Comunicação *publish/subscribe*

Noemi Rodriguez

2019

Ambientes com fraco acoplamento

a medida que temos muitas coisas...

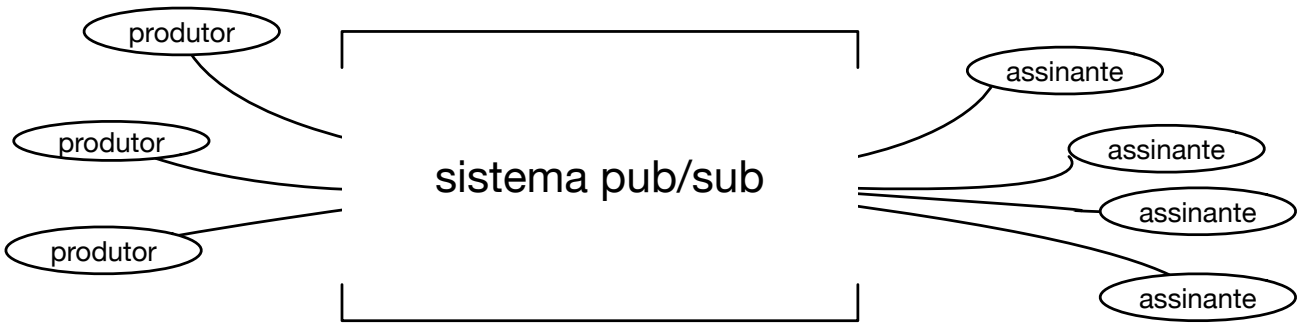
- como organizar a comunicação?

- modelos de comunicação menos p2p facilitam gerenciamento
- ... voltamos aos *canais* de comunicação

Publish/subscribe: muitos para muitos

- sistemas *pub/sub* ou sistemas de *filas de mensagens*
- registro de interesse em filas ou canais
  - modelos *push* e *pull*

- modelos de comunicação menos 1-2-1 facilitam gerenciamento


produtor
produtor
produtor
sistema pub/sub
assinante
assinante
assinante
assinante

possibilidades

- centralização de preocupações 
- garantias de segurança
  - autenticação
  - confidencialidade
- convivência com *firewalls*

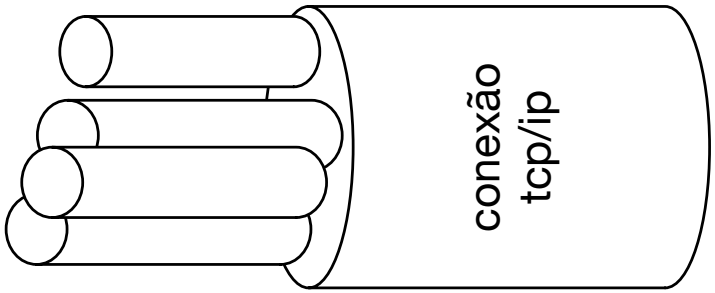
sistemas existentes

- TIBCO, IBM MQ, ...
- nuvem: sistemas de filas AWS, Azure, ...
- protocolos: AMQP, MQTT
- ênfase em disponibilidade e tolerância a falhas

Modelo de comunicação

- registro de interesse em filas
- mensagem traz apenas *payload*
  - conhecimento sobre parceiros a cargo da aplicação
- mensagens entregues em *round-robin* a registrados
- servidor só retira mensagem da fila após receber confirmação
  - quarentena e reenfileiramento
- criação de filas com nomes ou anônimas

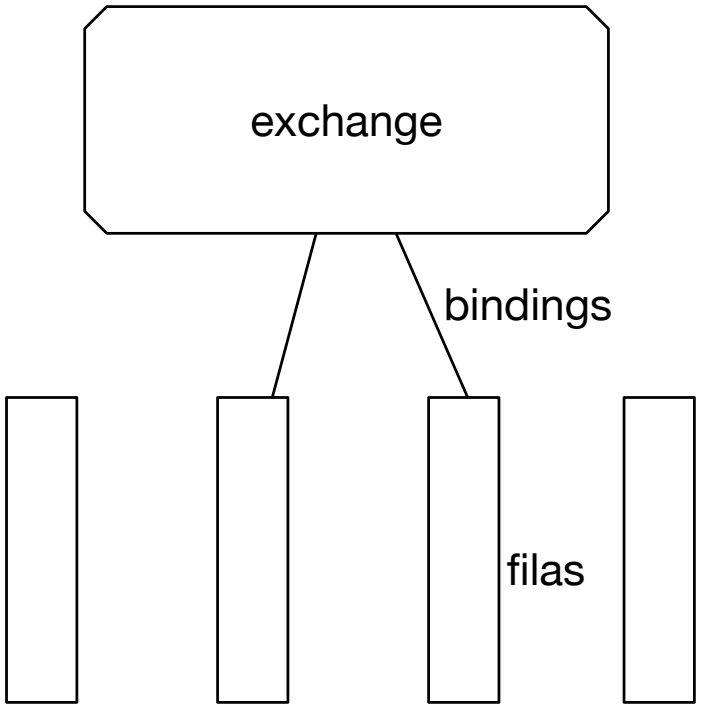
- conexões e canais



Rabbit: exchanges

- um nível de indireção a mais permite definir propriedades e forma de distribuição
- exchanges podem ser *direct*, *fanout* e *topic*
- sempre existe exchange default de nome "vazio":

$$channel- > basic\_publish(msg,'' ,' queue - name');$$

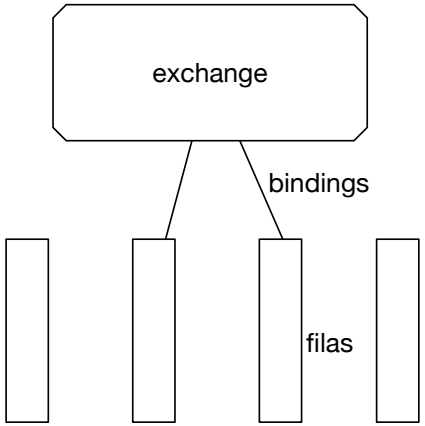


filas e confirmações

- Rabbit: mecanismos para acks e rejeição de mensagens
  - mecanismo que ajuda a construir tolerância a falhas

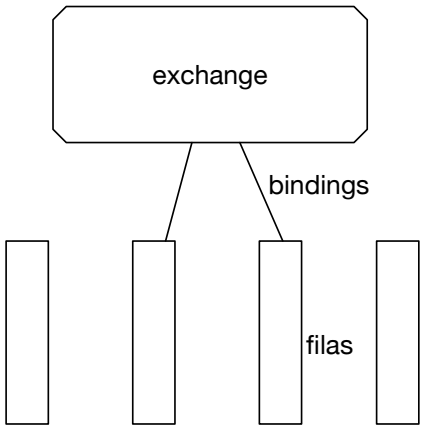
exchanges em modo direto

- *round-robin*: tipicamente distribuição de trabalho



exchanges em modo *fanout*

- replicação: facilidade de acoplar consumidores ao longo do tempo
  - *logs*, pontuação, etc



exchanges em modo *tópicos*

- distribuição conforme *chave de roteamento*

```
channel->queue_bind('msg-inbox-errors',
                    'logs-exchange', 'error.msg-inbox');
channel->queue_bind('msg-inbox-logs',
                    'logs-exchange', '*.msg-inbox');
...
channel->basic_publish(msg, 'logs-exchange', 'error.msg-inbox')
```

Persistência

- exchanges podem ser *duráveis* ou não
  - persistência de filas e bindings
- mensagens podem ser *persistentes* ou não
- custos!
  - escrita em disco

MQTT (e servidor Mosquitto)

- servidor e cliente muito mais simples
  - considerado apropriado para dispositivos de recursos limitados
- canais com replicação de mensagens para todos os inscritos

Implementação do serviço de mensagens

- escala e tolerância a falhas no serviço
- tolerância a desconexão de clientes
  - clientes móveis


PUC
RIO

Trabalho 3

- uso de MQTT para criação de jogo p2p
- uso de löve2d