Mensageria

Kiev Gama

kiev.gama@gmail.com

@kievgama

slideshare.net/kievgama

Objetivos

- **Entender a motivação e o funcionamento de comunicaçao indireta e mensageria**

Princípio fundamental: Fraco acoplamento

- **Reduzir o número de suposições entre duas partes comunicantes**
- **Exemplo de acoplamento forte: chamada de método local**
  - Suposições sobre
    - tipo/formato dos dados
    - localização dos componentes chamados
    - tempo de resposta
  - Suposições facilitam o desenvolvimento, mas limitam escalabilidade e capacidade de mudanças
- **RPC**
  - Reduz acoplamento da chamada local
  - Não faz suposições sobre quais componentes serão chamados
  - Entretanto, introduz problemas inerentes da comunicação distribuída
    - Latência
    - Muitas vezes inadequado para comunicação síncrona
- **Apelo para chamadas baseadas em mensagem**
  - Assíncronas
  - Menor acoplamento que RPC
  - Minimiza problemas da comunicação distribuída
  - Garantia de entrega de mensagens

Acoplamento Forte x Fraco

- **Forte**
  - Representação interna dos dados é conhecida
  - Necessidade de conhecer o receptor
  - Dependência temporal
  - E se uma das partes falhar?
  - Como lidar c/ vários receptores?

Acoplamento Forte x Fraco

- **Forte**
  - Representação interna dos dados é conhecida
  - Necessidade de conhecer o receptor
  - Dependência temporal
  - E se uma das partes falhar?
  - Como lidar c/ vários receptores?

- **Fraco**
  - Nível de indireção acrescentado
  - Dados auto descritivo e independentes de plataforma (ex: XML)
  - Envio para canal de comunicação ao invés de máquina específica
  - Canal de comunicação pode ser armazenar dados, ser interceptado (disparar eventos, transformar dados da mensagem, notificar terceiros, etc)

Aumento de complexidade

- **Trade-off de soluções com baixo acoplamento**
  - Vantagens como flexibilidade e escalabilidade
  - Modelo de programação mais complexo
- **Maiores dificuldades para**
  - Projetar
  - Implementar
  - Depurar
- **Necessidade de middleware específico**

Comunicação Indireta

- **Problemas em computação são resolvidos com indireção**
- **Utiliza um mediador para a comunicação**
- **Comunicação indireta em sistemas distribuídos é evita o acoplamento entre *sender* (emissor) e *receiver* (receptor)**
- **Promove desacoplamento de**
  - Tempo: emissor e receptor possuem tempos de vida independentes
  - Espaço

Comunicação Indireta
Abordagens

- Comunicação em grupo
- Publish-subscribe
- Filas de mensagem

Relembrando

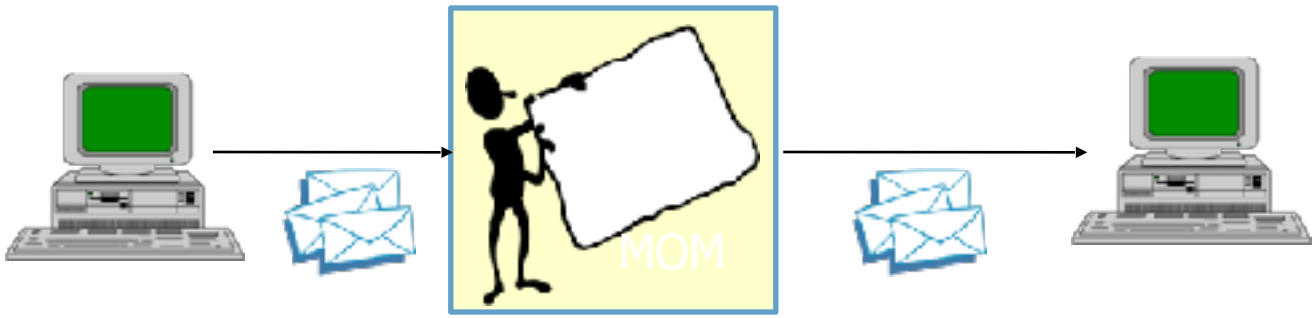
- **Padrões de comunicação em sistemas distribuídos**
  - Orientado a request: transmissor envia mensagem e espera uma resposta do receptor
  - **Orientado a mensagem**: transmissor envia mensagem e não espera resposta

Message-based communication

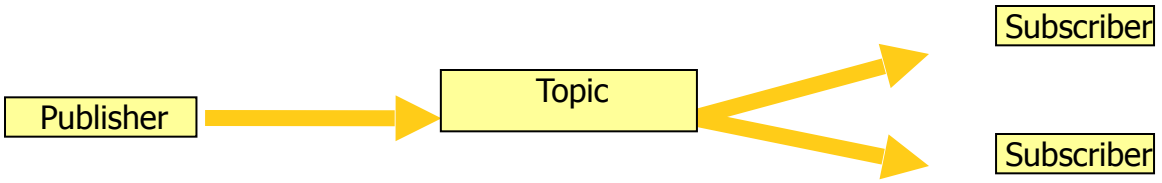
- **O que é uma mensagem?**
  - Unidade de informação bem definida e isolada
  - Transmitida entre sistemas comunicantes, com o propósito de troca de informação
  - Exemplos:
    - Um email em uma aplicação de email
    - Uma notificação de evento em uma apliação orientada a eventos
    - Uma requisição ou uma resposta em uma aplicação baseada em RPC

Mensageria

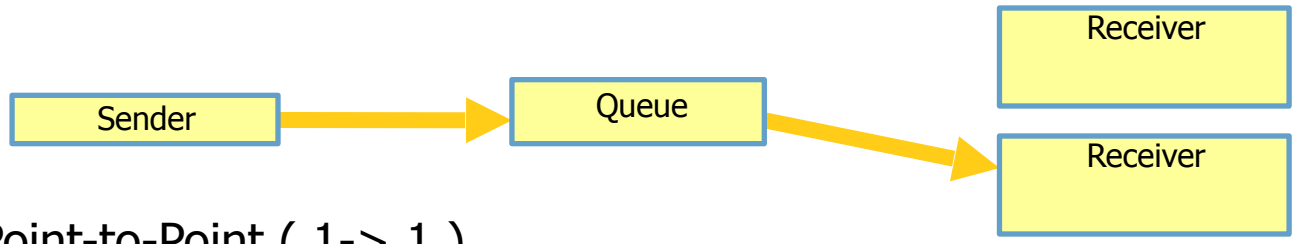
- Consiste em um modelo no qual aplicações são fracamente acopladas através da troca de mensagens autodescritivas



Modelos de Mensageria



Publish/Subscribe (1-> many)
(pub/sub)

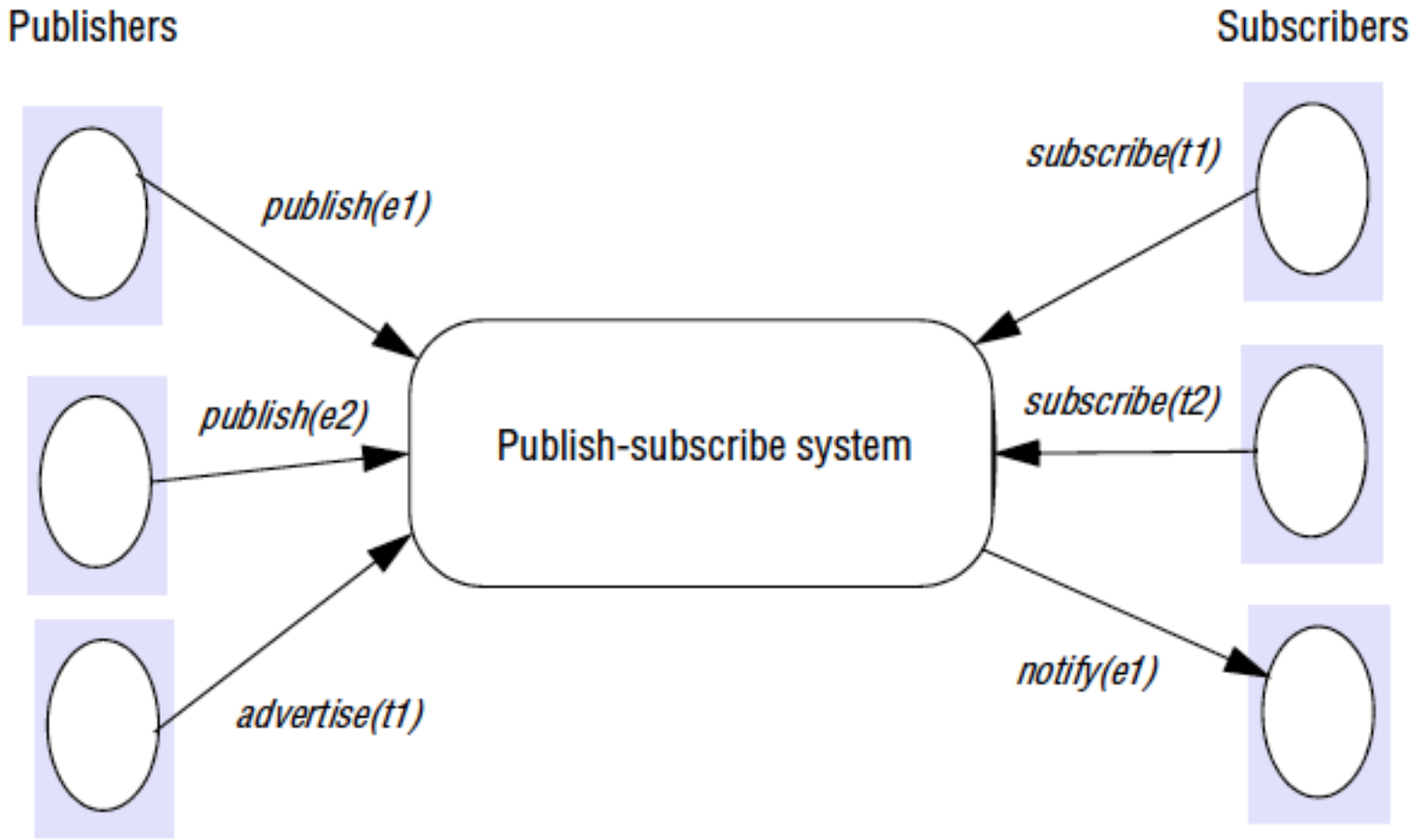


Point-to-Point (1-> 1)
(p2p, PTP)

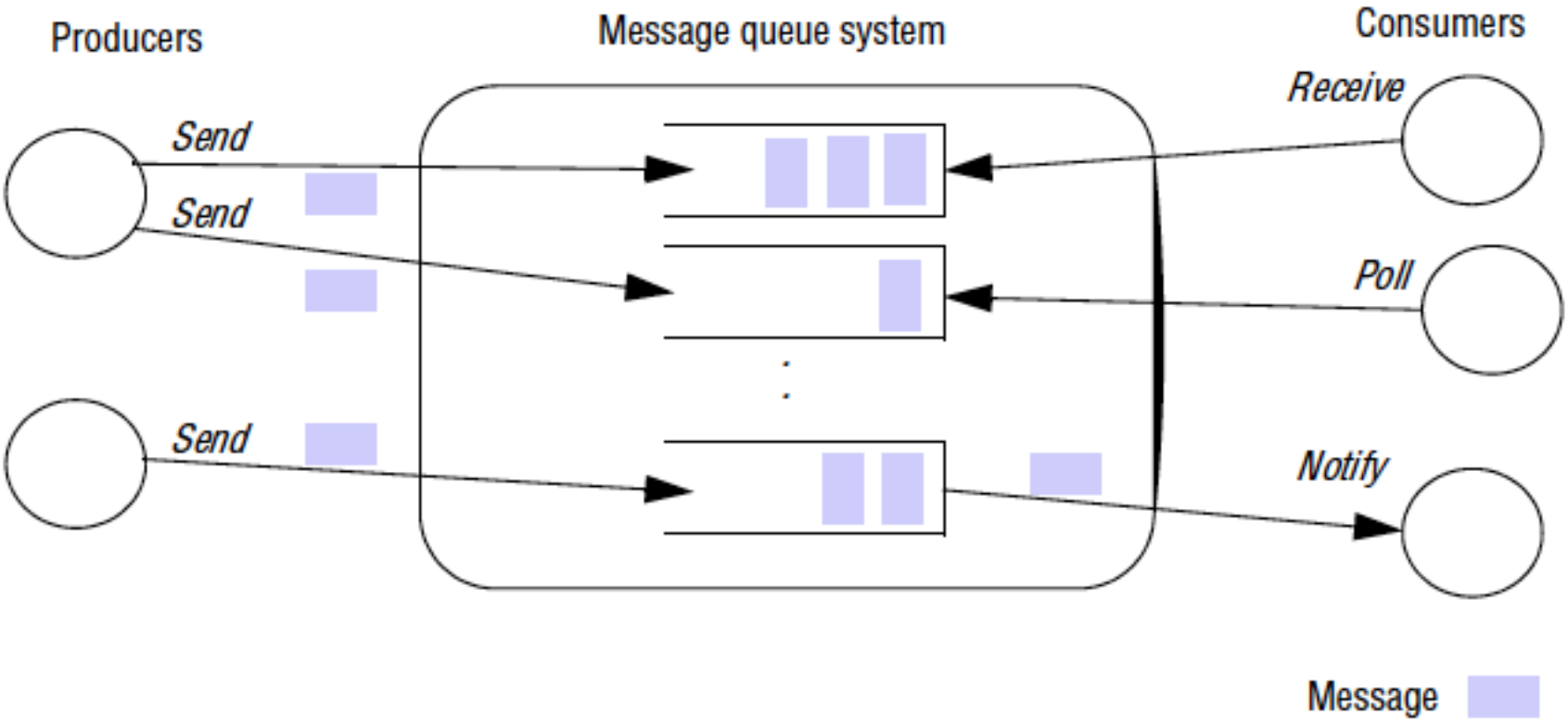
Publish/Subscribe

- **Modelo de comunicação fracamente acoplado**
- **Aplicação emissora publica uma mensagem rotulada ("label") e a torna disponível a todos receptores interessados**
- **Terminologia:**
  - Aplicação emissora da mensagem: *publisher*
  - Receptores interessados: *subscribers*
- **O emissor (*publisher)* não precisa conhecer nem explicitamente indicar os receptores pretendidos para uma mensagem**
- **Um receptor (*subscriber*), deve receber apenas mensagens para as quais ele se registrou previamente (ex: mensagem com o rótulo que o interessa)**

Publish / subscribe



Message Queue System



Estilos de interação

- **Dois principais modos de consumo pelo receptor:**
  - Bloqueante (síncrono, pull mode):

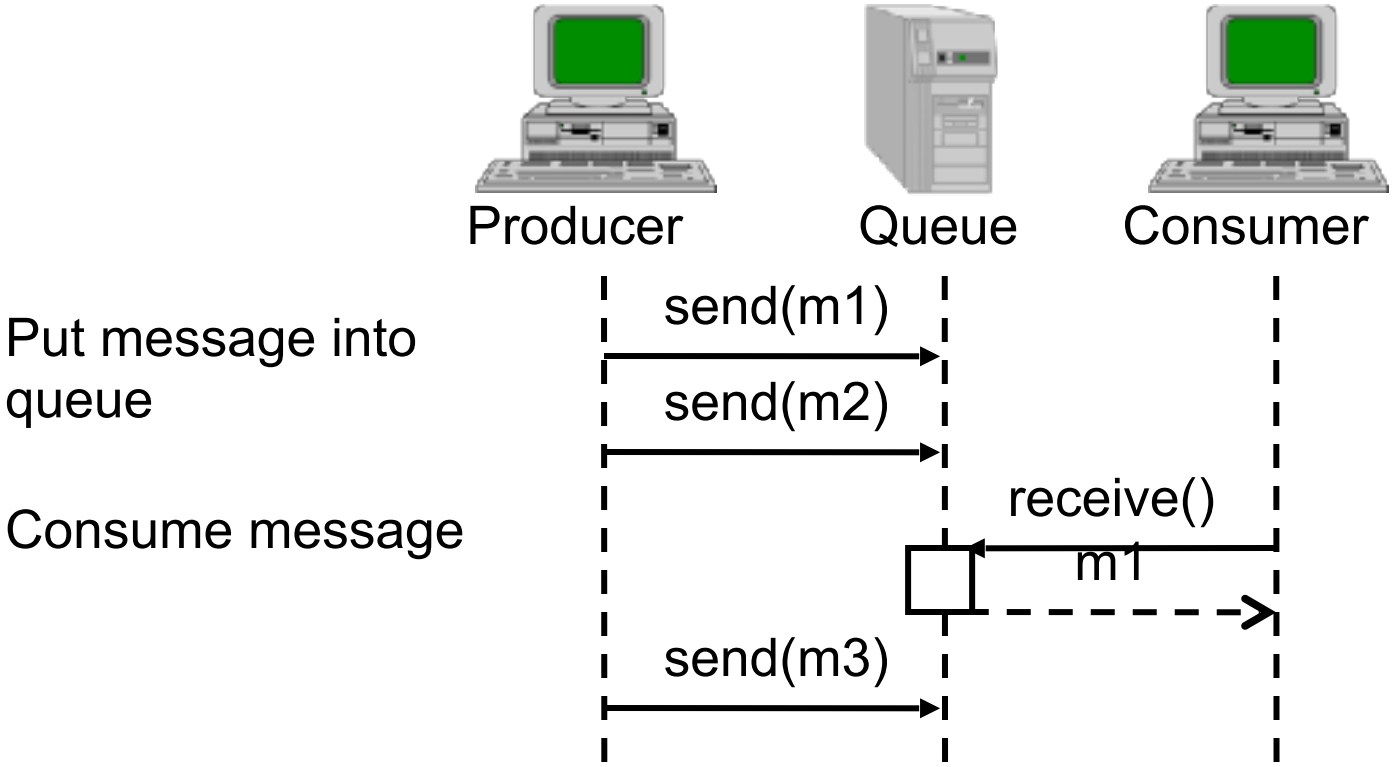
    MessageConsumer.receive()
  - Não bloqueante (assíncrono, push mode):

    MessageListener.onMessage(..)

X

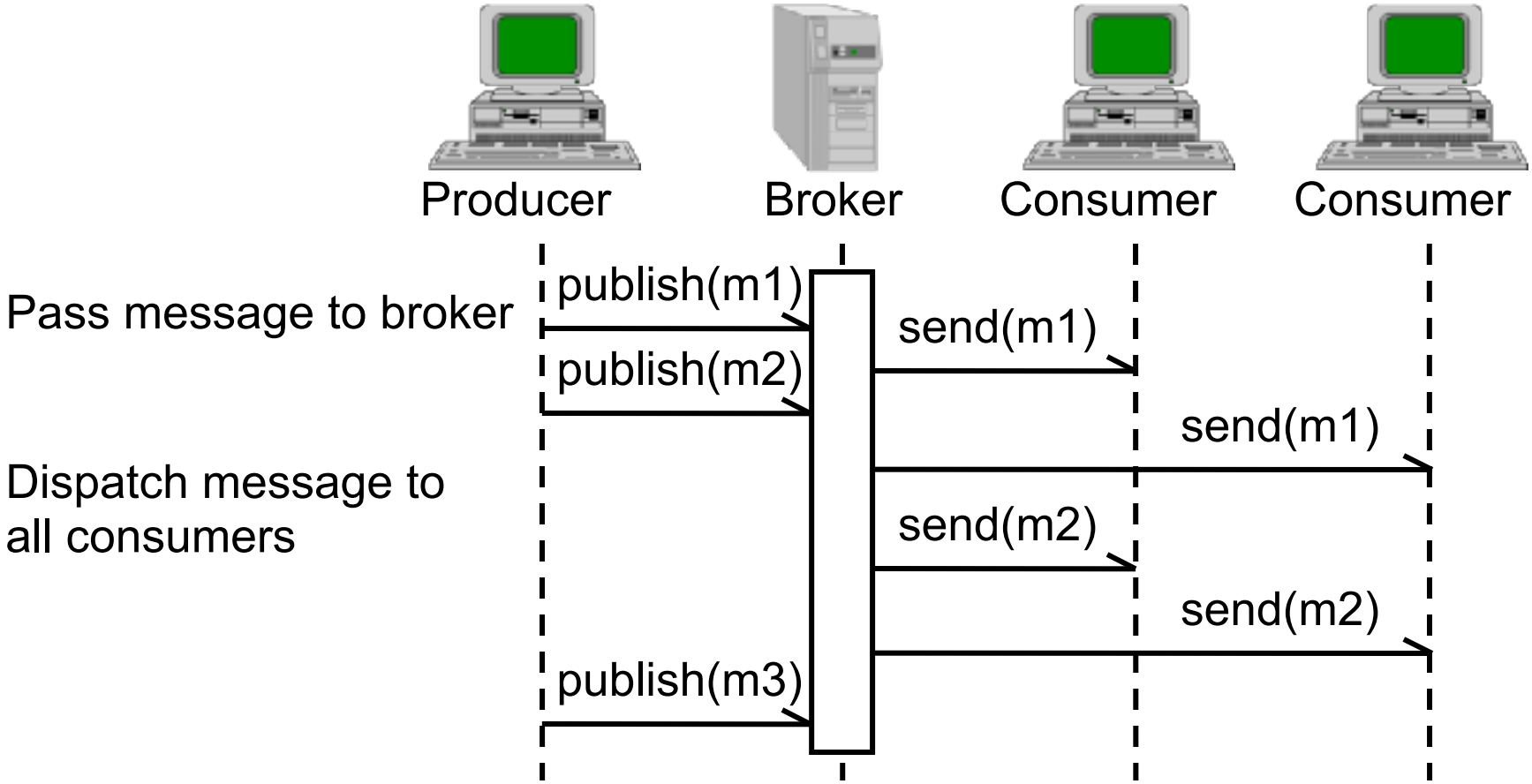
Pull

- **Recepção Ativa**
- **Tipicamente usado com filas point-to-point**



Push

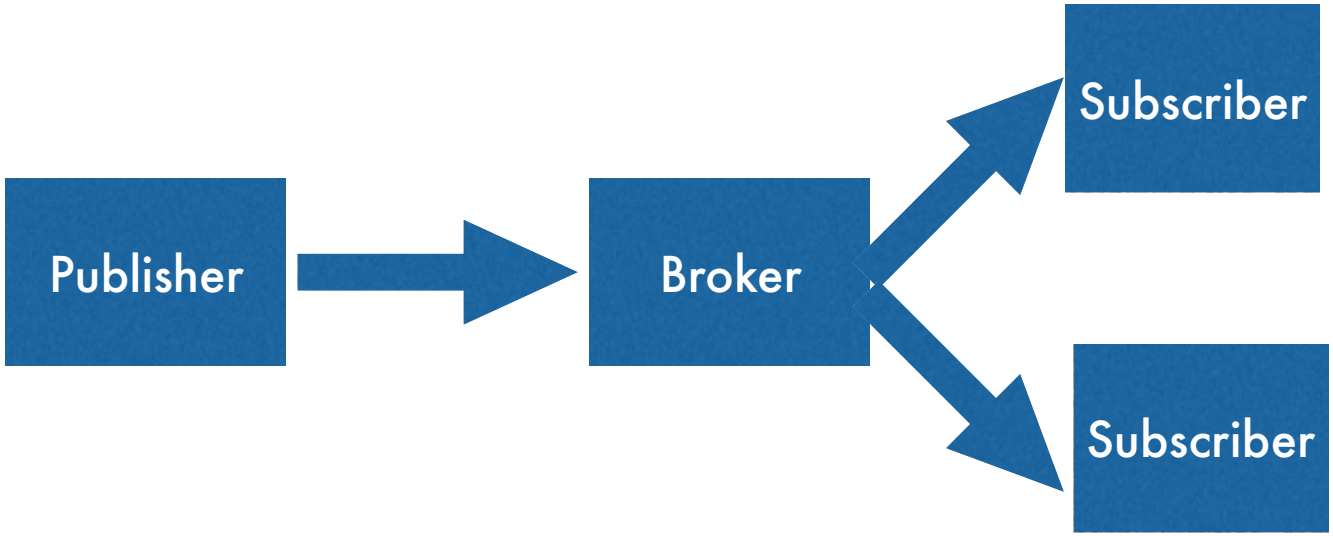
- **Recepção passiva**
- **Tipicamente usado com comunicação 1-n (Publish/Subscribe)**



Mensageria no nosso contexto:

Message Queue Telemetry Transport

MQTT

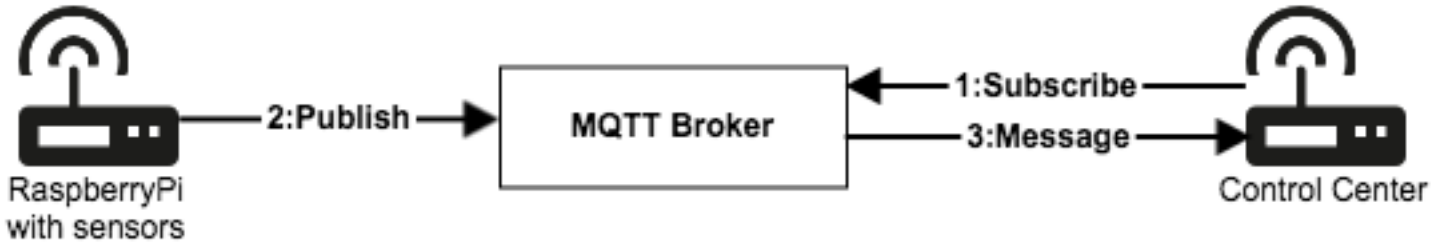


É um protocolo de conectividade para a M2M/ IoT

Concebido como um mecanismo publish/subscribe extremamente leve

Alguns brokers suportam níveis de QoS:

Fire and forget, deliver at least once, deliver exactly once



Connect

Disconnect

Subscribe

UnSubscribe

Publish

Free brokers

Server	Broker	Port	Websocket
iot.eclipse.org	Mosquitto	1883 / 8883	n/a
broker.hivemq.com	HiveMQ	1883	8000
test.mosquitto.org	Mosquitto	1883 / 8883 / 8884	8080 / 8081
test.mosca.io	mosca	1883	80
broker.mqttdashboard.com	HiveMQ	1883	

Resumindo

Mecanismos de mensageria fornecem boa escalabilidade por serem assíncronos, mas dependem de um mediador da comunicação (broker).

Neste contexto, para a IoT o protocolo que vem sendo mais recomendado é o MQTT, que é o que iremos utilizar nesta disciplina