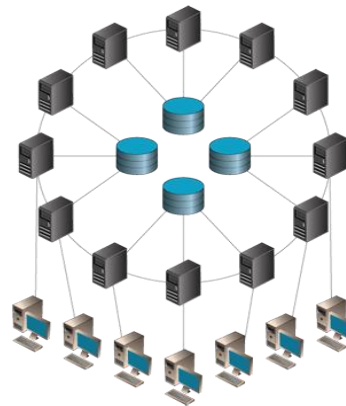


CKP7500 - SISTEMAS DISTRIBUÍDOS E REDES DE COMUNICAÇÃO

SMD0050 - SISTEMAS DISTRIBUÍDOS - T02

MODELOS E ESTILOS ARQUITETÔNICOS

SLIDES SÃO BASEADOS NOS SLIDES DO COULORIS E TANENBAUM



Modelagem e projeto de um sistema distribuído

Muitos desafios a serem suplantados

Projetar um SD em três perspectivas

- Modelos Físicos
 - Hardware envolvido (rede e dispositivos)

Modelos Arquiteturais

- Camadas de software e distribuição entre as entidades envolvidas

Modelos Fundamentais

- Quais estratégias serão utilizadas para garantir a coordenação, a segurança e dependabilidade do sistema distribuído?

Modelos Físicos

Três Gerações de Sistemas distribuídos

- Fase I : Acesso Remoto e Compartilhamento de recursos
- Fase II: Sistemas baseados na Internet
- Fase III: Sistemas contemporâneos em larga escala com grande heterogeneidade de acesso

Sistemas de Terceira Geração podem ser vistos como Sistemas de Sistemas ou Sistemas Complexos

Gerações de Sistemas Distribuídos

<i>Distributed systems:</i>	<i>Early</i>	<i>Internet-scale</i>	<i>Contemporary</i>
<i>Scale</i>			
<i>Heterogeneity</i>			
<i>Openness</i>			
<i>Quality of service</i>			

Gerações de Sistemas Distribuídos

<i>Distributed systems:</i>	<i>Early</i>	<i>Internet-scale</i>	<i>Contemporary</i>
<i>Scale</i>	Small	Large	Ultra-large
<i>Heterogeneity</i>	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
<i>Openness</i>	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
<i>Quality of service</i>	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Modelos Arquiteturais

Organização ou estrutura que responde às seguintes questões

- Quais são entidades (software) e seus papéis na execução das tarefas do sistema distribuído?
- Como essas entidades irão se conectar e qual paradigma de comunicação será utilizado?
- Onde as entidades estarão fisicamente localizadas?

Arquiteturas

Sistemas distribuídos muitas vezes são complexas peças de software cujos componentes estão espalhados por várias máquinas.

Organização de um sistema distribuído:

- Organização lógica do conjunto de componentes de software;
- A realização física propriamente dita.

Modelos Arquiteturais

Quais são entidades comunicantes em um SD?

- Processos
 - Entidade de software comunicante
 - Pode conter subprocessos (Threads)
- Nós físicos
 - Sensores, sistemas de automação

Do ponto de vista de desenvolvimento de um SD

- Objetos
- Componentes
- Serviços

Modelos Arquiteturais

Arquitetura Cliente-Servidor

- Cliente magro ou cliente gordo
- Proxy
- Múltiplos servidores
- Orientada a Serviços

Arquitetura Peer-to-Peer

- Totalmente descentralizada
- Hierárquica

Arquiteturas de Sistemas

Arquiteturas Centralizadas

- Modelo Cliente-Servidor

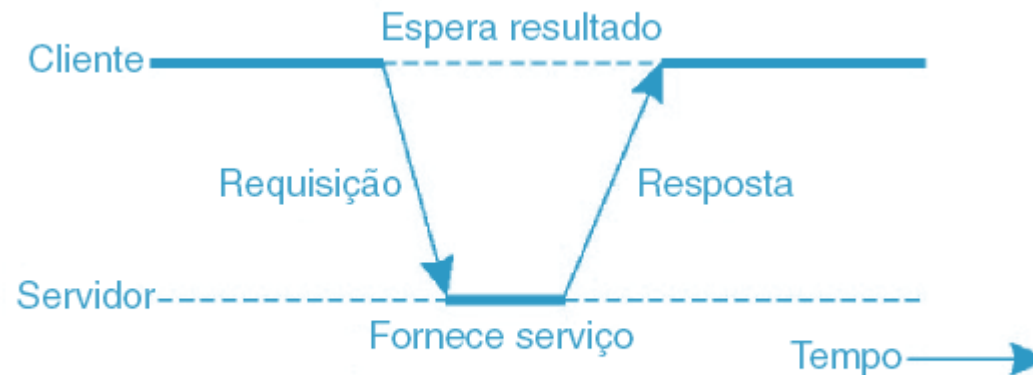


Figura 2.3 Interação geral entre um cliente e um servidor.

- Problemas: Orientado ou não-orientado a conexão (TCP ou UDP)?

Arquiteturas Centralizadas

Um exemplo de divisão em 3 níveis:

- Nível de interface de usuário
 - Contém tudo o que é necessário para fazer interface com o usuário
- Nível de processamento
 - Contém as aplicações
- Nível de dados
 - Gerencia os dados propriamente ditos

Arquiteturas Centralizadas

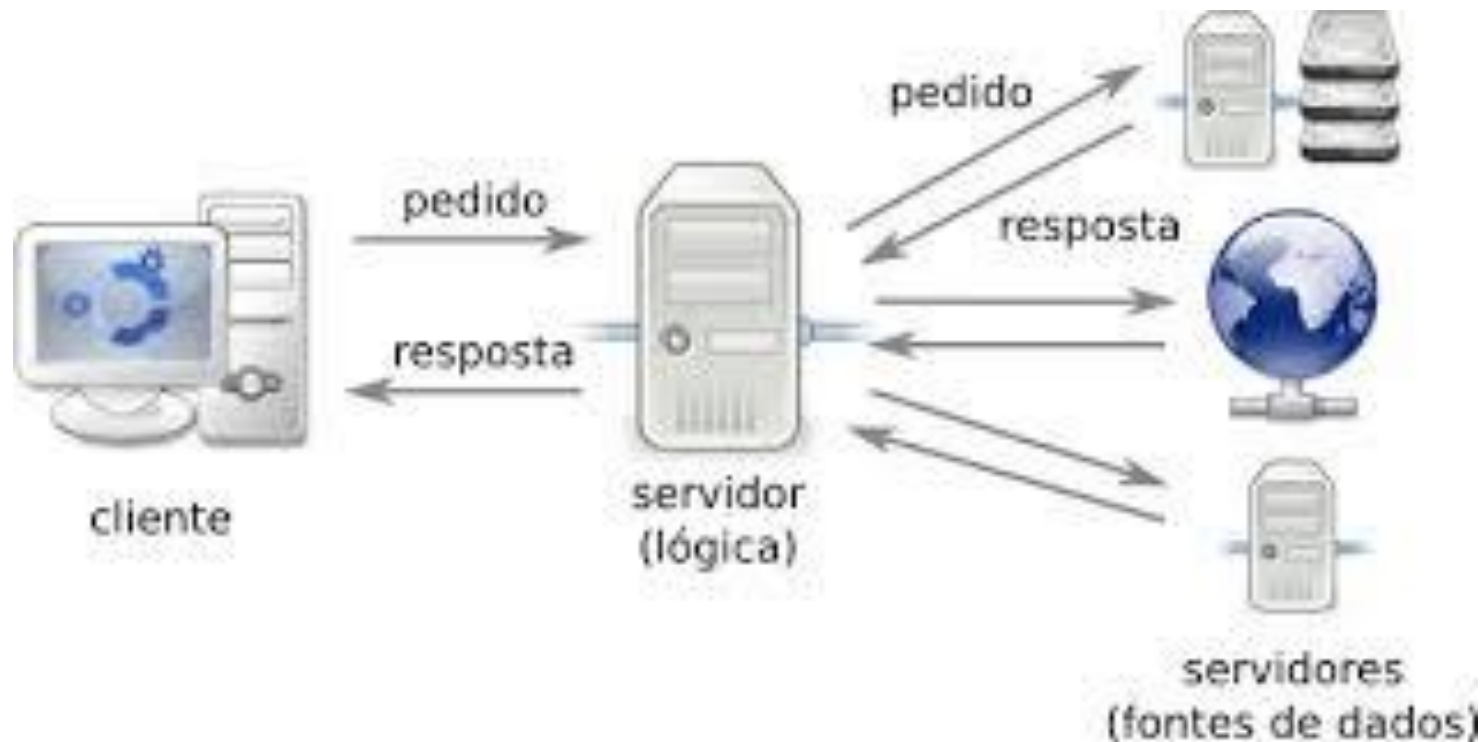
Arquiteturas multidividadas

Com base na organização de três níveis lógicos discutida anteriormente,

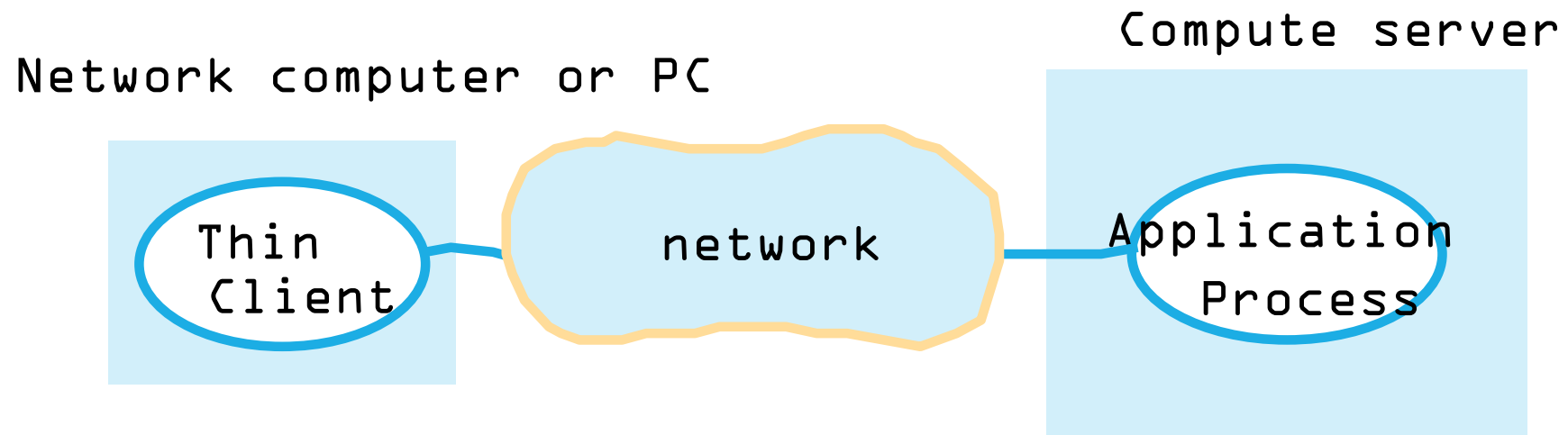
É necessária a distribuição física. A maneira mais simples, denominada arquitetura de duas divisões (físicas) é distribuída da seguinte forma:

- Uma máquina cliente que contém apenas os programas que implementam o nível (ou parte do nível) de interface de usuário
- Uma máquina do servidor que contém todo o resto, ou seja, os níveis de processamento e de dados.

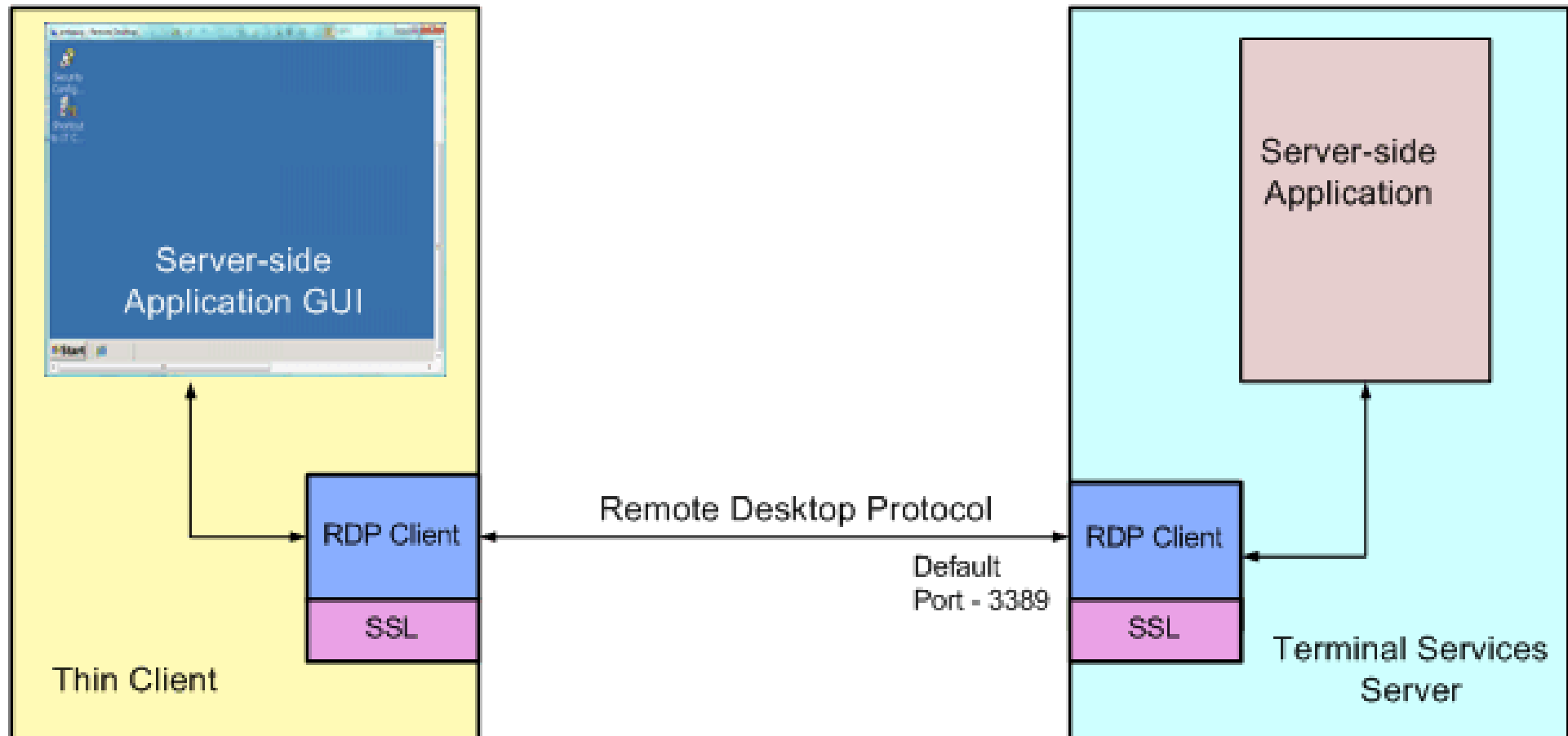
Arquitetura Cliente-Servidor



Thin client (Cliente “magro”)



Thin client (Cliente “magro”) – Terminal Remoto



Arquiteturas Centralizadas

Arquiteturas multidividadas

Arquitetura de três divisões (físicas)

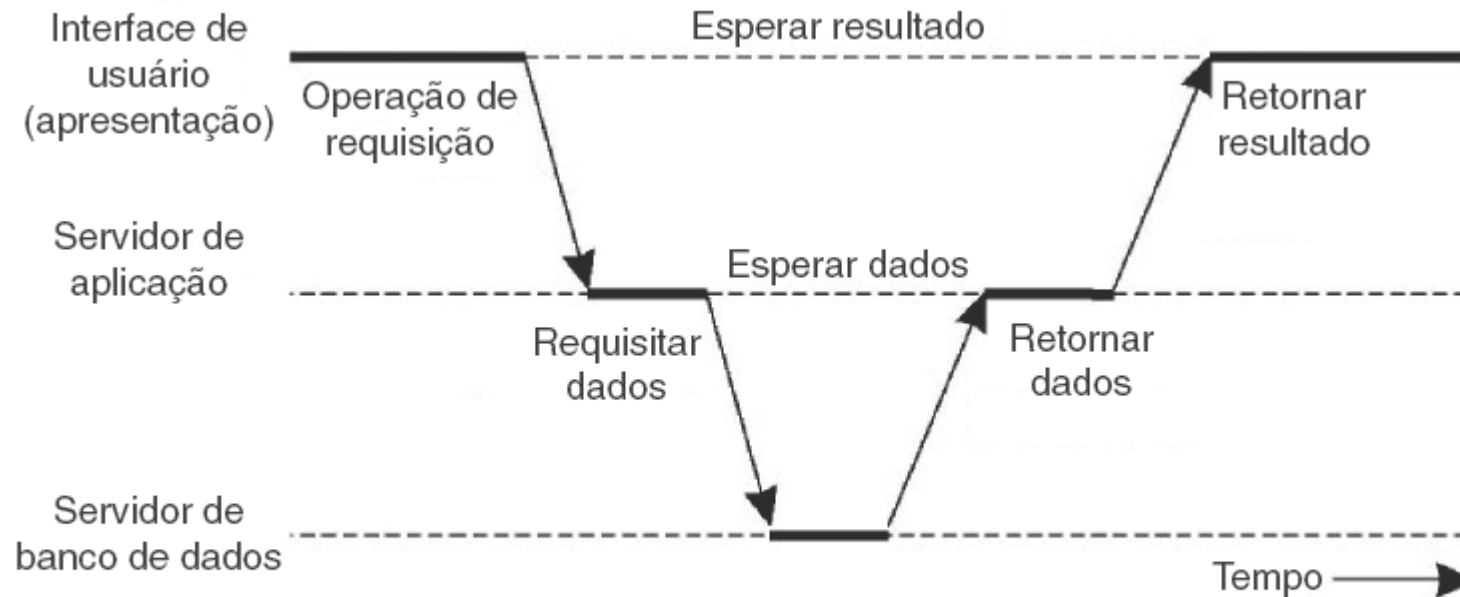


Figura 2.6 Exemplo de um servidor que age como cliente.

Arquiteturas Centralizadas

Arquiteturas multidividadas

Alternativas de distribuição

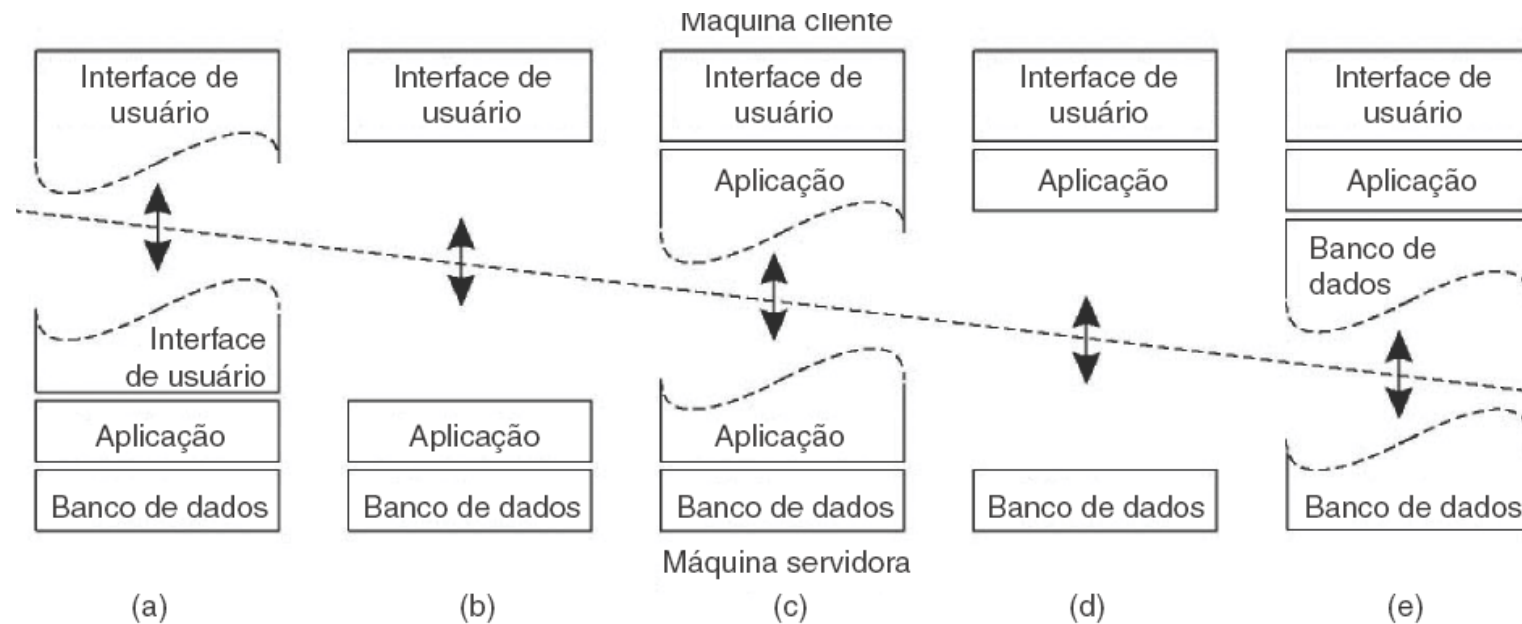


Figura 2.5 Alternativas de organizações cliente-servidor (a)–(e).

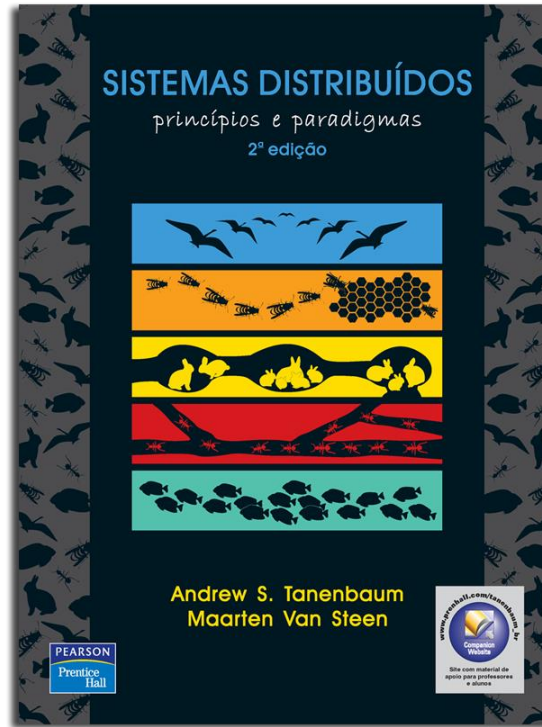
Exercício em Sala de Aula

Classifique cada exemplo abaixo em uma arquitetura multi-dividida

Exemplos

- Progressive Web Applications
- Aplicações Cross-Platform Interpretadas
- Wikipedia
- Aplicativo móvel do Facebook
- Ifood
- Plataforma de Jogos baseada em Stream

Arquiteturas



capítulo

2

Estilos Arquitetônicos

Formado em termos de componentes, do modo como esses componentes estão conectados uns aos outros, dos dados trocados entre componentes e, por fim, da maneira como esses elementos são configurados em conjunto para formar um sistema.

- Componente é uma unidade modular com interfaces requeridas e fornecidas bem definidas que é substituível dentro de seu ambiente.
- Conector é um mecanismo mediador da comunicação ou da cooperação entre componentes.

Estilos Arquitetônicos

Arquitetura em camadas

- Componentes organizados em camadas, onde componentes da camada Li pode chamar métodos da camada Li-1, mas não o contrário;

Arquiteturas baseadas em objetos

- Objetos correspondem às definições de componentes, que são conectados por meio de chamadas de procedimento remotas;

Estilos Arquitetônicos

(a) em camadas e (b) baseado em objetos

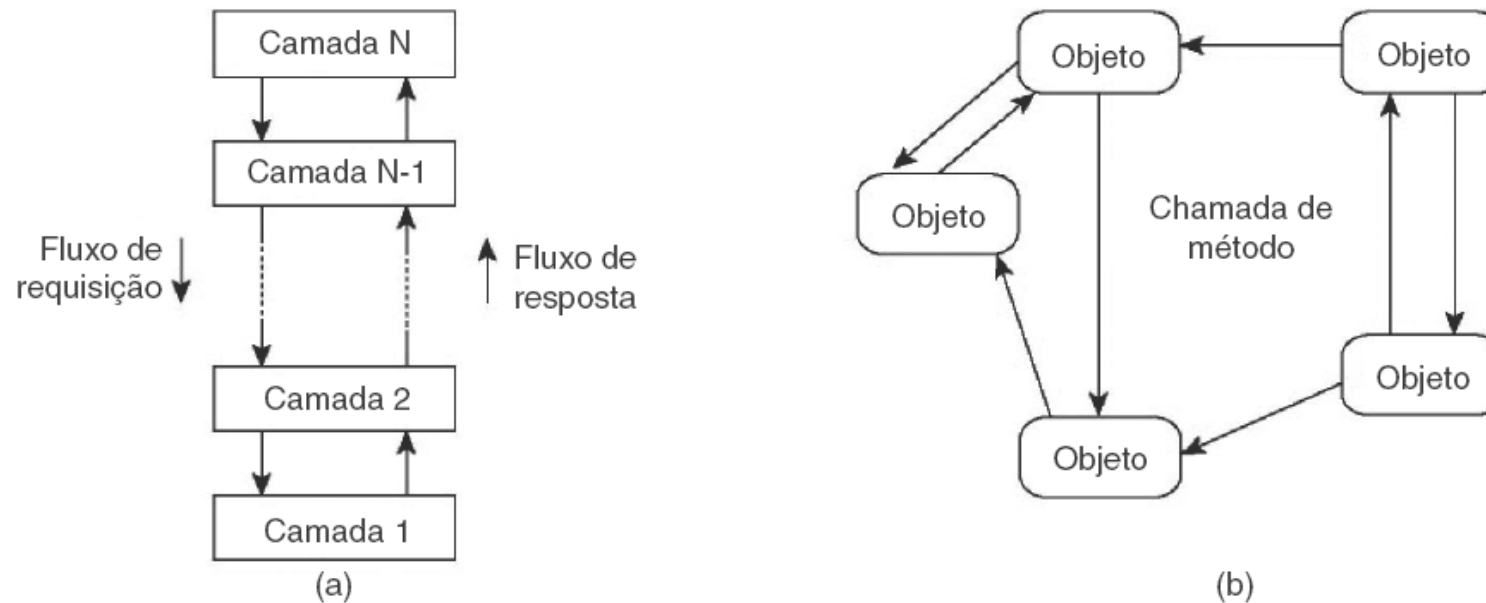


Figura 2.1 Estilo arquitetônico (a) em camadas e (b) baseado em objetos.

Estilos Arquitetônicos

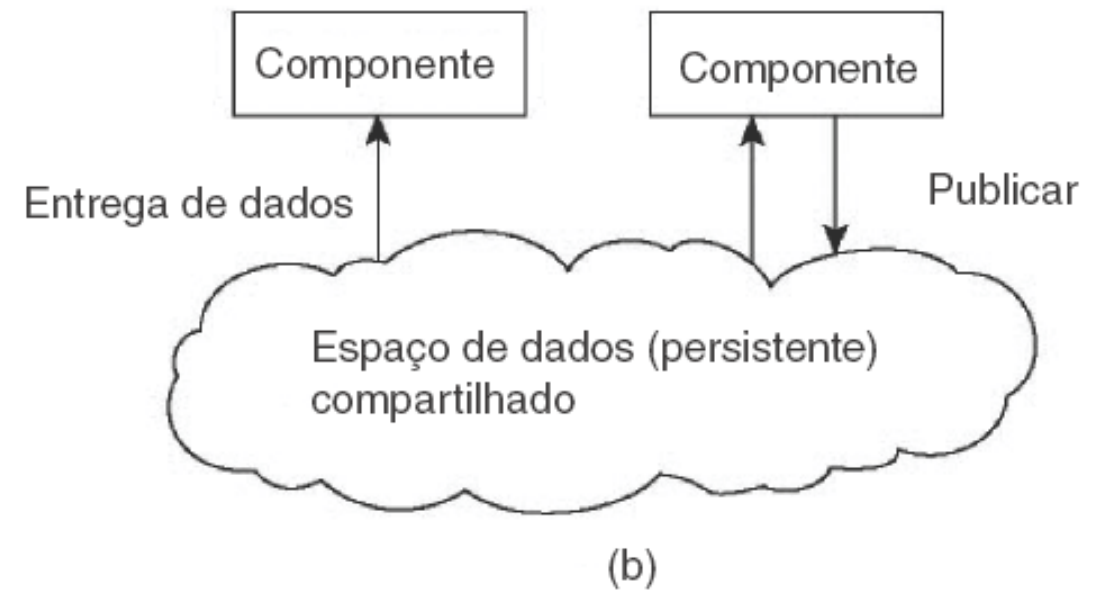
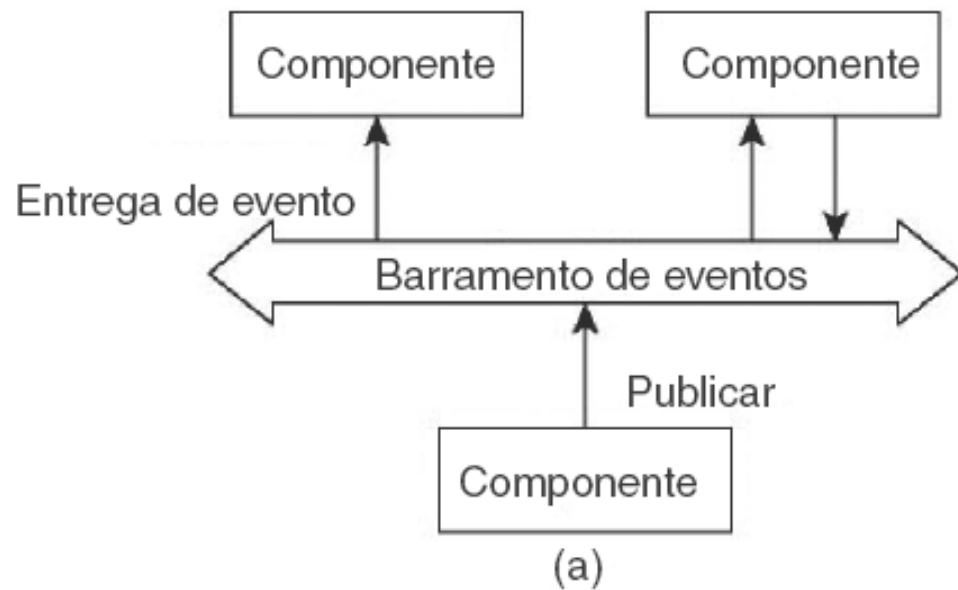
Arquiteturas centradas em dados

- Processos se comunicam por meio de repositório comum (passivo ou ativo);

Arquiteturas baseadas em eventos

- Processos se comunicam por meio de propagação de eventos que podem transportar dados;
- Sistemas Publicar/Subscrever;
- São referencialmente desacoplados.

Estilos Arquitetônicos



(a) baseados em eventos e (b) centradas em dados

Estilos Arquitetônicos

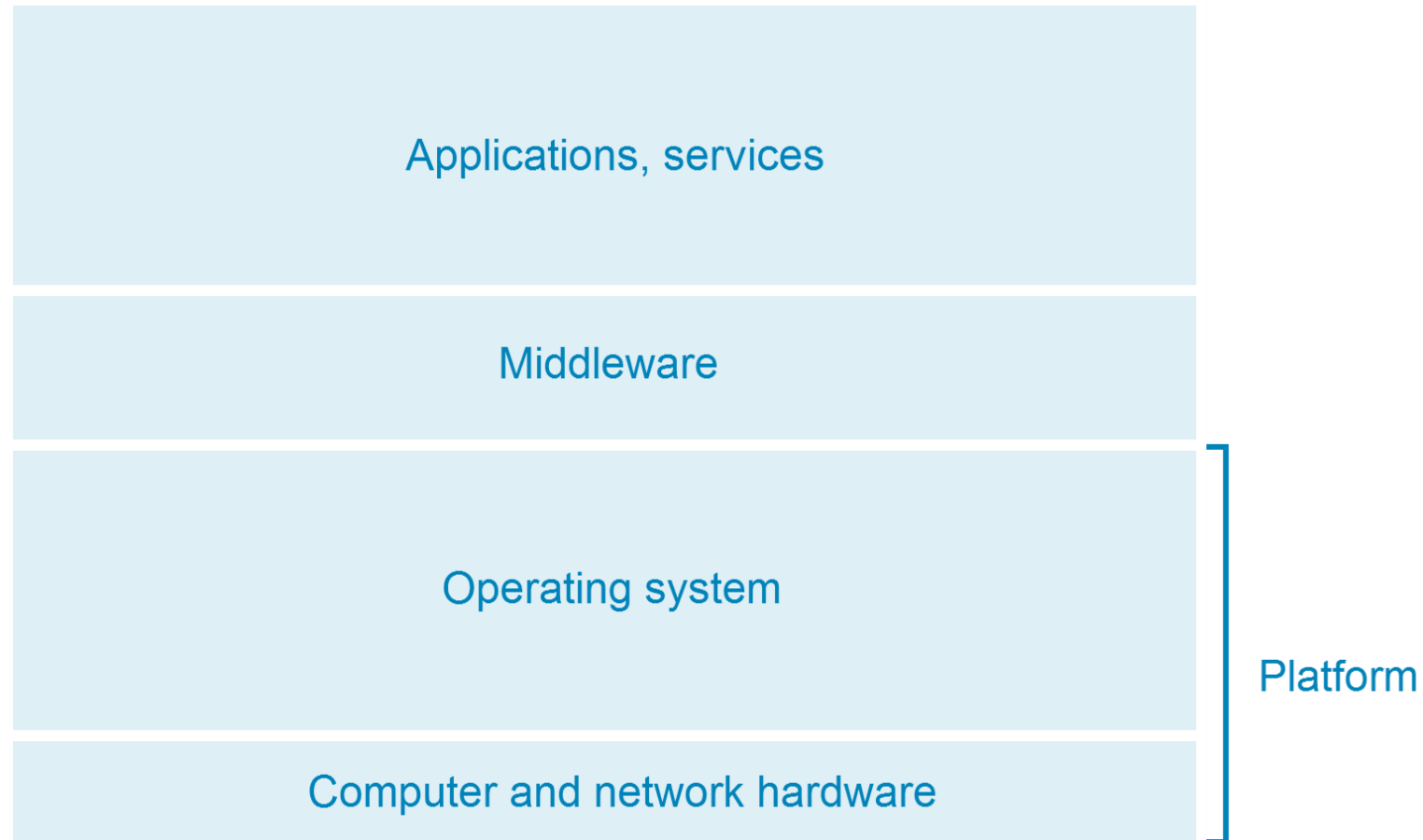
Estilos podem ser híbridos, como arquiteturas baseadas em eventos juntamente com centradas em dados, também conhecidas como espaços compartilhados de dados.

Padrões Arquiteturais

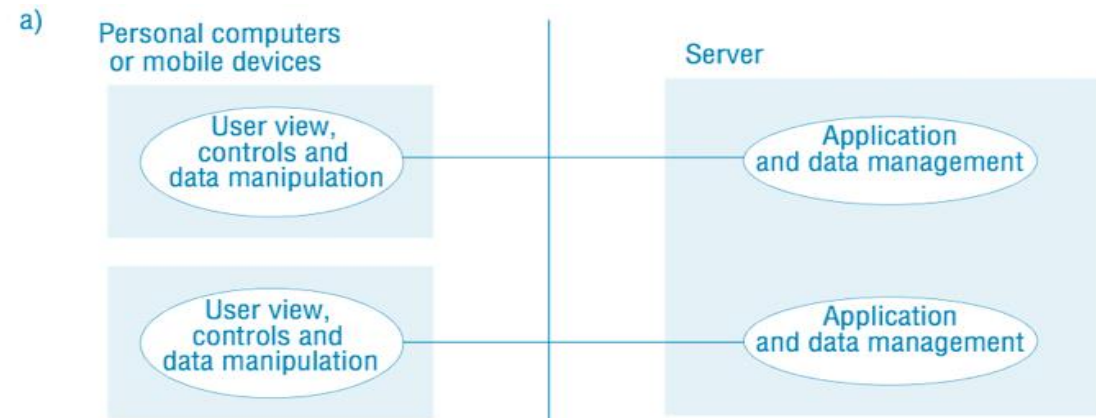
Arquiteturas recorrentes em SD

- Arquitetura em camadas
- Arquitetura Two-Tier
- Arquitetura Three-Tier
- Arquitetura Orientada a Serviços

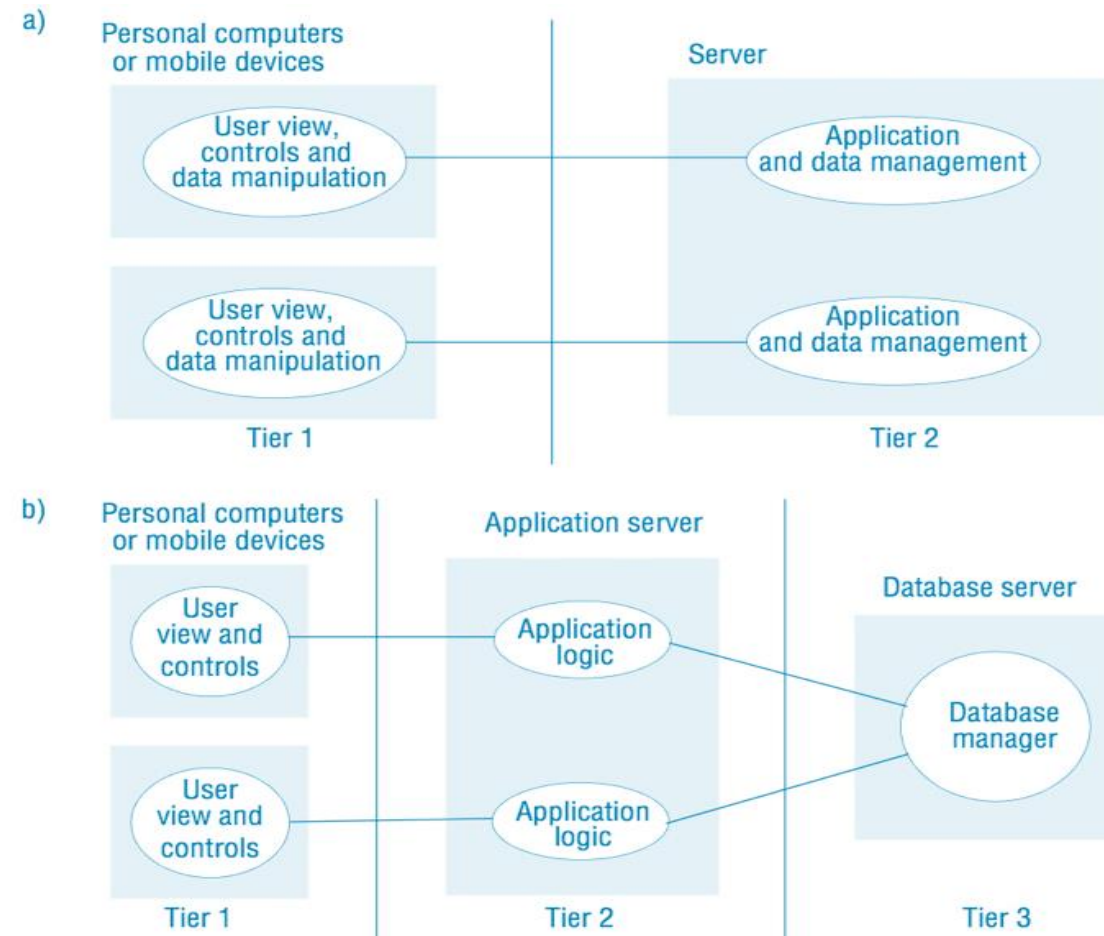
Representação Arquitetural em Camadas



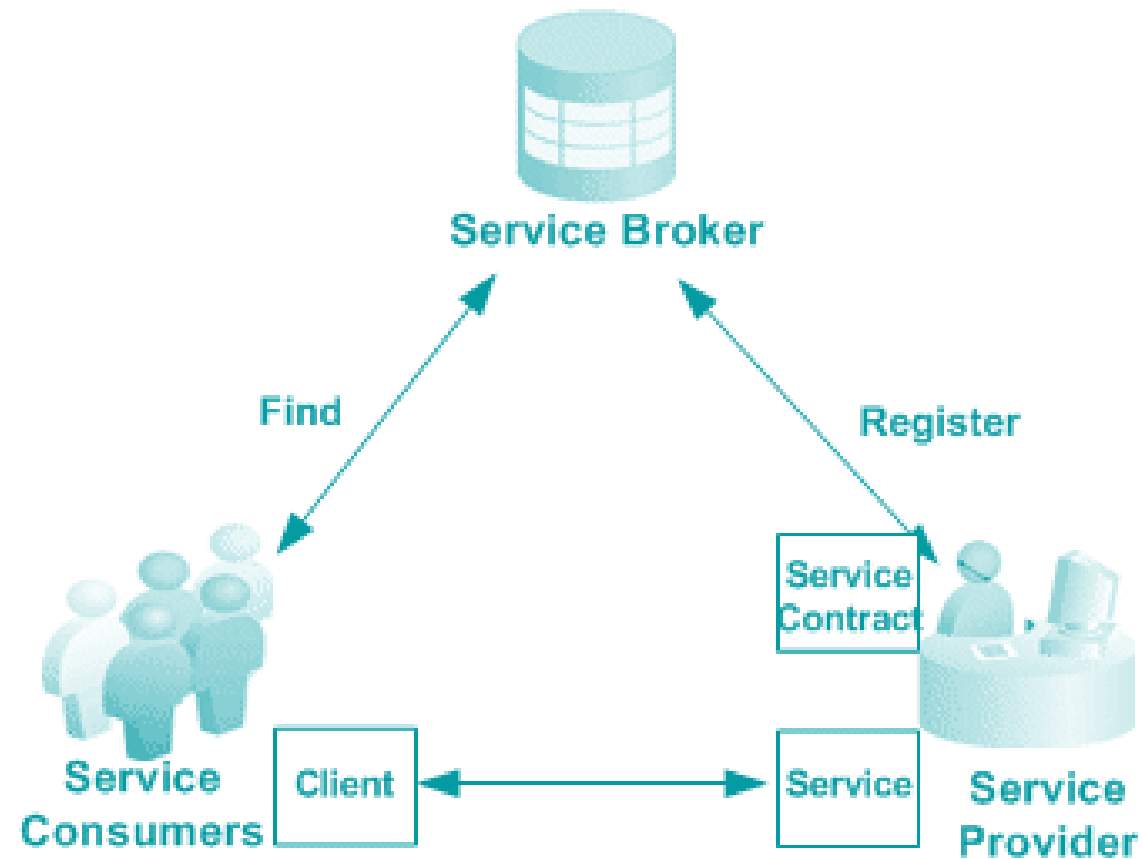
Arquiteturas Two-tier e Three-tier



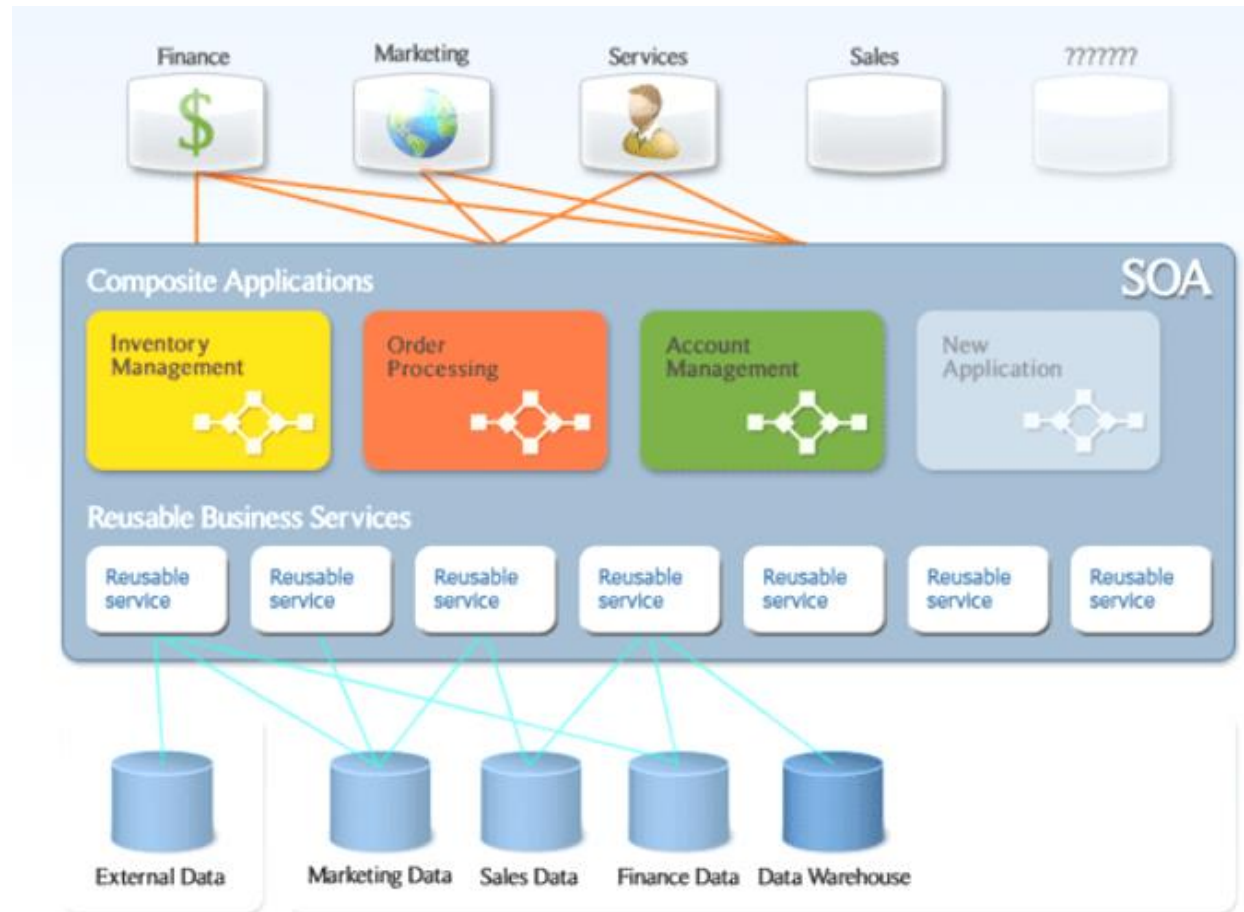
Arquiteturas Two-tier e Three-tier



Arquitetura orientada a serviços



Arquitetura Orientada a Serviços



Exercício em Sala de Aula 2

Encontre um exemplo para cada padrão/estilo arquitetural

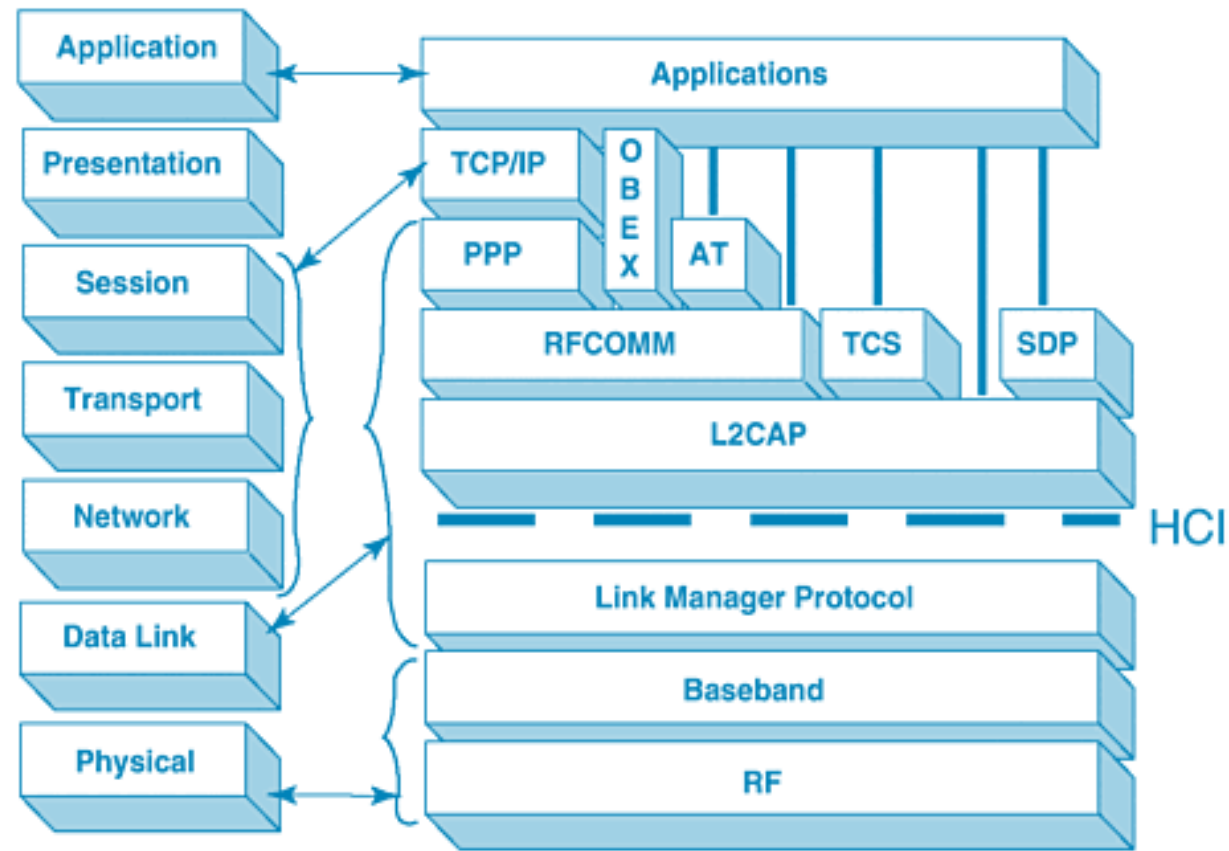
Em camadas

Two-Tiers

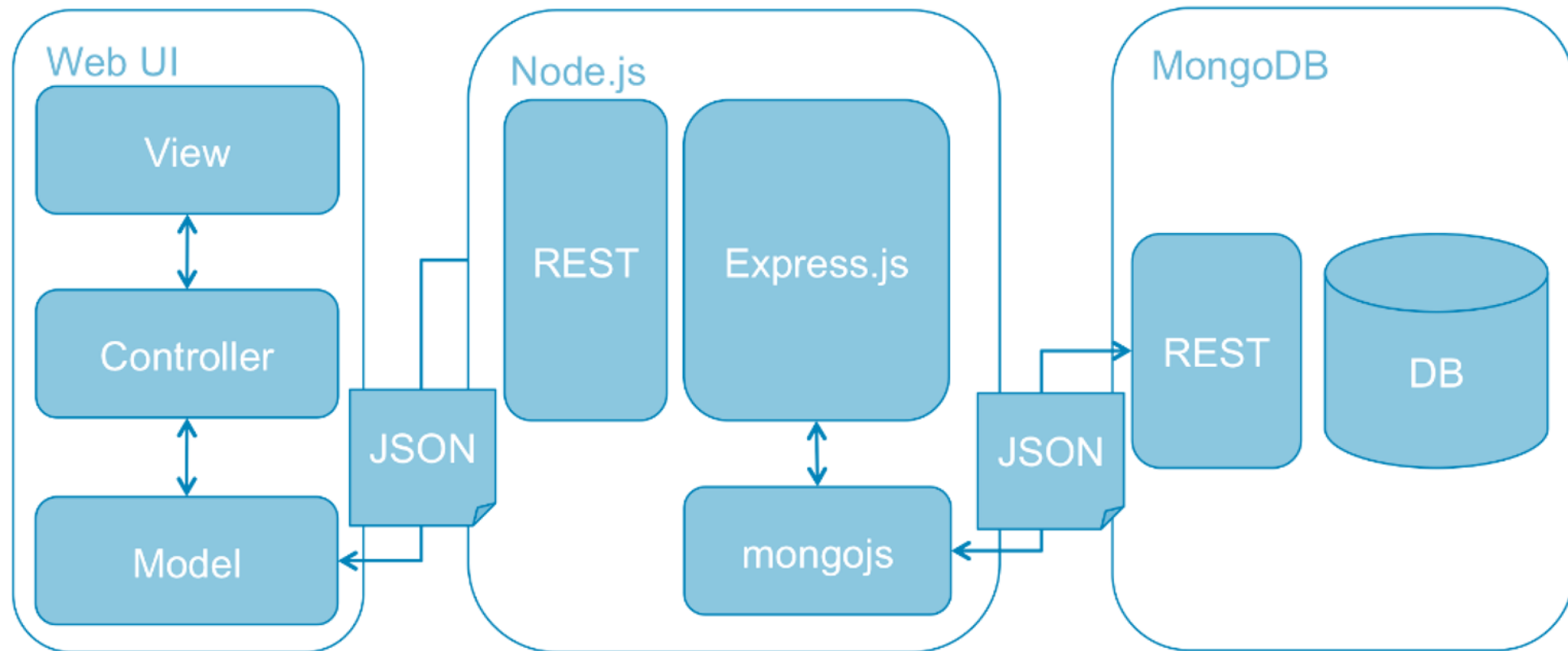
Three-Tiers

Orientado a Serviços

Bluetooth



Um Sistema Web



Organização em 3 Camadas de Aplicação

Um mecanismo de busca da Internet

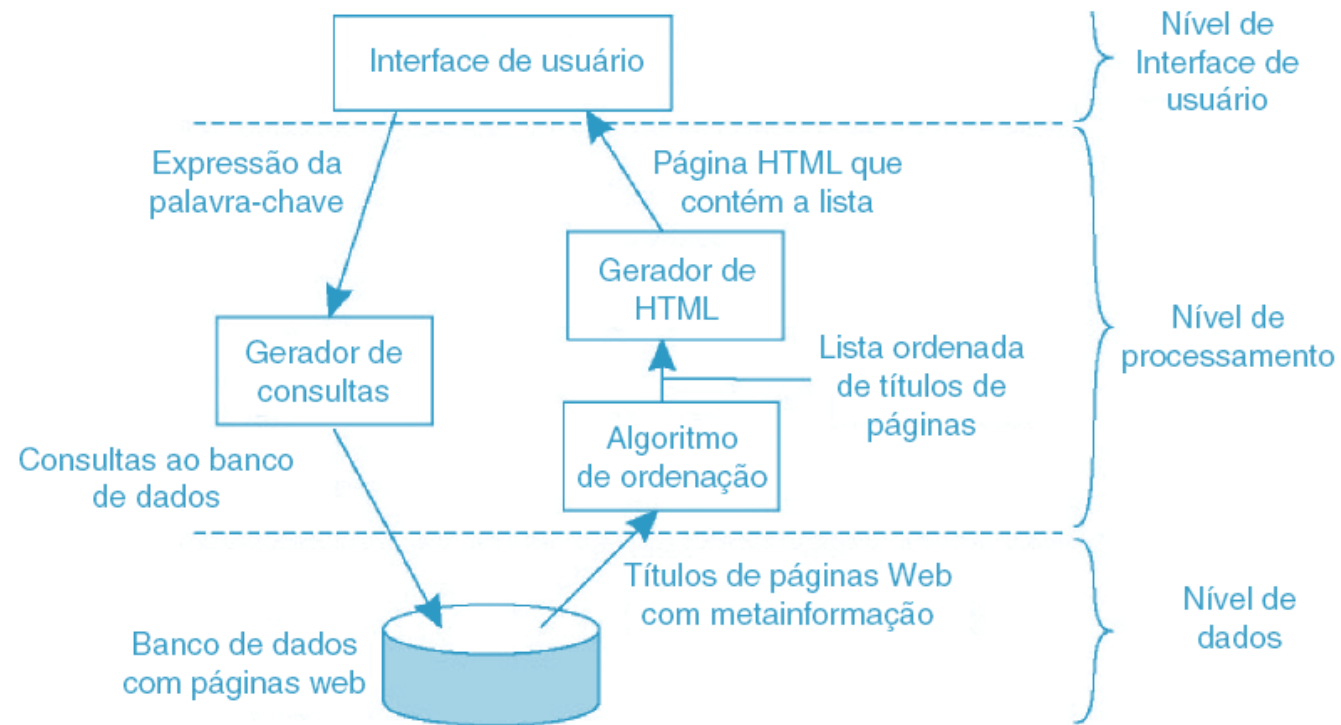


Figura 2.4 Organização simplificada de um mecanismo de busca da Internet em três camadas diferentes.

E Microservices?

Define an architecture that structures the application as a set of loosely coupled, collaborating services.

Each service implements a set of narrowly, related functions.

- For example, an application might consist of services such as the order management service, the customer management service etc.

Services communicate using either synchronous protocols such as HTTP/REST or asynchronous protocols such as AMQP.

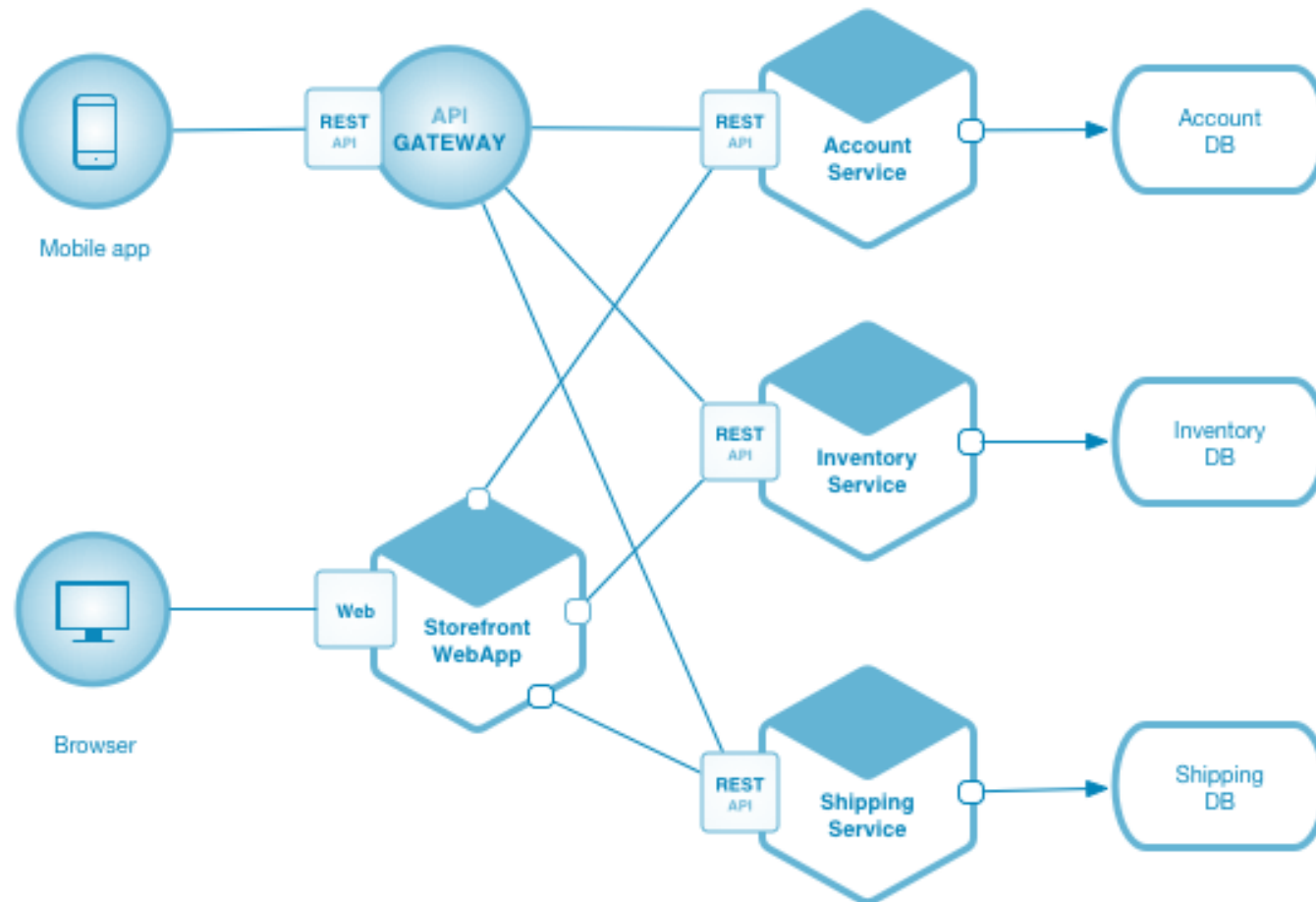
E Microservices?

Services can be developed and deployed independently of one another.

Each service has its own database in order to be decoupled from other services.

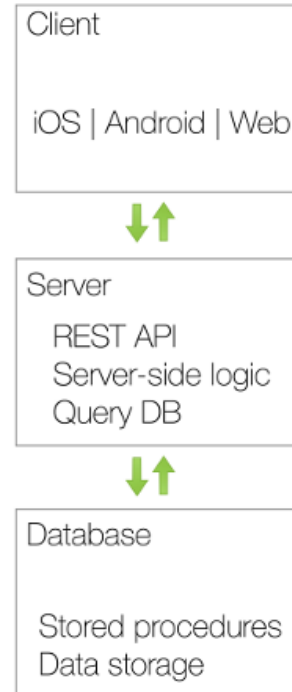
Data consistency between services is maintained using an event-driven architecture

E Microservices?

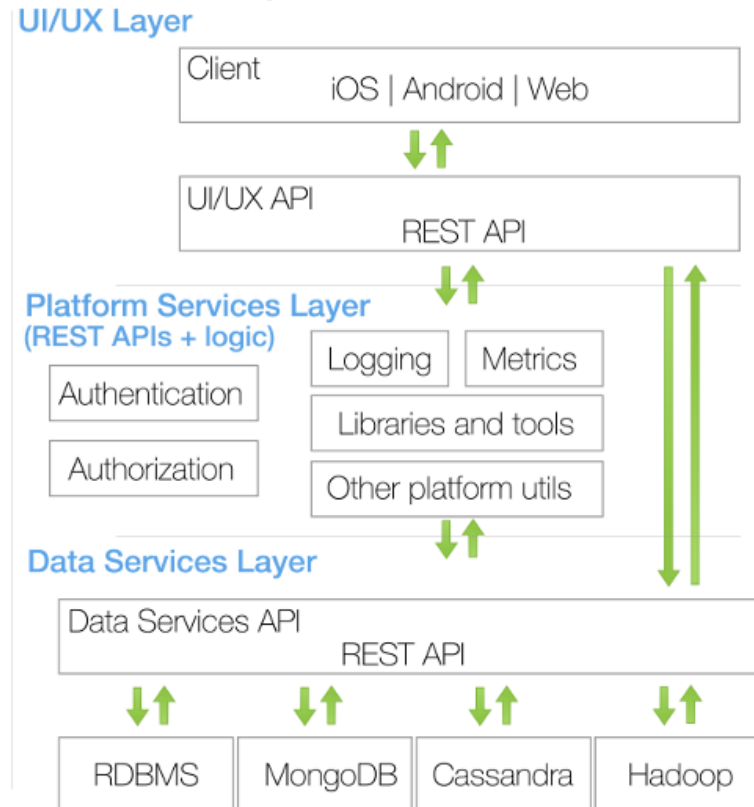


Aumentando a escalabilidade

3-tier architecture



3-layer architecture



Arquiteturas descentralizadas

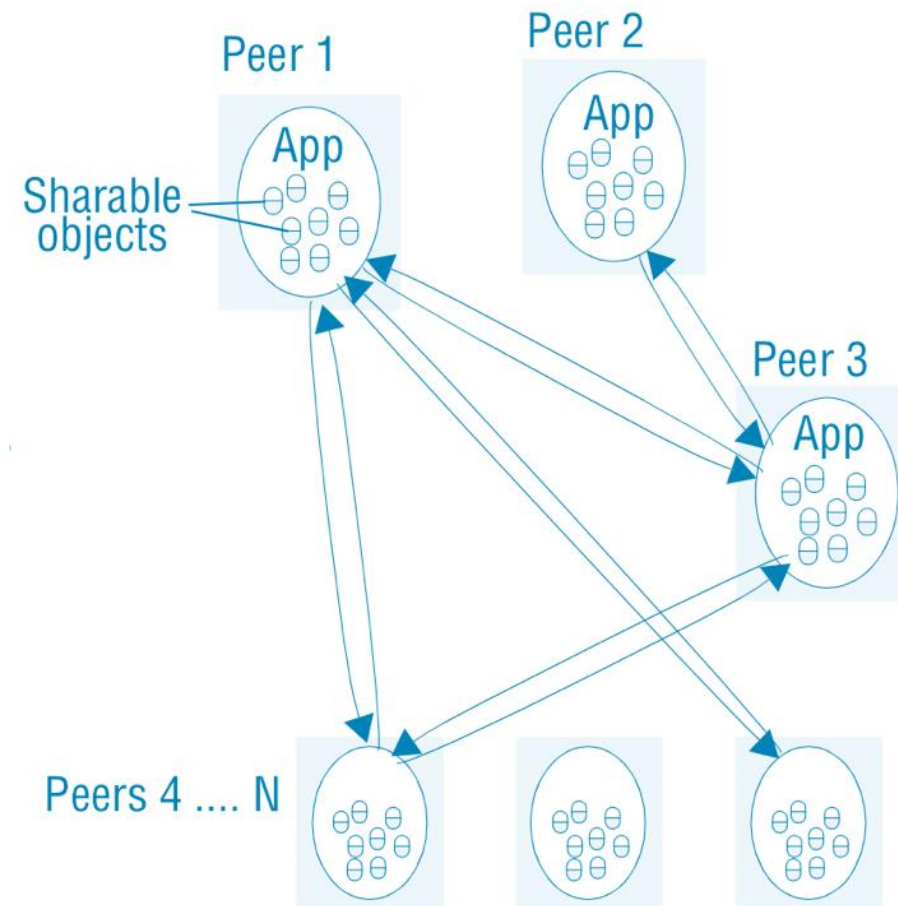
Distribuição vertical

- divide componentes logicamente diferentes em máquinas diferentes;

Distribuição horizontal

- Um cliente ou servidor pode ser subdividido em partes logicamente equivalentes, mas cada parte está operando em sua própria porção do conjunto de dados, equilibrando a carga.
- Ex.: Peer to Peer (servidor e cliente ao mesmo tempo, também chamada “servente”)

Arquitetura Peer-to-Peer (P2P)



Modelos Arquiteturais

Como essas entidades irão se conectar e qual paradigma de comunicação será utilizado?

A resposta está muitas vezes relacionada a tecnologia de desenvolvimento utilizada

- Comunicação entre processos
- Invocação remota
- Comunicação indireta

Paradigmas de comunicação

INVOCAÇÃO REMOTA

Sockets, RPC, RMI

Passagem de mensagens

Request-Reply

COMUNICAÇÃO INDIRETA

Eventos (Publish-Subscriber)

Memória Compartilhada –Espaço de Tuplas

Multicasting

Applications, services

Remote invocation, indirect communication

Underlying interprocess communication primitives:
Sockets, message passing, multicast support, overlay networks

UDP and TCP

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 bloqueante

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

<i>Nome</i>	<i>Mensagens enviadas</i>		
	<i>Clien</i>	<i>Servido</i>	<i>Clien</i>
R	<i>Reque</i>	<i>r</i>	<i>te</i>
RR	<i>Reque</i>	<i>Reply</i>	
RRA	<i>Reque</i>	<i>Repl</i>	<i>Acknowledge reply</i>
	<i>st</i>	<i>y</i>	

Requisição-Resposta (Request-Reply)

Request-reply message structure

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

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

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

HTTP *Reply* message

<i>HTTP version</i>	<i>status code</i>	<i>reason</i>	<i>headers</i>	<i>message body</i>
HTTP/1.1	200	OK		resource data

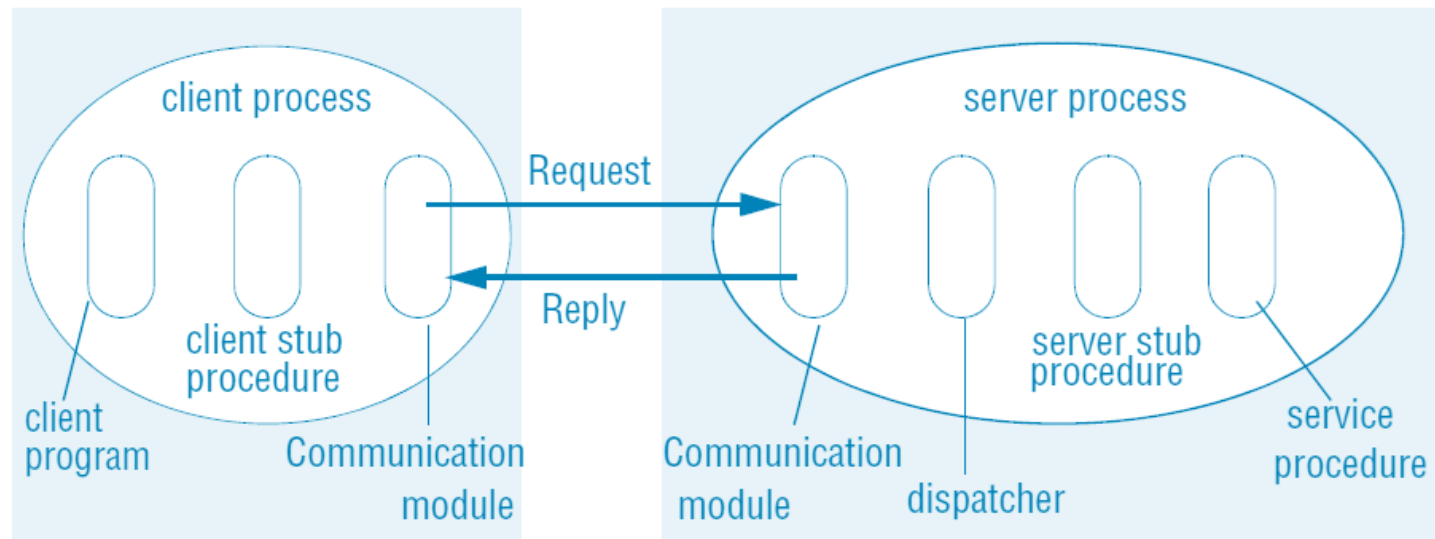
RPC – Remote Procedure Call

Procedimento passível de invocação remota

- Uso de uma IDL (Interface Description Language) para descrever a interface

Referências via stubs

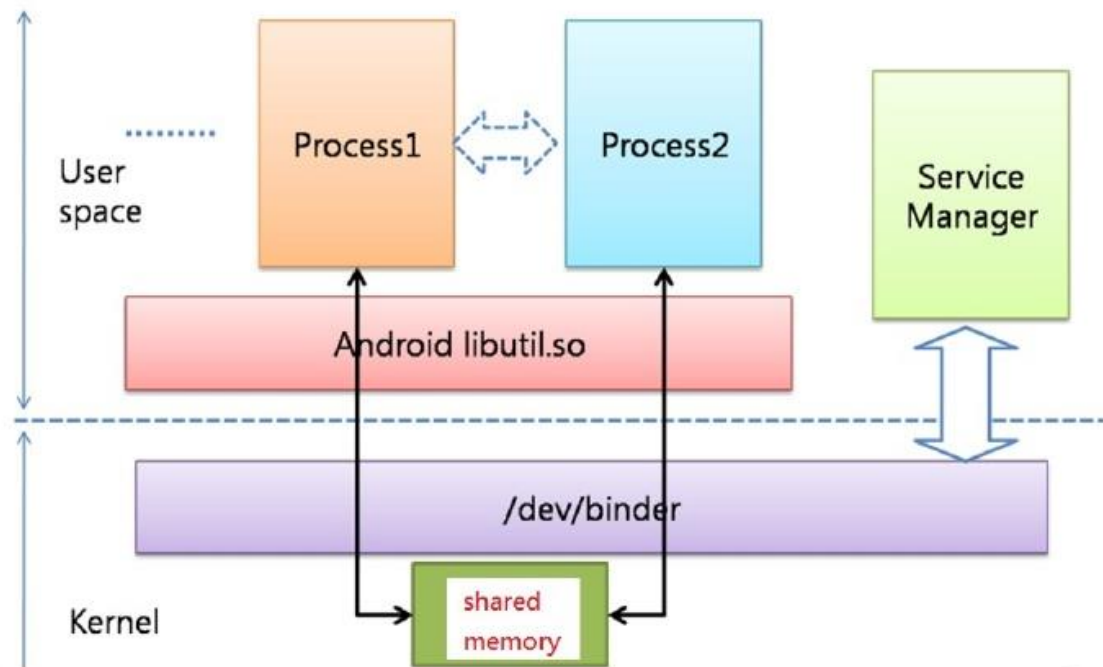
- doOperation, getRequest and sendReply



Binder no Android

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

Binder



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

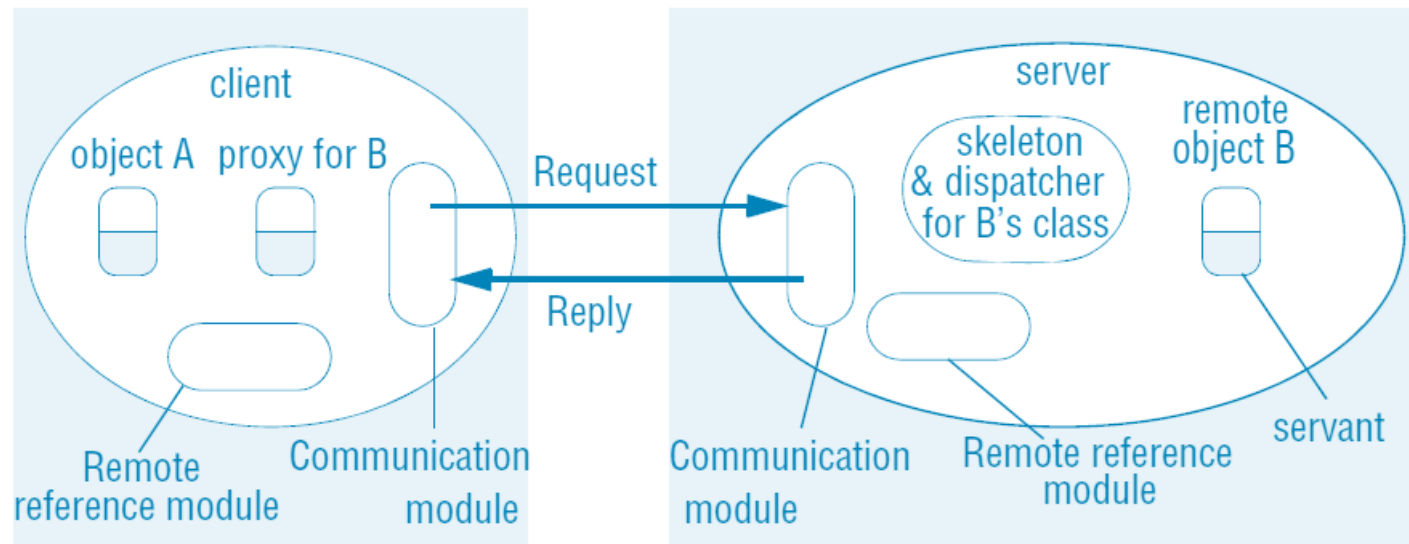
RMI – Remote Method Invocation

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

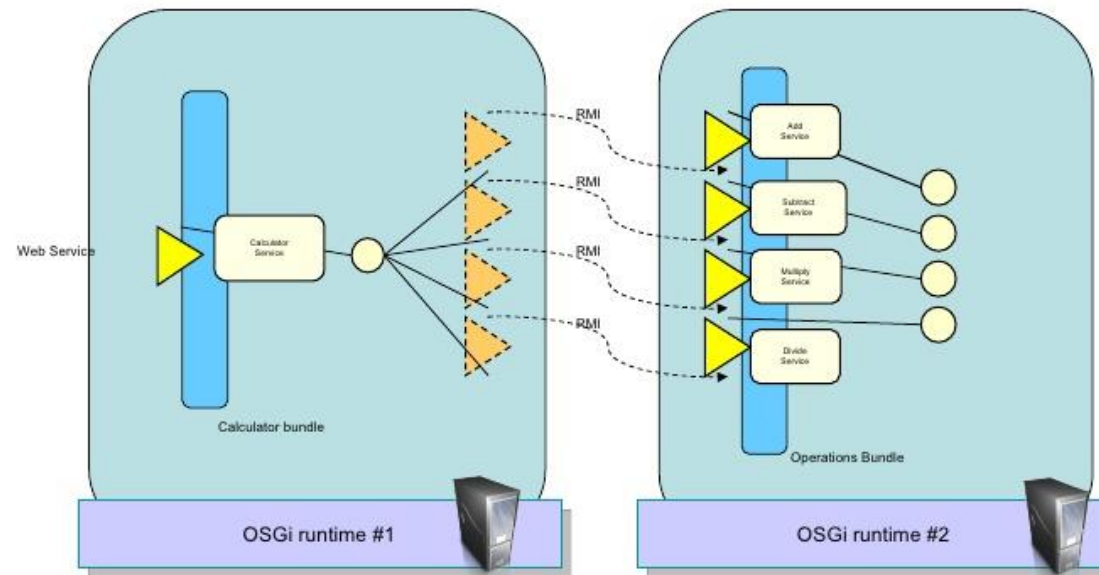
- proxies, esqueletos e despachantes

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

OSGi Remote Services enabled Calculator



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

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

Exemplos

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

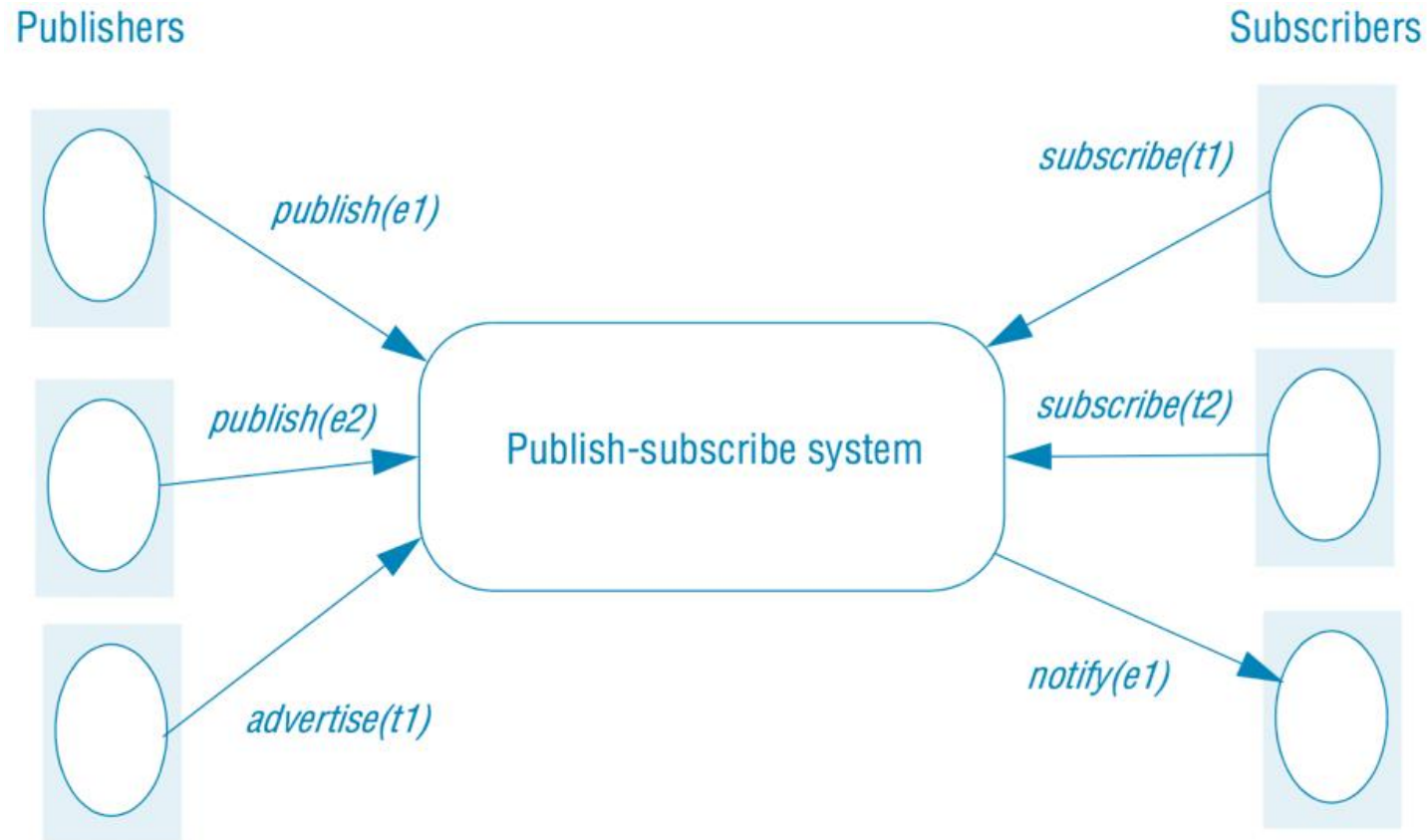
Sistemas Distribuídos baseados em Eventos

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

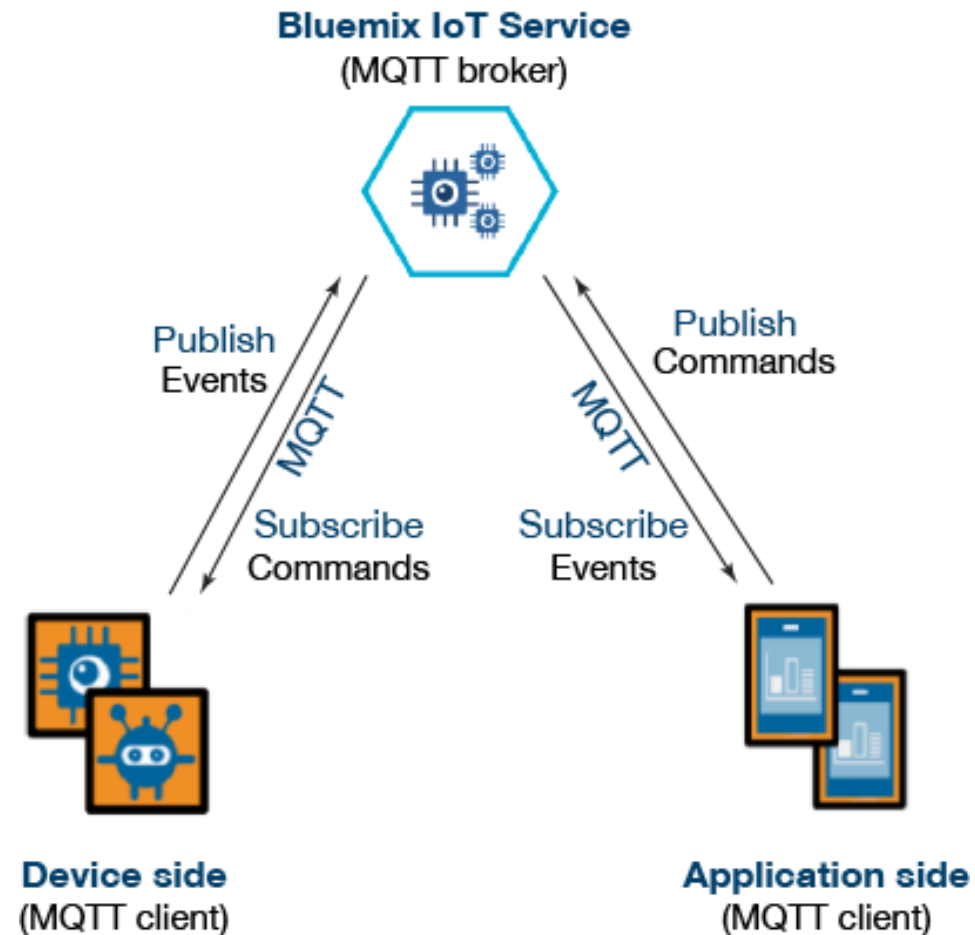
Assinante

- 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:

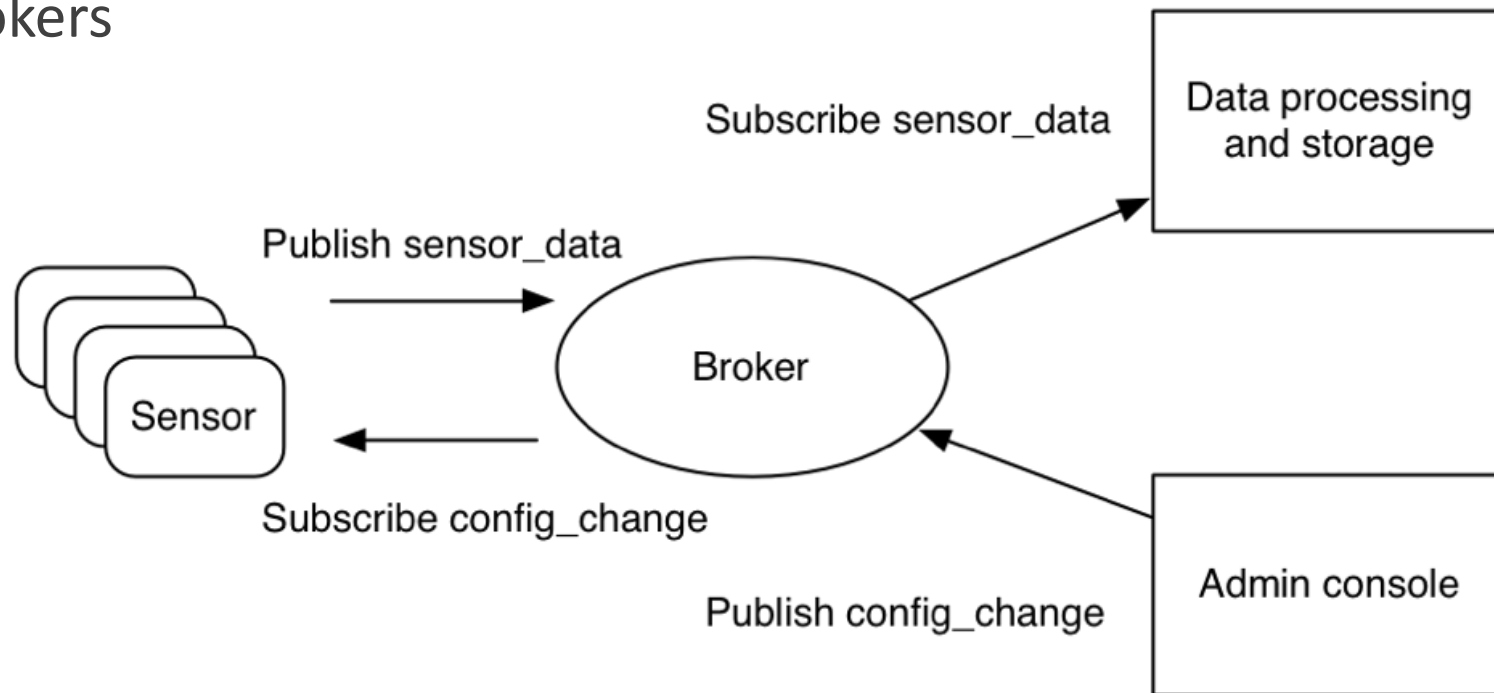
- Implementação em hardware de dispositivo restritos
- Redes de largura da banda limitada e de alta latência.
- Suporte a diversos cenários de aplicações para dispositivos e serviços de IoT.

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

Implementações

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

Modelos Fundamentais

A modelagem permite identificar, em fase de projeto de um SD, requisitos a serem implementados

- Garantia de performance e corretude
- Requisitos funcionais e não-funcionais
- Especificação Formal

Modelo Interação

Modelo de Falhas

Modelo de Segurança

Dúvidas?

