implementa um serviço de cálculo matemático remoto com validação robusta e política de *retry*.

Arquitetura Geral



Características Técnicas

- Modelo: Requisição–Resposta (1:1)
- Acoplamento: **espacial e temporal**

- Protocolo: HTTP/1.1
- Formato: JSON
- *Overhead*: ~500 bytes (headers + payload)
- Tolerância a falhas: política de *retry* (3 tentativas, 2s)

Endpoints

- `POST /soma, /subtracao, /multiplicacao, /divisao`
- `POST /expressao` (avaliação textual)
- `GET /info`

3. Implementação do Sistema MQTT

3.1 Sensores de Temperatura

Foram implementados:

- Dois sensores simulados em Java
- Um sensor real (smartphone Android via IoT MQTT Panel)

Os sensores simulados produzem temperatura entre **180 °C e 220 °C** e publicam a cada **60 segundos** via QoS 1.

Código Base do Sensor (Resumido)

```
double temp = TEMP_MIN + (TEMP_MAX - TEMP_MIN) * random.nextDouble();  
temp = Math.round(temp * 100.0) / 100.0;
```

```
MqttMessage msg = new MqttMessage();  
msg.setPayload(String.valueOf(temp).getBytes());  
msg.setQos(1);
```

```
client.publish("temp/sensor1", msg);  
Thread.sleep(60000);
```

3.2 Serviço CAT – Cálculo da Média Móvel

O serviço CAT consome `temp/#`, mantém uma janela deslizando de 120 s e publica alertas conforme regras.

Algoritmo Essencial

```
long agora = System.currentTimeMillis();
leituras.add(new Leitura(agora, temperatura));
leituras.removeIf(l -> (agora - l.timestamp) > JANELA_TEMPO_MS);
double media = soma / leituras.size();
```

Condições de Alerta

- Média > 200 °C → `alerts/high_temp`
- |média atual – anterior| > 5 °C → `alerts/temp_spike`

3.3 Serviço de Alarmes

Assina `alerts/#` e exibe alertas formatados.

4. Implementação REST/HTTP

4.1 Cliente REST com Retry

Retry ocorre apenas em **erros 5xx**.

Parâmetros:

- 3 tentativas máximas
- 2 segundos de delay entre tentativas
- Retry apenas em erros 5xx (servidor)
- Abort em erros 4xx (cliente)

4.2 Servidor PHP

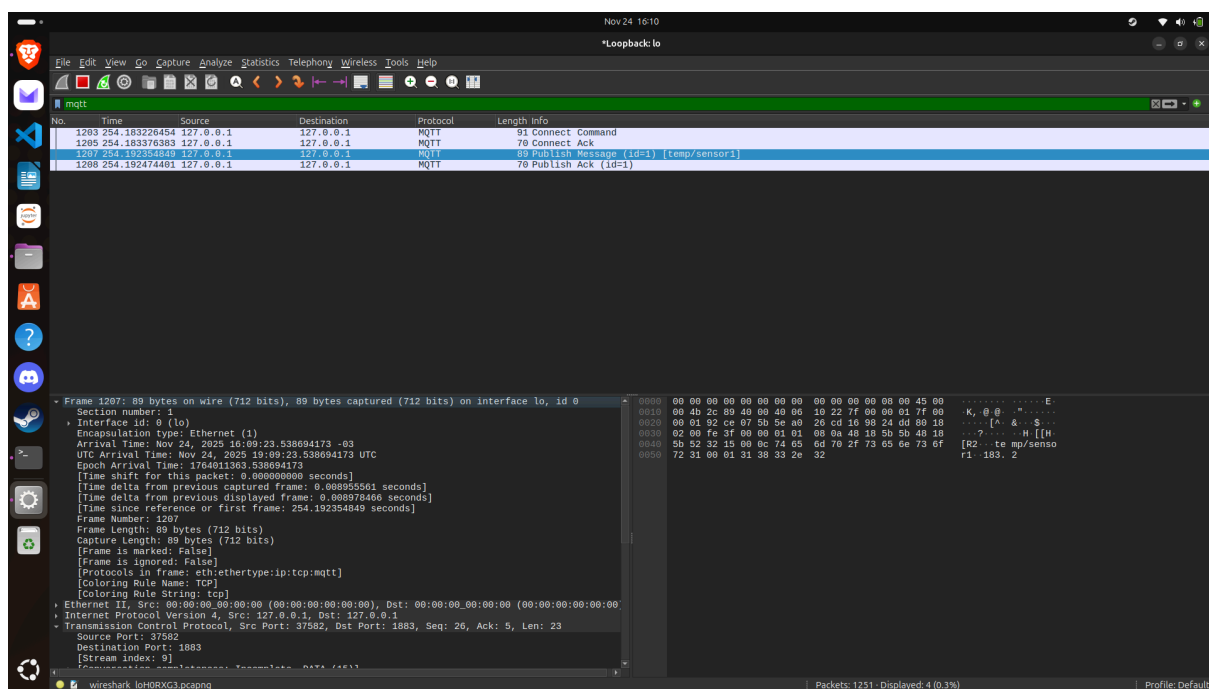
Inclui:

- Roteamento por PATH_INFO
- Validações (método, campos, divisão por zero)
- Sanitização de expressões

5. Análise via Wireshark

5.1 Captura MQTT – Publish

- Frame total: ~89 bytes
- *Overhead* extremamente baixo
- Ideal para IoT



5.2 Captura HTTP – POST

- *Overhead* ~463 bytes
- Headers representam > 90% da carga

The image shows a Wireshark network traffic capture. The top pane displays a list of captured packets. The middle pane shows the details of the selected packet (No. 58), which is an HTTP POST request to localhost:8080/calculadora.php/multiplicacao. The bottom pane shows the raw data of the selected packet, which is a JSON object: {"a":4, "b":5}.

No.	Time	Source	Destination	Protocol	Length	Info
30	65.419553842	127.0.0.1	127.0.0.1	HTTP/JSON	61	POST /calculadora.php/soma HTTP/1.1, JSON (application/json)
36	65.420336236	127.0.0.1	127.0.0.1	HTTP/JSON	66	HTTP/1.1 200 OK, JSON (application/json)
44	65.424025998	127.0.0.1	127.0.0.1	HTTP/JSON	80	POST /calculadora.php/subtracao HTTP/1.1, JSON (application/json)
50	65.424273812	127.0.0.1	127.0.0.1	HTTP/JSON	66	HTTP/1.1 200 OK, JSON (application/json)
58	65.425338826	127.0.0.1	127.0.0.1	HTTP/JSON	79	POST /calculadora.php/multiplicacao HTTP/1.1, JSON (application/json)
64	65.425572136	127.0.0.1	127.0.0.1	HTTP/JSON	66	HTTP/1.1 200 OK, JSON (application/json)
72	65.426555255	127.0.0.1	127.0.0.1	HTTP/JSON	80	POST /calculadora.php/divisao HTTP/1.1, JSON (application/json)
78	65.426784691	127.0.0.1	127.0.0.1	HTTP/JSON	66	HTTP/1.1 200 OK, JSON (application/json)
86	65.428495163	127.0.0.1	127.0.0.1	HTTP/JSON	91	POST /calculadora.php/expressao HTTP/1.1, JSON (application/json)
92	65.428729279	127.0.0.1	127.0.0.1	HTTP/JSON	66	HTTP/1.1 200 OK, JSON (application/json)
100	65.429843844	127.0.0.1	127.0.0.1	HTTP/JSON	80	POST /calculadora.php/divisao HTTP/1.1, JSON (application/json)
106	65.430151575	127.0.0.1	127.0.0.1	HTTP/JSON	66	HTTP/1.1 400 Bad Request, JSON (application/json)

Frame 58: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 49996, Dst Port: 8080, Seq: 199, Ack: 1, Len: 13
[2 Reassembled TCP Segments (211 bytes): #56(188), #58(13)]
Hypertext Transfer Protocol
POST /calculadora.php/multiplicacao HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: Java/11.0.20
Host: localhost:8080
Connection: keep-alive
Content-Length: 13
[Full request URI: http://localhost:8080/calculadora.php/multiplicacao]
[HTTP request 1/1]
[Response in frame: 64]
File Data: 13 bytes
JavaScript Object Notation: application/json
Object
{"a":4, "b":5}

6. Análise Comparativa

Acoplamento

- MQTT → totalmente desacoplado

- HTTP → acoplado (cliente precisa conhecer servidor)

Aspecto	MQTT	REST/HTTP
Overhead	~89 bytes	~463 bytes
Acoplamento	Desacoplado	Acoplado
Escalabilidade	1:N	1:1
Latência	<10ms	10-100ms

Escalabilidade

- MQTT → 1:N nativo, altamente escalável
- HTTP → 1:1, limitado para broadcast

Desempenho

- MQTT → baixa latência, baixo overhead
- HTTP → mais lento devido a headers

Interoperabilidade

- HTTP → excelente (padrão Web)
- MQTT → boa, mas depende de broker

7. Dificuldades e Soluções

- ✓ Janela temporal no CAT
- ✓ Retry diferenciado por códigos HTTP
- ✓ Uso de múltiplos clientes MQTT
- ✓ Problemas de rede no sensor real

8. Conclusão

O trabalho permitiu compreender diferenças práticas entre modelos distribuídos.

MQTT mostrou-se superior para cenários IoT, com baixo custo e alta eficiência.

HTTP/REST permanece mais adequado para APIs Web e operações CRUD.