INSTITUTO FEDERAL
Paraíba
Campus Cajazeiras

PROGRAMAÇÃO P/ WEB 2

4. COMUNICAÇÃO ASSÍNCRONA

PROF. DIEGO PESSOA

DIEGO.PESSOA@IFPB.EDU.BR

@DIEGOEP

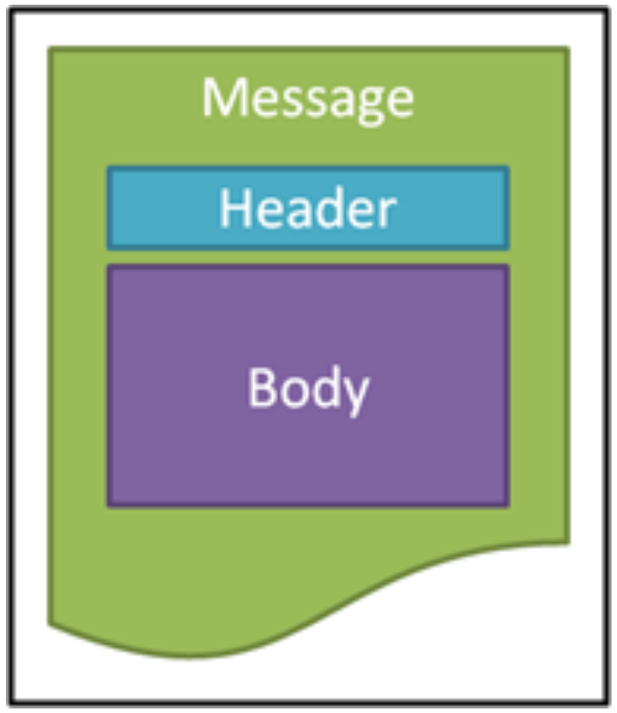
CST em Análise e Desenvolvimento de Sistemas

ENVIO DE MENSAGENS ASSÍNCRONAS

- Pode haver a presença de um *Message Broker* intermediando a requisição ou não (envio direto)
- Fluxo:
  - 1. Cliente envia mensagem para servidor
  - 2. Se a mensagem possui retorno, o servidor envia uma nova mensagem com a resposta para o cliente
- Como a comunicação é assíncrona, o cliente não fica bloqueado aguardando a resposta. Ele assume que a resposta não será imediata.

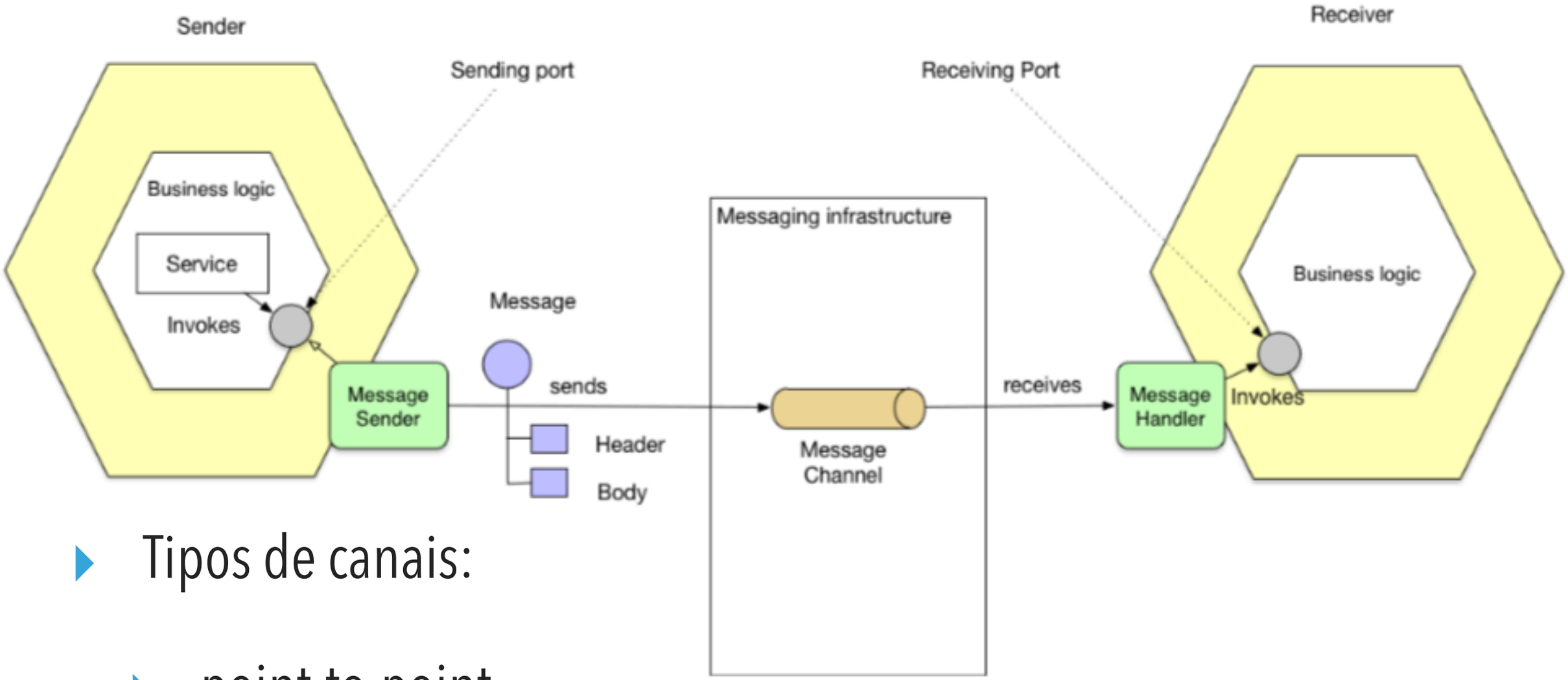
MENSAGENS

- Estrutura



- Tipos de mensagem:
  - Documento
  - Comando
  - Evento

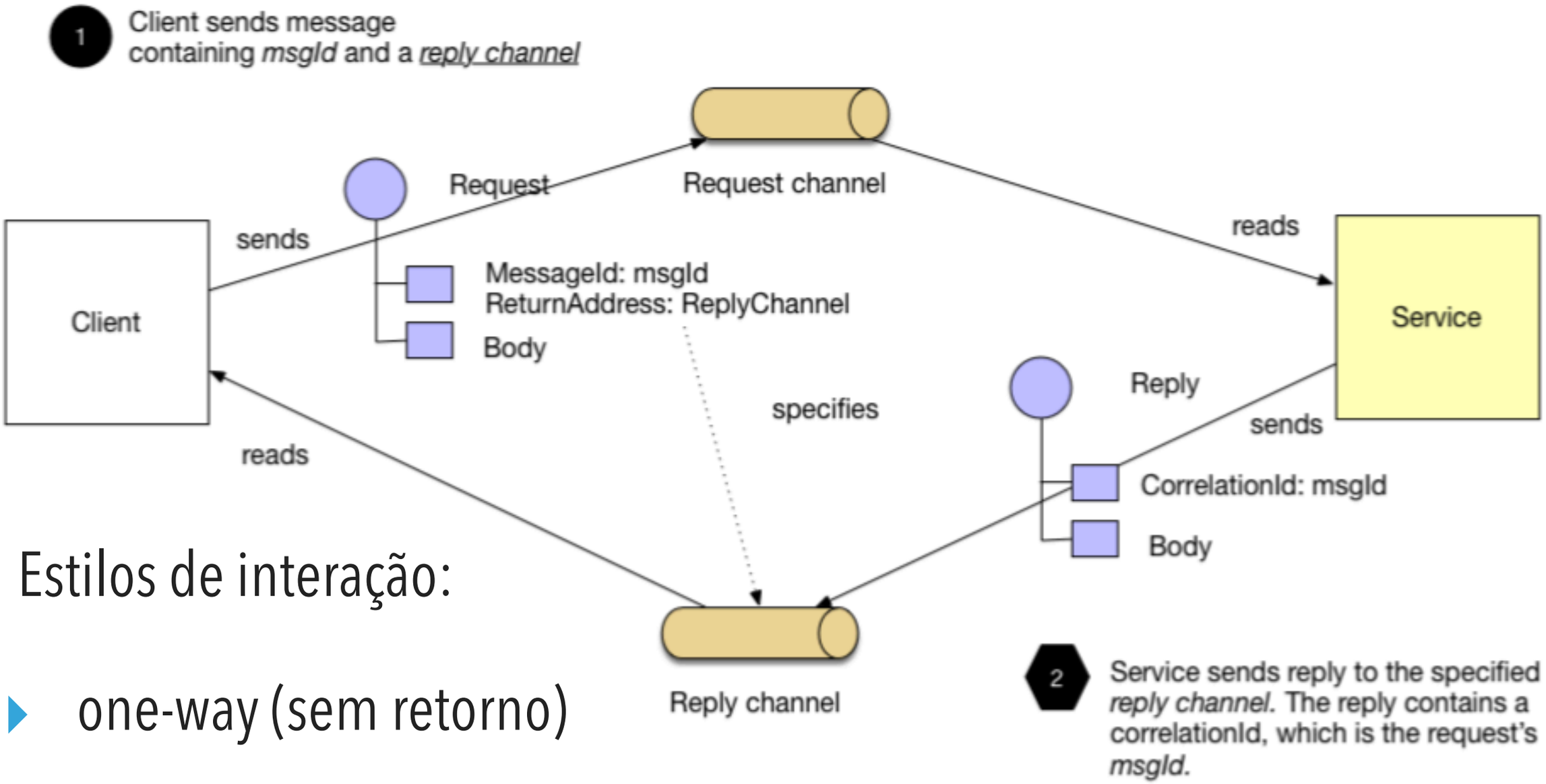
MESSAGE CHANNEL



- Tipos de canais:
  - point-to-point
  - publish-subscribe

Modelo proposto no livro "Enterprise Integration Patterns" por Gregor Hohpe

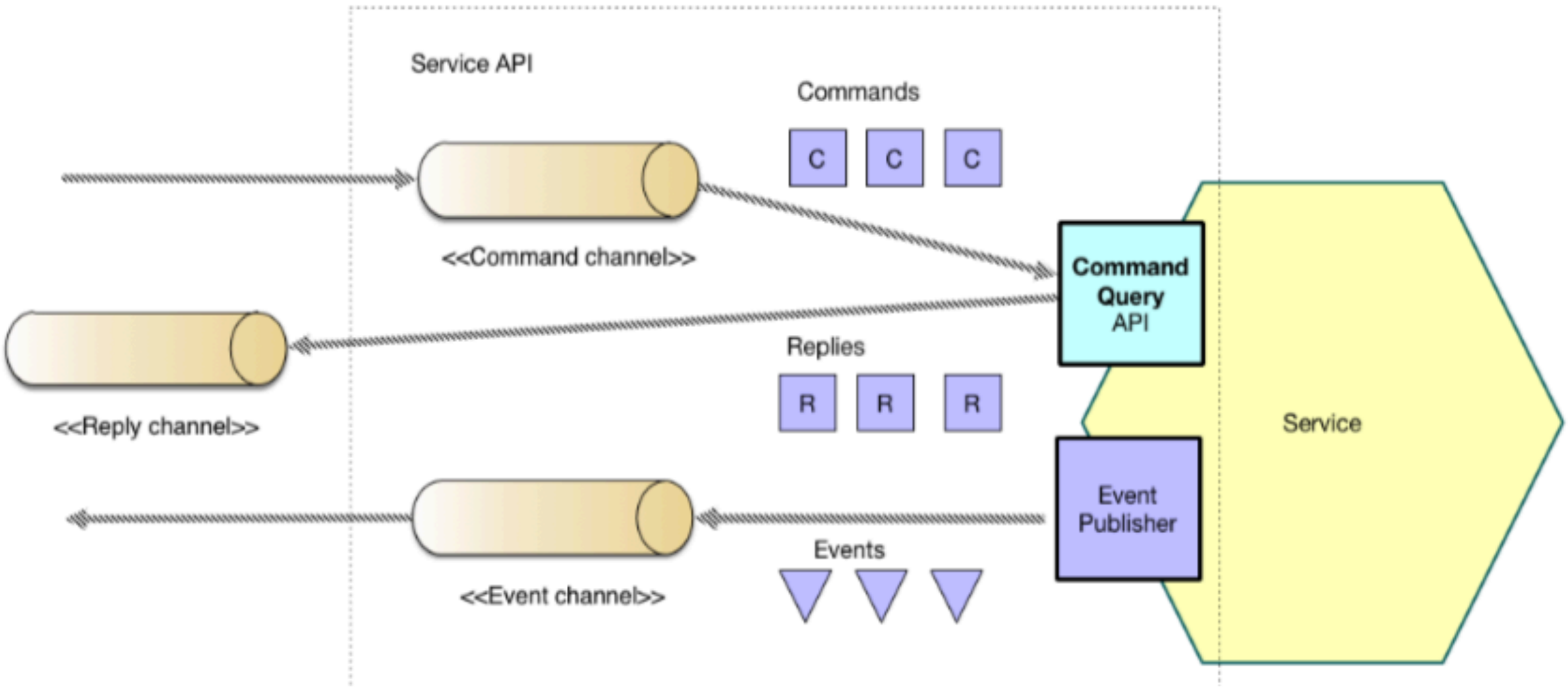
IMPLEMENTANDO OS ESTILOS DE INTERAÇÃO ATRAVÉS DE MENSAGENS



- Estilos de interação:
  - one-way (sem retorno)
  - publish/subscribe (sem resposta ou com resposta assíncrona)

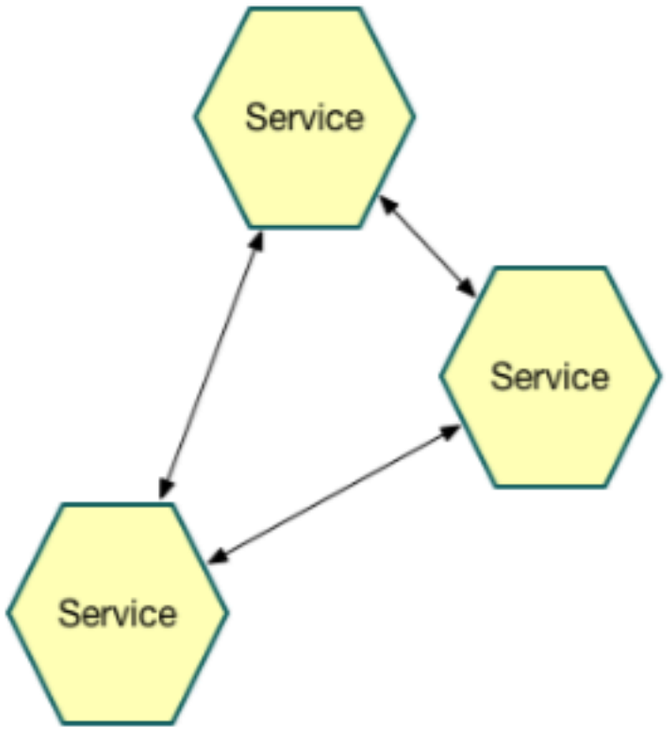
CRIANDO UMA API PARA UM SERVIÇO BASEADO EM MENSAGENS

- Documentar operações e documentar eventos



USANDO UM MESSAGE BROKER

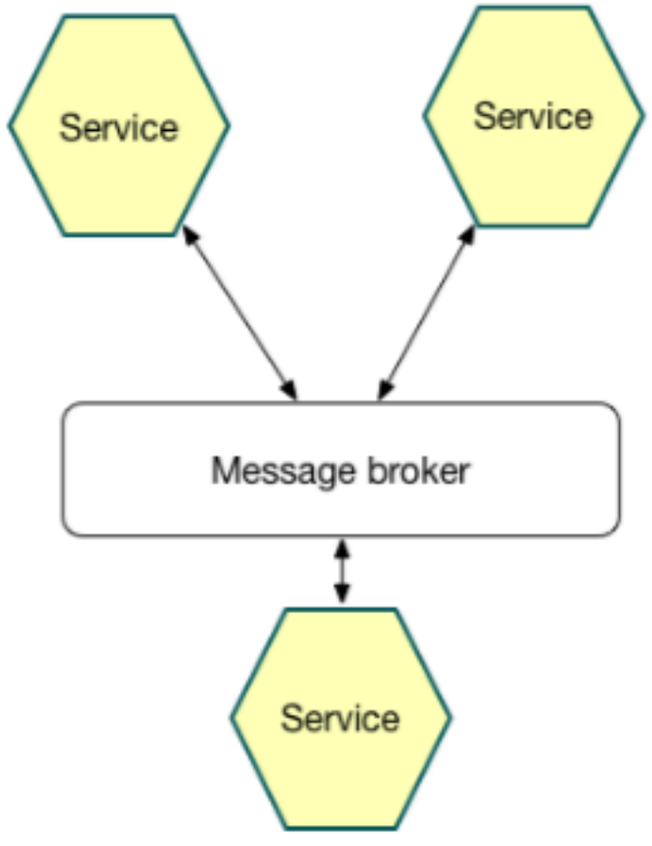
Brokerless architecture



Ex.: ZeroMQ

Vs.

Broker-based architecture



Ex.:
ActiveMQ, RabbitMQ (local)
AWS Kinesis, AWS SQS (nuvem)

BROKERLESS (SEM BROKER)

- Vantagens:
  - Menos tráfego na rede e melhor latência
  - Elimina o ponto único de falha (broker)
  - Menor complexidade (não há broker para manter)
- Desvantagens:
  - Serviços precisam conhecer um ao outro e usar mecanismos de descoberta (ex.: Service Discovery)
  - Disponibilidade reduzida (ambos os pontos precisam estar on-line)
  - Dificuldade para implementar mecanismos como a garantia de entrega

BROKER-BASED

- Vantagens:
  - Menor acoplamento (um serviço não precisa conhecer o destino)
  - Buffering de mensagens (os serviços não precisam estar disponíveis)
  - Comunicação flexível (suporta todos os estilos de interação)
- Desvantagens:
  - Gargalo de desempenho no broker (tudo passa por ele)
  - Broker precisa ter alta disponibilidade (ponto único de falha)
  - Complexidade operacional adicional

TECNOLOGIAS BROKED-BASED

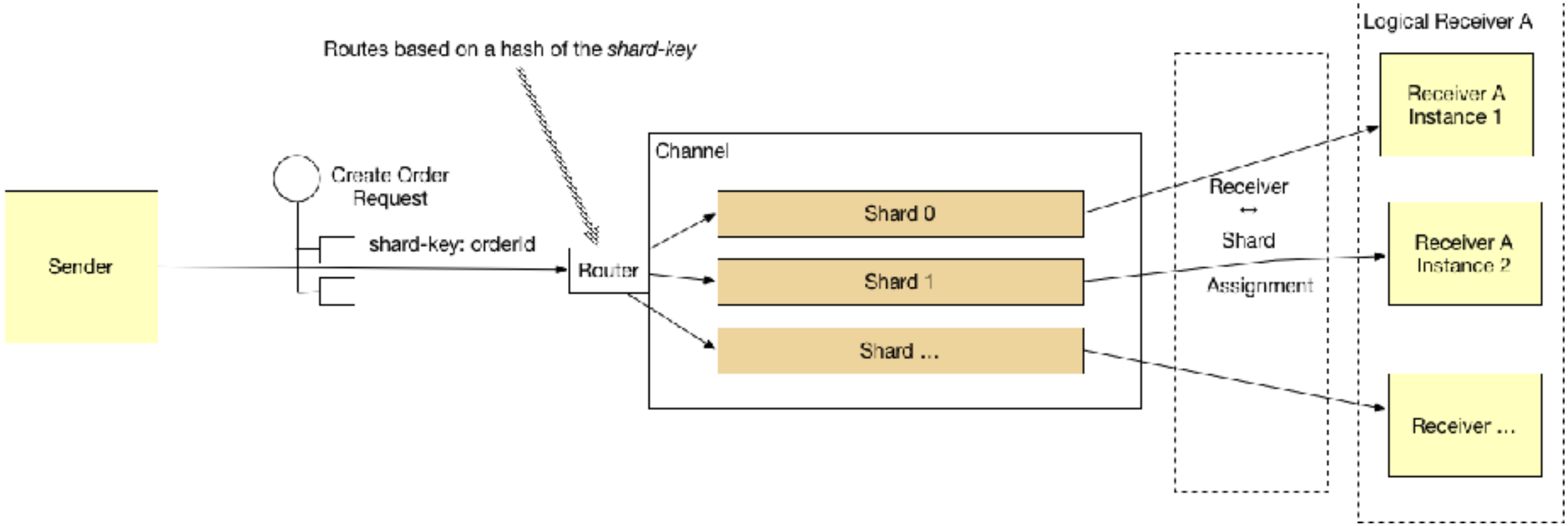
	Point-to-point channel	**Publish-subscribe channel**
JMS	Queue	Topic
Apache Kafka	Topic	Topic
AMQP-based brokers, such as RabbitMQ	Exchange + Queue	Fanout exchange and a queue per consumer
AWS Kinesis	Stream	Stream
AWS SQS	Queue	-

TECNOLOGIAS BROKED-BASED – QUAL ESCOLHER?

- **Fatores a considerar:**
- Linguagens de programação suportadas
- Suporte a protocolos padrões, como AMQP e STOMP ou é proprietário
- Ordenação de mensagens - preserva a ordem?
- Garantia de entrega - que tipo de garantia de entrega é oferecida?
- Persistência - as mensagens são persistidas no disco e são mantidas caso o broker quebre?
- Durabilidade - se um consumidor se reconecta ao broker, ele receberá as mensagens enviadas enquanto estava desconectado?
- Escalabilidade - o quão escalável é um message broker?
  Latência - qual é a latência fim a fim
- Consumidores Concorrentes - Como o broker trata a concorrência entre consumidores?

BROKER-BASED – DESAFIOS:

- Concorrência e ordenação de mensagens



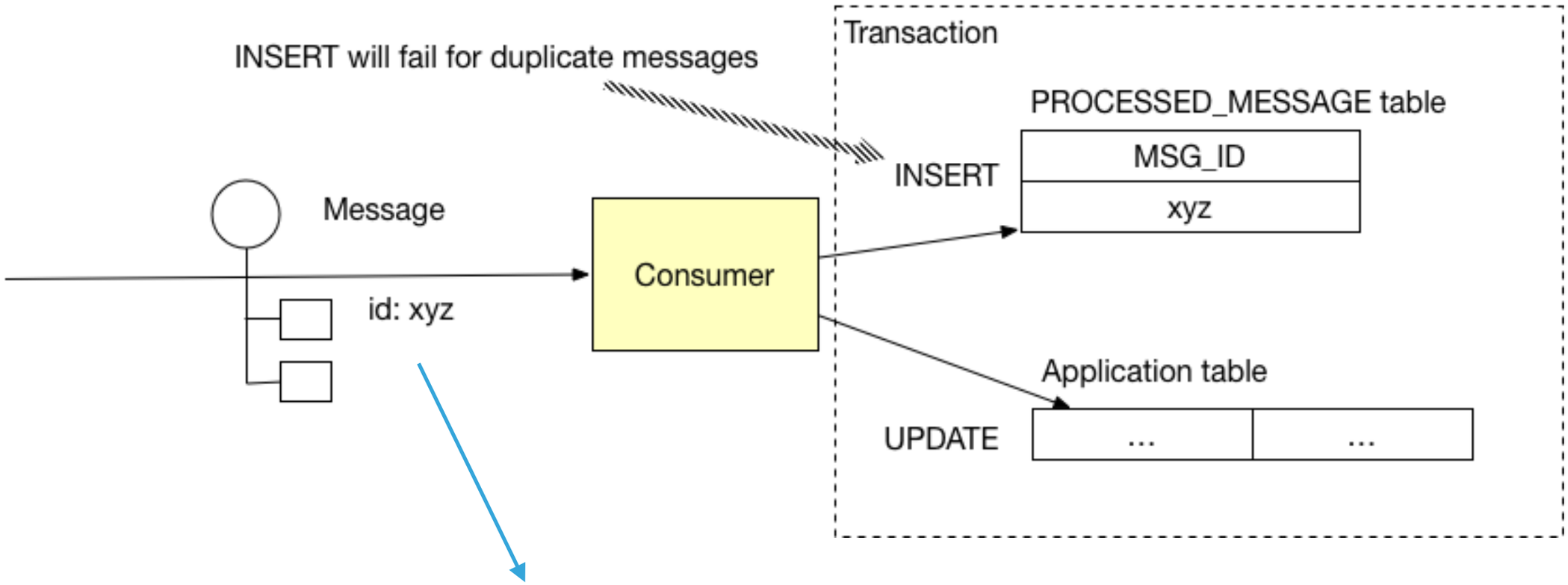
- Shards= canal particionado

BROKER-BASED – DESAFIOS:

- Tratando mensagens duplicadas
  - Broker deve entregar a mensagem apenas uma vez
  - Mas isso é custoso, então os brokers prometem entregar ao menos uma vez
- Estratégias para lidar com mensagens duplicadas:
  - Escrever tratadores de mensagens "idempotentes" (pode enviar quantas mensagens duplicadas for que ele não irá quebrar)
  - Monitorar mensagens e descartar duplicidades

BROKER-BASED – DESAFIOS:

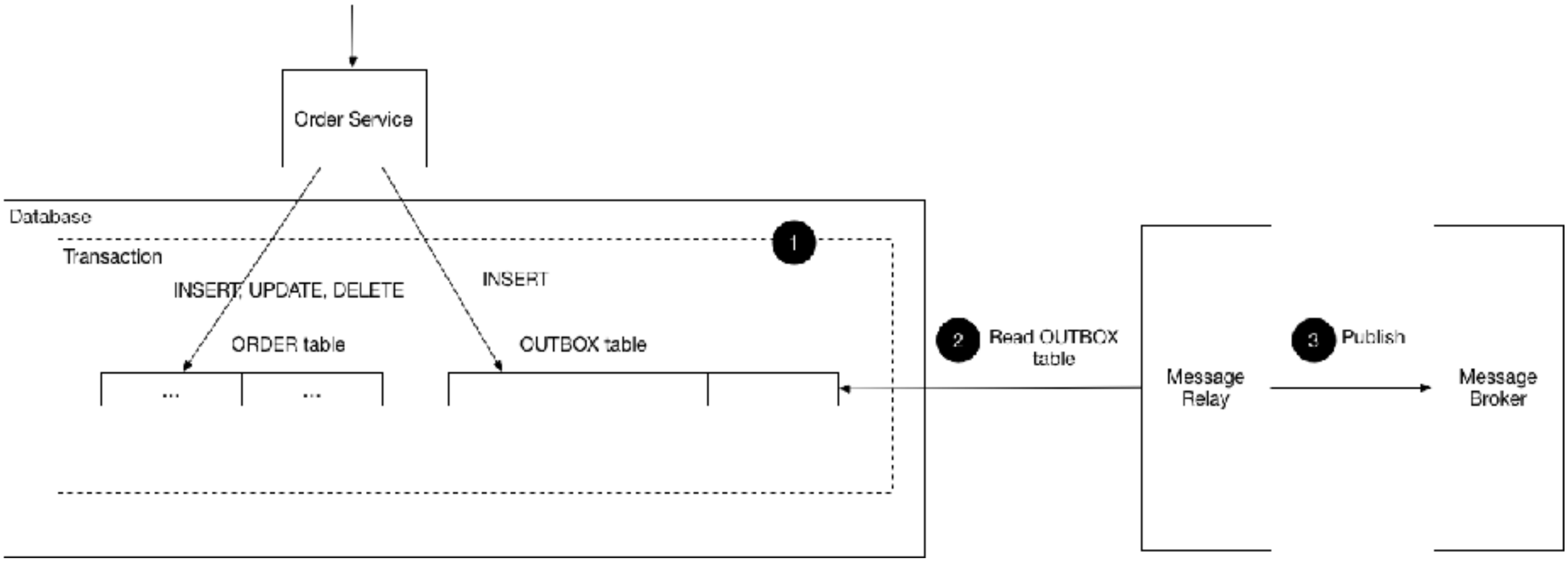
- Monitorar mensagens e descartar duplicatas



Mensagem com o ID da entidade

MENSAGENS TRANSACIONAIS

- Usando banco de dados como fila de mensagens.



MENSAGENS TRANSACIONAIS

PADRÃO TRANSACTIONAL OUTBOX
Publish an event or message as part of a database transaction by saving it in an *outbox* in the database. See http://microservices.io/patterns/data/transactional-outbox.html

- 1. Ler periodicamente a OUTBOX

  SELECT * FROM OUTBOX ORDERED BY ... ASC

- 2. Publicar as mensagens em um Message Broker

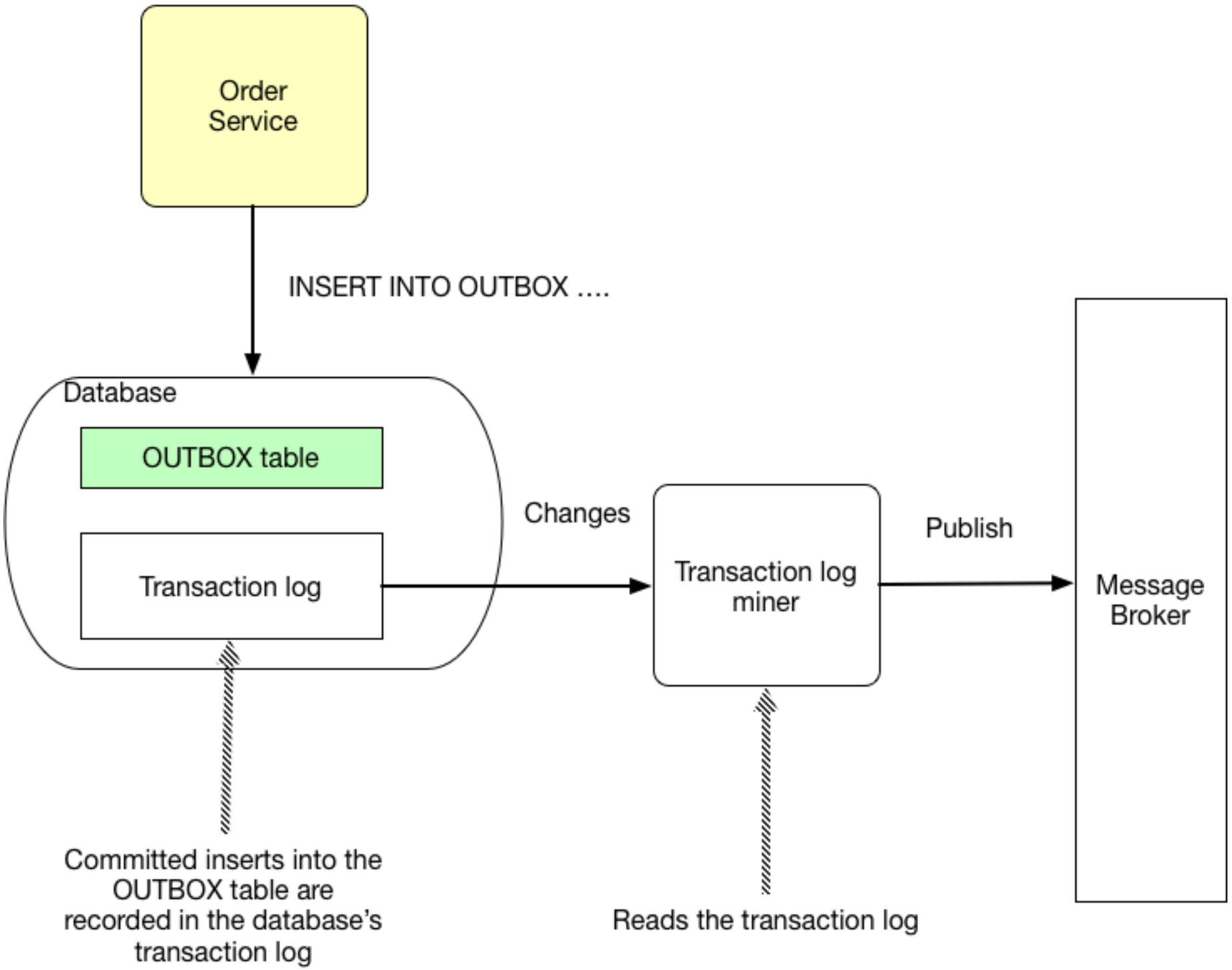
- 3. Limpar a OUTBOX

```
BEGIN
 DELETE FROM OUTBOX WHERE ID in (....)
COMMIT
```

ALTERNATIVA:

PATTERN: POLLING PUBLISHER
Publish messages by polling the outbox in the database. See http://microservices.io/patterns/data/polling-publisher.html

PUBLICANDO EVENTOS APLICANDO O PADRÃO LOG TAILING



EXEMPLOS DE TECNOLOGIAS QUE APLICAM LOG TAILING

- Debezium
- LinkedIn Databus
- DynamoDB streams
- Eventuate Tram