UFC

DEPARTAMENTO DE COMPUTAÇÃO

MDCC

GREat

UFC VIRTUAL

Revisão Paradigmas de Comunicação

Professores:

Fernando Antonio Mota Trinta

Windson Viana de Carvalho

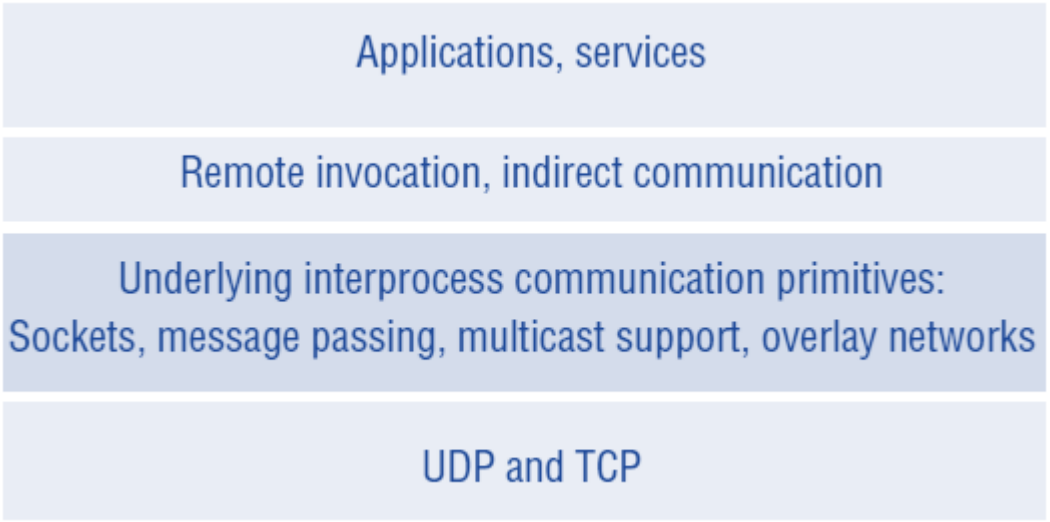
Paradigmas de comunicação

Invocação Remota

- Sockets, RPC, RMI
- Passagem de mensagens
- Request-Reply

Comunicação Indireta

- Eventos (Publish-Subscriber)
- Memória Compartilhada –Espaço de Tuplas
- Multicasting



Comunicação entre processos

- A forma mais simples é uso de APIs para acessar a camada de sockets do sistema operacional
  - Problemas inerentes a desconexões, não presença e protocolo de troca de informação ficam a cargo do programador
- Modelos de conexão atrelados ao protocolo de transporte
  - Socket TCP, socket UDP
  - Uso de Multicast ou Unicast

Comunicação entre processos

Suporte a modelo com conexão permanente e manutenção de estado de conexão

Protocolos de troca de mensagens

- Separadores
- Serialização e deserialização
- XML, JSON

TCP e UDP – Representação Externa de Dados

- Passagem de mensagens
  - Recebimento bloquante
- UDP – Não tem confirmação de erro de recebimento e nem garante ordem dos pacotes
  - Menor latência
- TCP – Garante ordem e controle de erro
  - Maior latência

TCP e UDP – Representação Externa de Dados

Exigência de uma representação externa e de transformação dos dados a serem transmitidos

- Marshalling e Unmarshalling
- XML, Serialização de Objetos, Protocolos de Middleware

Invocação remota

- Framework de mais alto nível disponível pelo Sistema Operacional ou middleware para facilitar a comunicação entre processos
  - Protocolos do tipo requisição-resposta
  - Remote Procedure Call (RPC)
  - Remote Method Invocation (RMI)

O que muda em relação a uma invocação local?

Identificadores de mensagem

Modelo de falhas do protocolo requisição-resposta

Timeouts;

Descartando mensagens de requisição duplicadas

Perda de mensagens de resposta

Histórico

Exemplo de Histórico

Nome	*Mensagens enviadas pelo*		
	Cliente	*Servidor*	*Cliente*
R	*Request*		
RR	*Request*	*Reply*	
RRA	*Request*	*Reply*	*Acknowledge reply*

Requisição-Resposta (Request-Reply)

Request-reply message structure

messageType	*int (0=Request, 1= Reply)*
requestId	*int*
remoteReference	*RemoteRef*
operationId	*int or Operation*
arguments	*// array of bytes*

doOperation bloqueia a execução até o recebimento do reply

getRequest é método invocado pelo servidor para descobrir e executar a operação

sendReply é método invocado para enviar o resultado da operação

HTTP um exemplo de Requisição-

Figure 5.6 HTTP *Request* message

method	*URL or pathname*	*HTTP version*	*headers*	*message body*
GET	http://www.dcs.qmul.ac.uk/index.html	HTTP/ 1.1		

HTTP *Reply* message

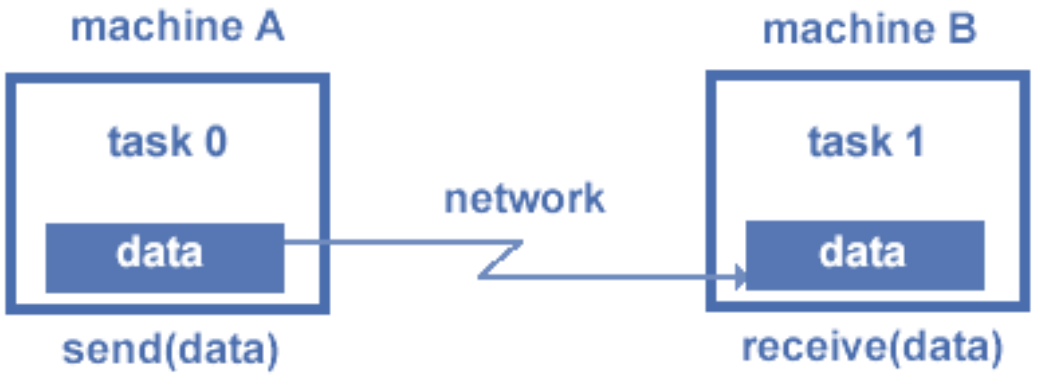
HTTP version	*status code*	*reason*	*headers*	*message body*
HTTP/1.1	200	OK		resource data

MPI-Message Passing Interface

Modelo normalmente associado a ambiente de memória distribuída;

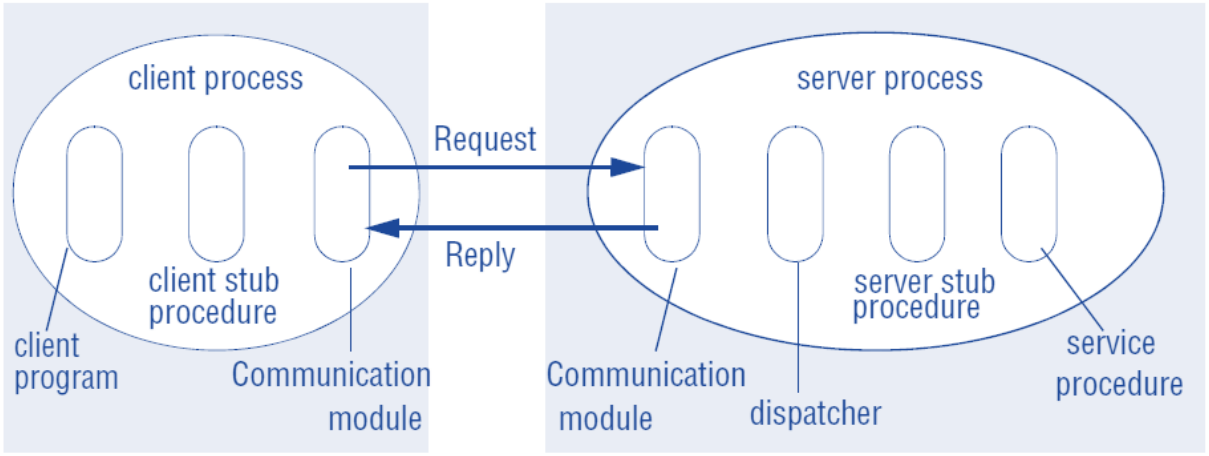
Múltiplas tarefas iniciadas e distribuídas pelos processadores do ambiente, utilizando o seu próprio endereçamento de memória;

As tarefas compartilham dados através de comunicação de envio e recebimento de mensagens (“message-passing”);



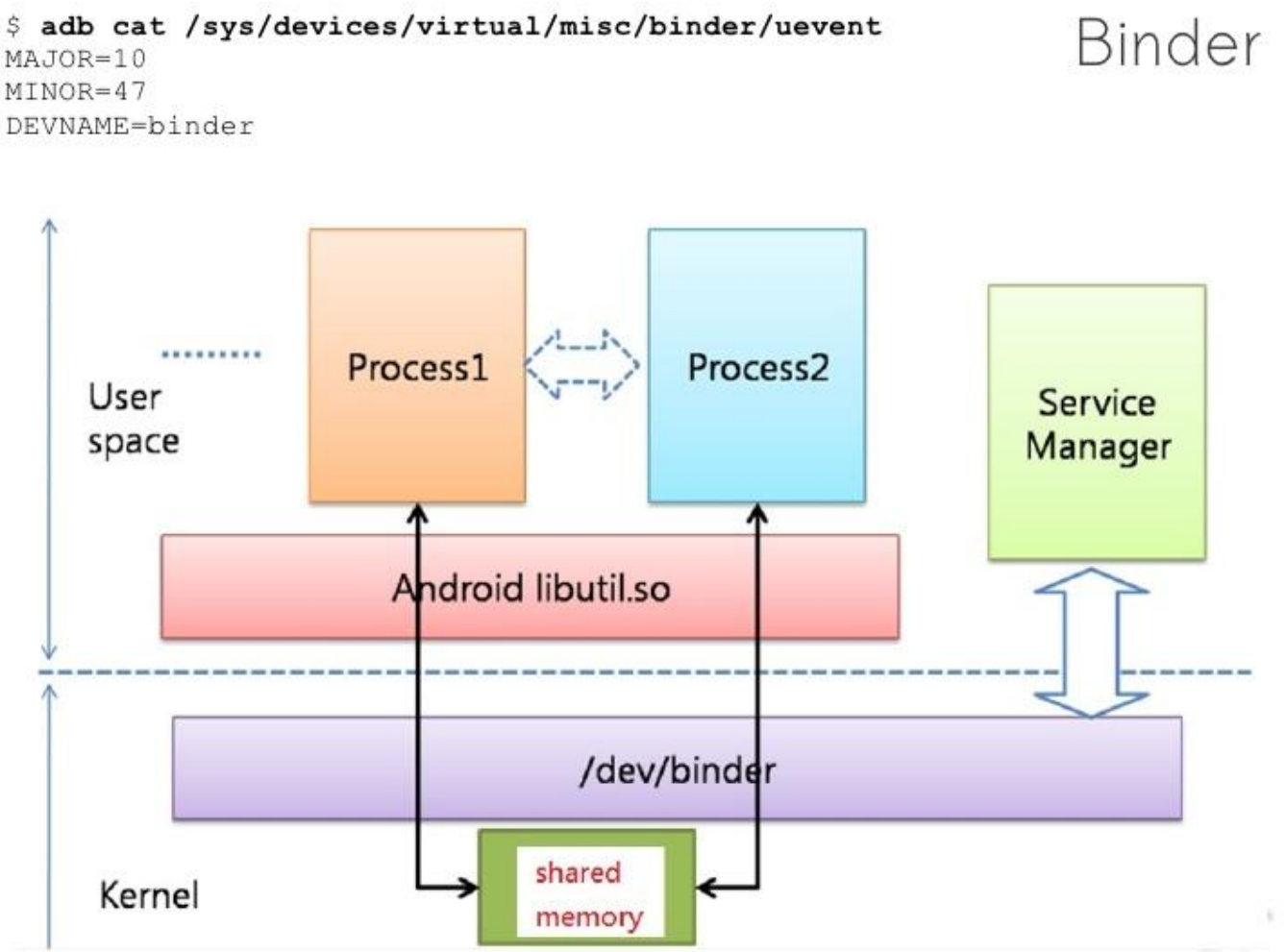
RPC – Remote Procedure Call

- Procedimento passível de invocação remota
  - Uso de uma IDL (Interface Description Language) para descrever a interface
- Referências via stubs
  - doOperation, getRequest and sendReply



Binder no Android

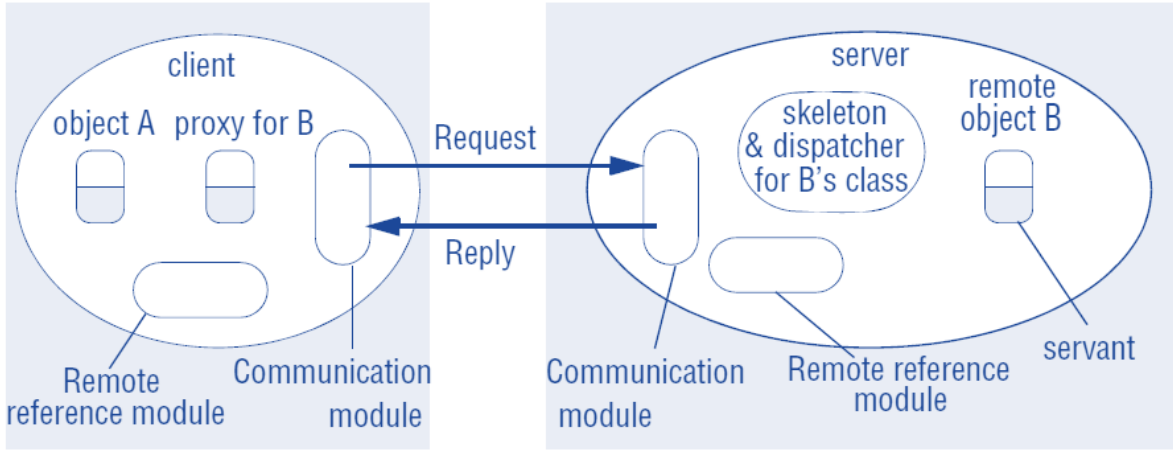
```
$ adb cat /sys/devices/virtual/misc/binder/uevent
MAJOR=10
MINOR=47
DEVNAME=binder
```



https://anatomyofandroid.com/tag/rpc/

RMI – Remote Method Invocation

- Procedimento passível de invocação remota
  - Uso de uma IDL (Interface Description Language) para descrever a interface
- Referências via stubs
  - doOperation, getRequest and sendReply



RMI – Remote Method Invocation

As invocações a métodos locais fornecem a semântica exatamente uma vez, enquanto as invocações a métodos remotos não podem garantir o mesmo.

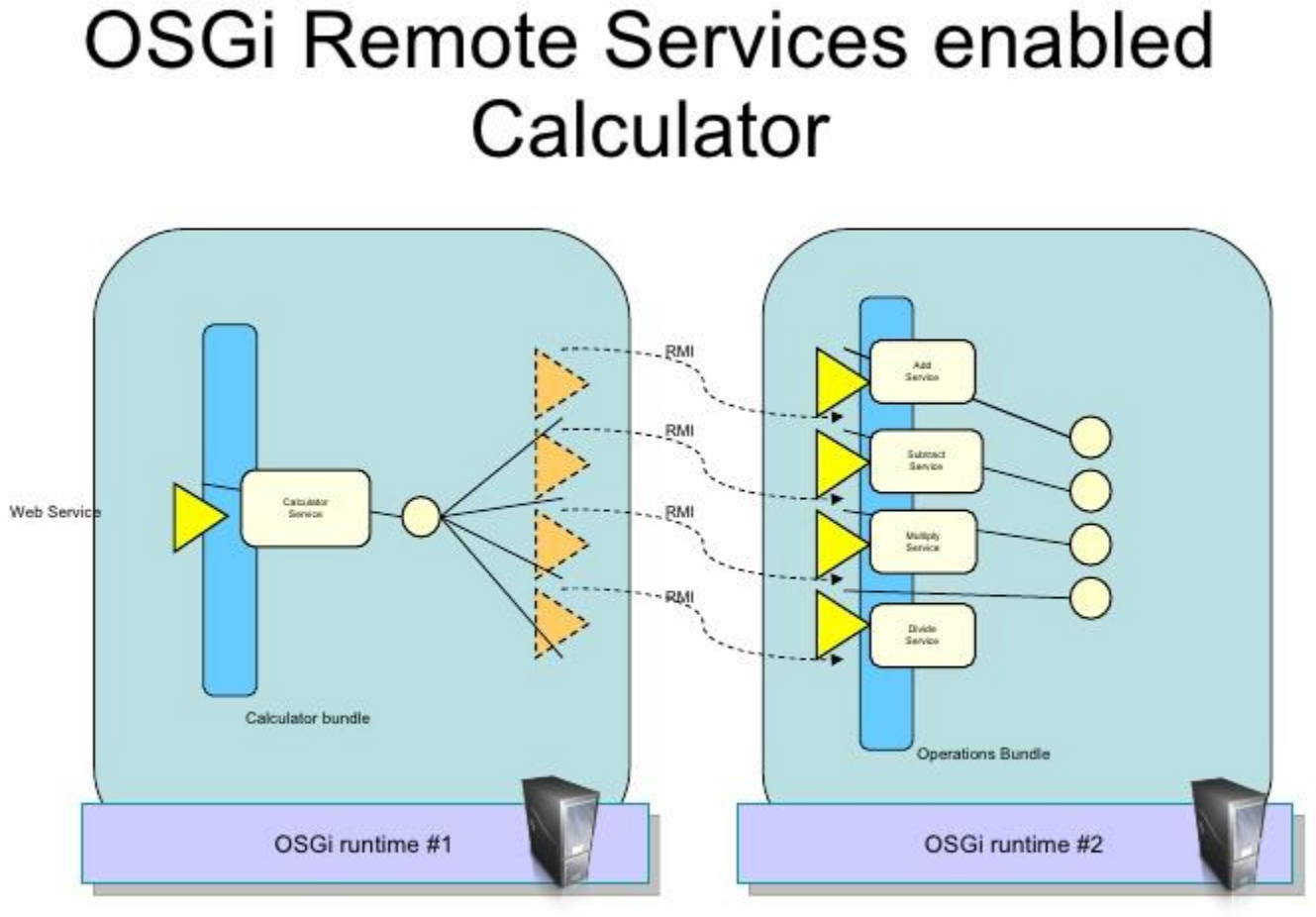
As implementações de middleware da RMI fornecem componentes

- proxies, esqueletos e despachantes

Ocultar os detalhes do empacotamento, passagem de mensagem e da localização de objetos remotos.

- Esses componentes podem se gerados por um compilador de interface IDL

Exemplo de RMI: OSGi



Paradigmas de comunicação

- Comunicação baseada em eventos X Comunicação requisição -resposta
  - Esses dois paradigmas consideram os objetos distribuídos como entidades independentes que podem se comunicar
  - No primeiro caso, um objeto em particular é invocado de forma síncrona.

Desvantagens da Invocação remota

Desvantagens da Invocação remota

- Um acoplamento entre as partes comunicantes ainda é grande
  - Conhecimento de quem está comunicando
  - Ambos conectados ao mesmo tempo
  - Pode oferecer transparência de localização e acesso usando descoberta de serviços

Comunicação indireta

- Menor acoplamento entre os sistemas
- Promove os desacoplamentos espacial e temporal
  - Sem a necessidade de conhecimento da localização e com quem está se comunicando
  - Sem a necessidade de os comunicantes estarem conectados ao mesmo

Comunicação indireta

Exemplos

- Sistemas baseados em eventos (Event-based ou Publish-Subscribers)
- Fila de Mensagens
- Espaço de Tuplas
- Memória Compartilhada e Distribuída

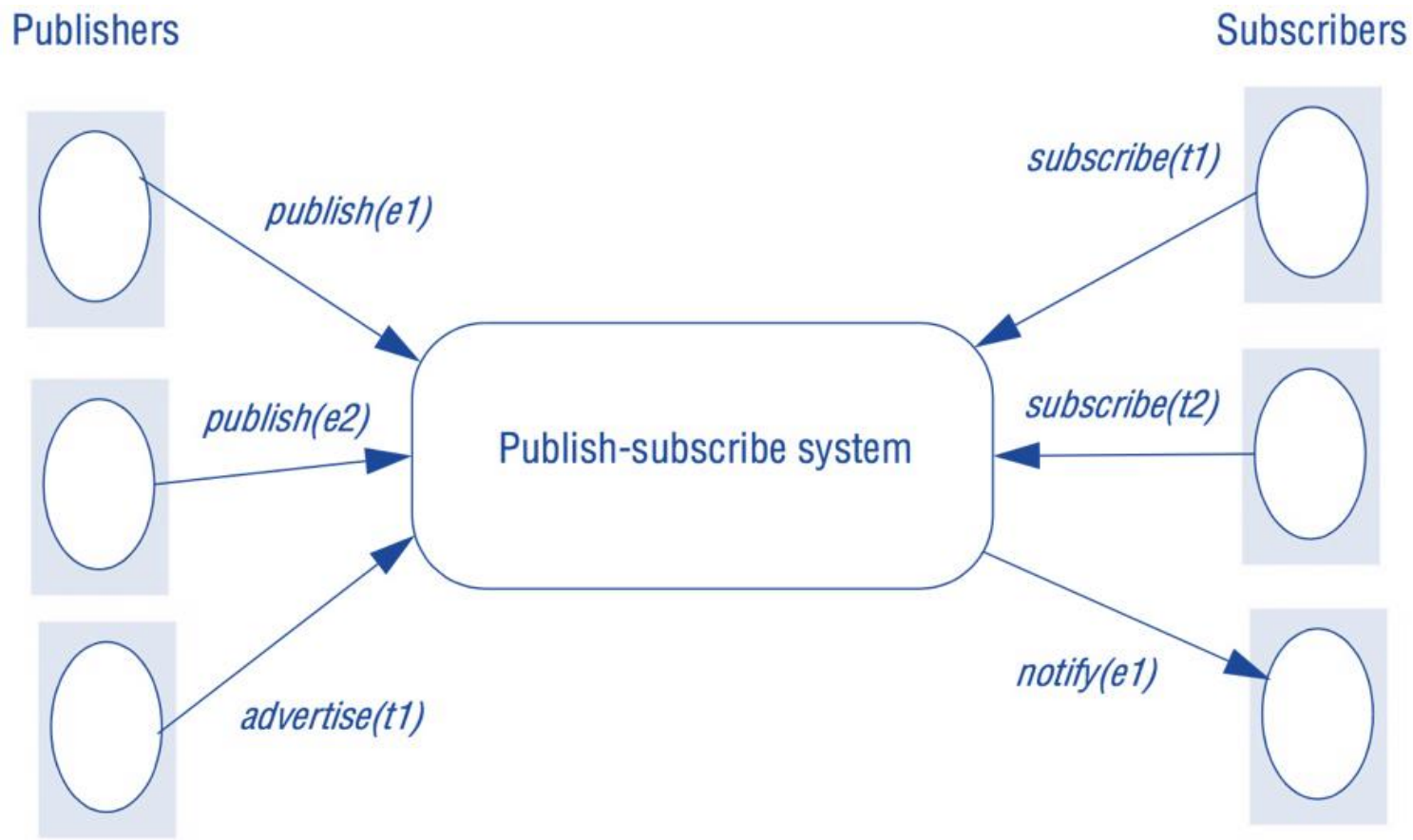
Sistemas Distribuídos baseados em Eventos

Heterogêneos: quando notificações de evento são usadas como meio de comunicação entre objetos distribuídos, os componentes de um sistema distribuído que não foram projetados para interagir podem trabalhar em conjunto.

Rede doméstica

Assíncronos: as notificações são enviadas de forma assíncrona pelos objetos geradores de eventos, para todos os objetos que fizeram uma assinatura deles

Eventos e notificações



Eventos e notificações

- A ideia é que um objeto pode reagir a uma alteração ocorrida em outro objeto
  - As notificações de eventos são basicamente assíncronas e determinadas pelos seus receptores
- Os sistemas distribuídos baseados em eventos ampliam o modelo de evento local
  - vários objetos em diferentes localizações sejam notificados de eventos ocorrendo em um objeto

Eventos e notificações

O paradigma empregado é o publicar-assinar (publish-subscriber)

- um objeto que gera eventos publica os tipos de eventos que tornará disponíveis para observação por outros objetos.
- objetos interessados em um evento fazem uma assinatura para receber notificações a respeito desse evento

Elementos Participantes

O objeto de interesse

- trata-se de um objeto que sofre mudanças de estado, como resultado da invocação de seus métodos;

Evento

- um evento ocorre em um objeto de interesse como resultado da conclusão da execução de um método;

Notificação

- é um objeto que contém informações sobre um evento

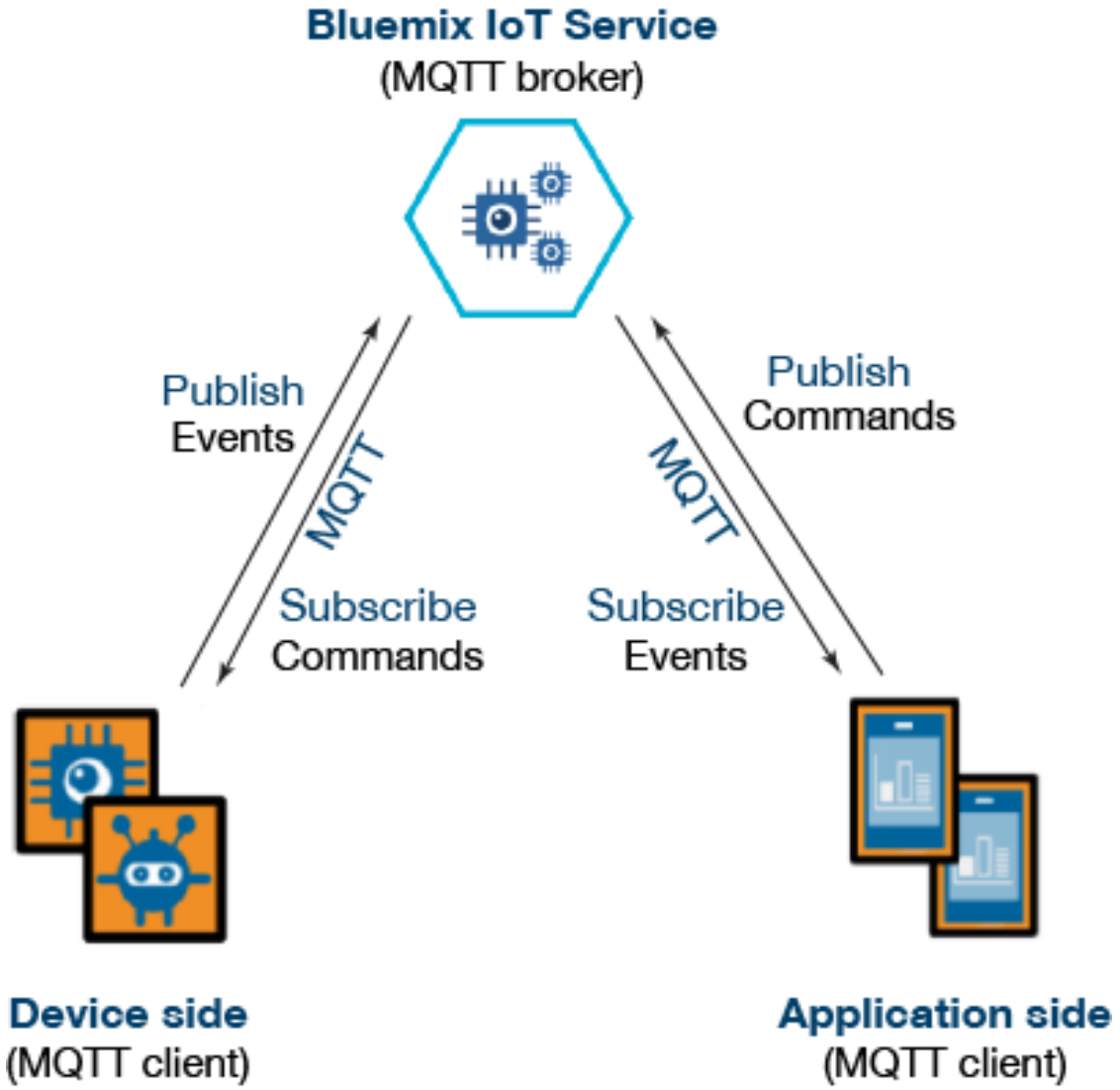
Assinante

- um assinante é um objeto que se inscreveu em algum tipo de evento em outro objeto;

Objetos observadores

- desvincula um objeto de interesse de seus assinantes;

Exemplo de Sistema



MQTT - Exemplo

MQTT (Message Queue Telemetry Transport)

- Criado nos anos 90 pela IBM

Protocolo para comunicação (camada aplicação) leve e assíncrono baseado em TCP

Flexível, oferece o equilíbrio ideal para os desenvolvedores de IoT:

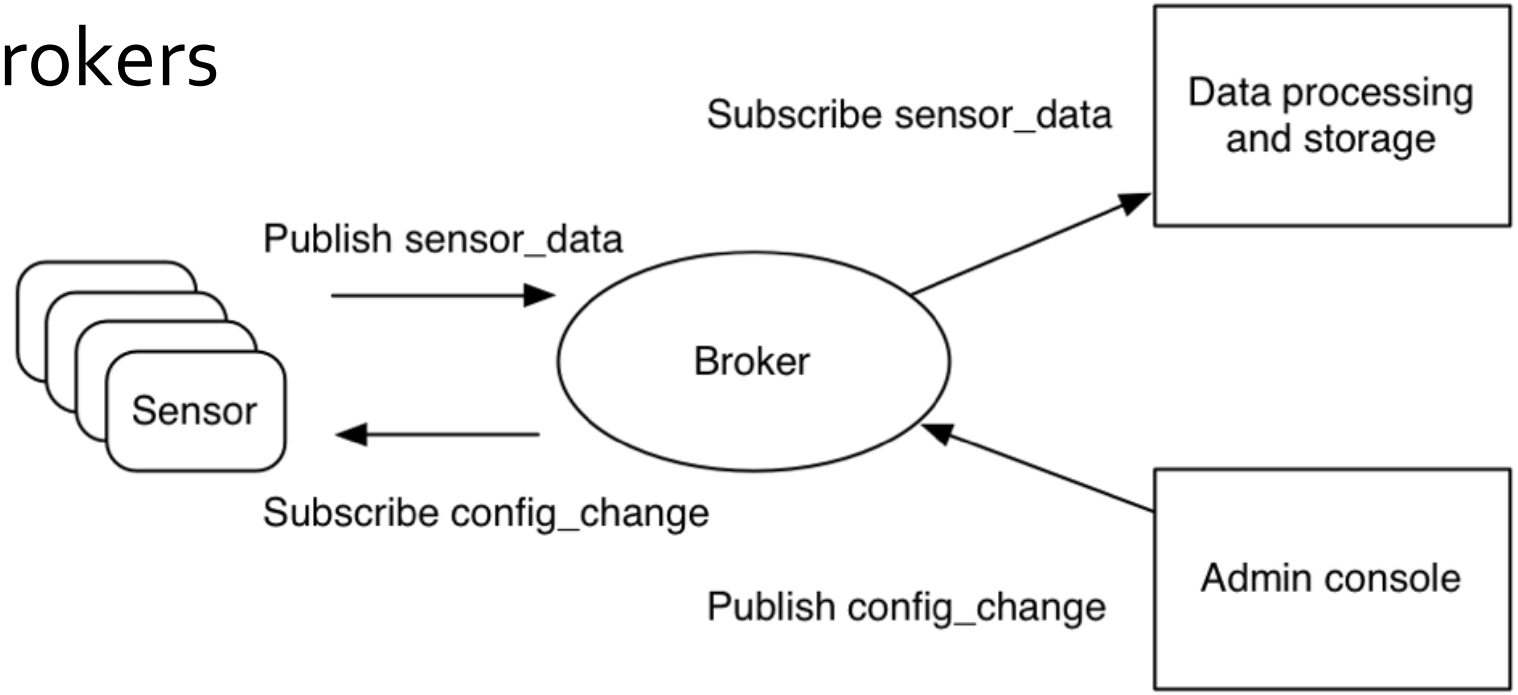
- Implementação em hardware de dispositivo restritos
- Redes de largura da banda limitada e de alta latência.
- Suporte a diversos cenários de aplicações para dispositivos e serviços de IoT.

MQTT - Exemplo

Modelo Pub-Sub baseado em tópicos

"context/ambient/temperature"

Uso de Brokers



MQTT

- Cabeçalho simples para especificar o tipo de mensagem,
  - um tópico baseado em texto
  - carga útil binária arbitrária (XML, JSON, Base64)
- Modelo de qualidade simplificado (IBM BlueMix)
  - Tentativa de entrega
  - Entregar ao menos uma vez
  - Entregar exatamente uma vez

Exemplos de Implementações: Mosquito e IBM BlueMix
https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html

Dúvidas?

Vamos praticar!