**PERGUNTAS IMPORTANTES**

**COMPUTAÇÃO DISTRIBUÍDA**

Conteúdo

[Sistemas Distribuídos 6](#_Toc138771744)

[1. O que é um sistema distribuído? 6](#_Toc138771745)

[2. Quais são as principais características de um sistema distribuído? 6](#_Toc138771746)

[3. Em questão de coordenação entre os vários nós de uma rede, qual a diferença entre um grupo aberto e fechado? 6](#_Toc138771747)

[4. O que é um Middleware? Qual a sua funcionalidade? 6](#_Toc138771748)

[5. Que serviços é que o Middleware tipicamente oferece? 7](#_Toc138771749)

[6. Quais são os objetivos do desenho de um Sistema Distribuído? 7](#_Toc138771750)

[7. Que PitFalls se destacam na construção de um sistema distribuído? 7](#_Toc138771751)

[8. Que tipo de sistemas de computação distribuída existem? Defina cada um deles. 7](#_Toc138771752)

[9. Quais são as 4 layers do grid computing? Qual a função de cada uma delas? 8](#_Toc138771753)

[Arquiteturas 8](#_Toc138771754)

[1. Defina arquiteturas de software 8](#_Toc138771755)

[2. Que tipos de arquiteturas de software existem? Defina-as. 8](#_Toc138771756)

[3. Que tipo de organizações de middleware existem? Defina-os. 9](#_Toc138771757)

[4. Defina arquitetura de sistema 10](#_Toc138771758)

[5. Defina os dois tipos de organizações centralizadas 10](#_Toc138771759)

[6. Defina os tipos de organizações descentralizadas(P2P). 10](#_Toc138771760)

[7. Defina os tipos de arquitetura híbrida 10](#_Toc138771761)

[8. Defina o Protocolo Chord 10](#_Toc138771762)

[Processos 10](#_Toc138771763)

[1. Diferencie Processos de Threads. 10](#_Toc138771764)

[2. Distinga o modelo many-to-one, one-to-one e many-to-many. 11](#_Toc138771765)

[3. Defina LightWeight Process. 11](#_Toc138771766)

[4. Qual a diferença entre dispatcher e workers? 11](#_Toc138771767)

[5. O que é a virtualização? 11](#_Toc138771768)

[6. Diferencie Processo Virtual Machine, Virtual Machine Nativa e Virtual Machine Hosted. 11](#_Toc138771769)

[7. Quais são os tipos de ligação cliente-servidor? 12](#_Toc138771770)

[8. Defina os três níveis de clusters de servidores. 12](#_Toc138771771)

[9. Distinga TCP Handoff e NAT. 12](#_Toc138771772)

[10. Distinga a ligação daemon da implementação de um super server? 12](#_Toc138771773)

[11. Determine as razões para migração de código. 13](#_Toc138771774)

[12. Que modelos de migração existem? 13](#_Toc138771775)

[13. StateFull vs StateLess ? 13](#_Toc138771776)

[Comunicações 13](#_Toc138771777)

[1. Quais são as camadas do modelo de referência da ISO (Pilha OSI)? 13](#_Toc138771778)

[2. Distinga mensagens persistentes de mensagens transientes? 13](#_Toc138771779)

[3. Defina Remote Procedure Call. 13](#_Toc138771780)

[4. Na passagem de parâmetros de métodos remotos, que possíveis problemas existem? Quais as soluções para esses problemas? 14](#_Toc138771781)

[5. Diferencie RPCs síncronos de assíncronos de derrefer synchronous RPC e de one-way RPC.](#_Toc138771782)

[6. Em que casos é que a comunicação orientada a mensagens pode ser a solução? 15](#_Toc138771783)

[7. O que são sockets? Como funcionam? 15](#_Toc138771784)

[8. Qual o conceito de ZeroMQ? 15](#_Toc138771785)

[9. O que é e para que é usado o sistema de filas? 15](#_Toc138771786)

[10. Qual a definição de um message broker? 16](#_Toc138771787)

[11. Quais são os tipos de organização de rede em Multicast ao nível da aplicação? 16](#_Toc138771788)

[12. Quais são os três parâmetros de avaliação dessa rede? 16](#_Toc138771789)

[13. Qual é a diferença entre Flooding e Flooding probabilístico? 16](#_Toc138771790)

[14. Defina os dois modelos de disseminação de informação, anti-entropy e rumor spreading? 16](#_Toc138771791)

[Designação (Naming) 17](#_Toc138771792)

[1. Defina nome, endereço e identificador. 17](#_Toc138771793)

[2. O que são o sistema de nomes: nomes planos? 17](#_Toc138771794)

[3. Explique como funciona o broadcasting neste sistema de nomes. 17](#_Toc138771795)

[4. Defina os Forwarding Pointers. 17](#_Toc138771796)

[5. O que é o Hierarquical Location Services ? 17](#_Toc138771797)

[6. Defina nomes estruturados. 18](#_Toc138771798)

[7. O que é um name space? 18](#_Toc138771799)

[8. Geralmente, quais são as camadas lógicas da organização hierárquica de um name space?](#_Toc138771800)

[9. Explicite o significado de attribute based naming. 18](#_Toc138771801)

[10. Para que serve o resource description framework (RDF)? 18](#_Toc138771802)

[11. Explique como funciona o protocolo LDAP. 18](#_Toc138771803)

[Coordenação 19](#_Toc138771804)

[1. Em que consiste o Universal Coordinated Time? 19](#_Toc138771805)

[2. O que entende por Clock Drift? 19](#_Toc138771806)

[3. O que é o Network Time Protocol? 19](#_Toc138771807)

[4. Como funciona o algoritimo de Berkeley? 19](#_Toc138771808)

[5. Como funciona a sincronização por difusão? 19](#_Toc138771809)

[6. Explique o funcionamento dos relógios de Lamport. 20](#_Toc138771810)

[7. Em que cenários os relógios lógicos não garantem consistência? 20](#_Toc138771811)

[8. Qual o problema dos relógios de Lamport em que os relógios vetoriais são a solução? 20](#_Toc138771812)

[9. O que são relógios vetoriais? 20](#_Toc138771813)

[10. A nível de exclusão mútua, que algoritmos existem? Explique o funcionamento de cada um deles. 20](#_Toc138771814)

[11. Quais são os algoritimos de eleição de um coordenador? Defina cada um deles. 21](#_Toc138771815)

[12. Que soluções para sistemas de localização existem? 21](#_Toc138771816)

[Consistência e Replicação 22](#_Toc138771817)

[1. Indique razões para replicação do sistema. Que problemas podem trazer? Como resolver esses problemas? 22](#_Toc138771818)

[2. Que modelos de consistência Data-Centric existem? Defina cada um deles. 22](#_Toc138771819)

[3. Que modelos de consistência Client-Centric existem? Defina cada um deles. 22](#_Toc138771820)

[4. Quais são as abordagens para descobrir quais os melhores locais para instalar os servidores? 23](#_Toc138771821)

[5. A nível de replicação de conteúdo, quais são as escolhas possíveis de um servidor alocar o conteúdo? 23](#_Toc138771822)

[6. Que formas existem para a distribuição de conteúdos? 23](#_Toc138771823)

[7. Justifique esta afirmação: “O tempo de leases deve ser dependente do comportamento do sistema”. 23](#_Toc138771824)

[8. Que tipo de protocolos de consistência existem? Explicite cada uma delas. 24](#_Toc138771825)

[Tolerância a Falhas 24](#_Toc138771826)

[1- Defina fiabilidade, disponibilidade, segurança e manutabilidade. 24](#_Toc138771827)

[2. Distinga falha por omissão de falha por comissão. 24](#_Toc138771828)

[3. O que entende por k-fault tolerant? 24](#_Toc138771829)

[4. Como funciona o consenso por flooding? 25](#_Toc138771830)

[5. Explicite o funcionamento do consenso realista (Paxos). Qual a sua vantagem em relação ao consenso por flooding? 25](#_Toc138771831)

[6. Explique o que é e para que foi criado o acordo Bizantino. 25](#_Toc138771832)

[7. Quais são os requisitos formais para consenso? 26](#_Toc138771833)

[8. Explique o teorema CAP. 26](#_Toc138771834)

[9. Qual a finalidade do módulo de deteção de falhas? 26](#_Toc138771835)

[10. Quais são as abordagens existentes para lidar com a perda de mensagens? 26](#_Toc138771836)

[11. Quais são as abordagens para um crash do cliente? 26](#_Toc138771837)

[12. Como funciona o protocolo 2PC? 27](#_Toc138771838)

[13. Que tipo de checkpoints existem e como funciona cada um deles? 27](#_Toc138771839)

[Perguntas extra 27](#_Toc138771840)

[1. O que é o Jinja2? 27](#_Toc138771841)

[2. O que é o Flask? 27](#_Toc138771842)

[3. O que é o Docker? 28](#_Toc138771843)

[4. O que é o Map Reduce? 28](#_Toc138771844)

[5. O que é o RabbitMQ? 28](#_Toc138771845)

[6. O que é o Redis? 28](#_Toc138771846)

[7. O que é HAProxy? 28](#_Toc138771847)

# Sistemas Distribuídos

## O que é um sistema distribuído?

Apesar de ainda não haver um acordo total sobre a definição de sistema distribuído, denota-se, para nossos propósitos, uma **coleção de elementos de computação autónomos** que aparentam, para os seus utilizadores, serem um **único sistema coerente**.

## Quais são as principais características de um sistema distribuído?

Uma das principais características é a **coleção de elementos de computação autónomos**. Isto significa que os nós de um sistema distribuído conseguem funcionar de forma independente, apesar de, se os nós se ignorarem uns aos outros, não faz sentido coloca-los no mesmo sistema.

Outra das principais características é um **sistema único coerente**. Um sistema distribuído deve ser reconhecido exteriormente como um sistema único coerente, ou seja, os utilizadores finais não devem perceber que estão perante um sistema distribuído ao longo da rede.

Um sistema é coerente quando **o conjunto de nós que o compõem operam da mesma forma, não importa onde, quando ou como a interação é efetuada**.

## Em questão de coordenação entre os vários nós de uma rede, qual a diferença entre um grupo aberto e fechado?

Em grupo aberto, qualquer nó tem permissão para se juntar ao sistema distribuído, significando que consegue enviar mensagens para qualquer nó do sistema.

Em grupo fechado, apenas os membros do grupo são capazes de comunicar entre si, sendo necessários mecanismos separados para um nó entrar e sair do grupo.

## O que é um Middleware? Qual a sua funcionalidade?

O Middleware é uma **camada de software acima do sistema operativo**, sobre a qual as aplicações trabalham. Esta camada é **comum** a todo o sistema. Criada para facilitar o desenvolvimento de um sistema distribuído, o Middleware **fornece os meios de comunicação entre aplicações distribuídas e diferentes aplicações e ainda esconde as diferenças de hardware e de sistema operativo**.

## Que serviços é que o Middleware tipicamente oferece?

Primeiramente, a **comunicação**, sendo o mais comum o **Remote Procedure Call**, que invoca uma função que é implementada e executada num computador remoto como se estivesse localmente disponível.

Segundo, oferece também **transações**, permitindo executar sobre a filosofia, All or Nothing.

Terceiro, a **composição de serviços**. Neste serviço, o middleware é responsável por estandardizar o acesso e fornecer meios de sincronização a aplicações que são criadas a partir da junção de outras.

Por fim, a **fiabilidade**, onde garante que, numa aplicação formada por um grupo de processos, uma mensagem é enviada a todos ou a nenhum.

## Quais são os objetivos do desenho de um Sistema Distribuído?

Garantir a fácil acessibilidade dos recursos.

Garantir a transparência.

Ser aberto.

Ser escalável.

## Que PitFalls se destacam na construção de um sistema distribuído?

A rede é fiável, segura e homogênea

A topologia não se altera

Latência é zero e largura de banda é infinita

Custo de transporte é zero

Existe um único administrador

## Que tipo de sistemas de computação distribuída existem? Defina cada um deles.

Um dos tipos é o **cluster computing**. Assume que o sistema é composto por um conjunto de máquinas semelhantes, que correm o **mesmo sistema operativo** e estão conectadas entre si por uma **rede local de alta velocidade**.

Outro é o **grid computing**. O sistema tem por base vários sistemas de computação sob **domínios administrativos diferentes** e que podem **variar** em aspetos de **hardware**, **software** e **tecnologia de rede**. Assim, é possível ter um **sistema híbrido** com nós especializados em aplicações intensivas, outros em gestão de ficheiros, armazenamento…

Surge ainda a **utility computing**. Com a criação dos seus **próprios datacenters**, as empresas passaram a dispor de recursos que não eram 100% utilizados. Sobre este conceito, o cliente pode carregar as suas tarefas para um datacenter e pagar pelos recursos utilizados para executar essas tarefas.

Por fim, a utility evoluiu para a **cloud computing**, um conjunto de recursos virtuais que podem ser utilizados e configurados dinamicamente. Neste conceito são então oferecidos três serviços:

1. **Infrastuture-as-a-service** (Datacentes, Amazon S3, Amazon EC2)
2. **Platform-as-a-service** (MS Azure, Google App engin)
3. **Software-as-a-service**(Google Docs, Gmail, Youtube, Flickr)

## Quais são as 4 layers do grid computing? Qual a função de cada uma delas?

Fabric layer – responsável por fornecer a interface aos utilizadores

Resource layer – responsável por gerir um recurso. Usa funções da connectivity layer e chama as interfaces feita pela fabric layer.

Connectivity layer – reponsável pela comunicação entre recursos, ou acesso a um remoto. Ainda responsável pela segurança de autenticação de utilizadores e recursos.

Collective layer – responsável pela gestão, alocamento e marcação de tarefas sobre recursos

# Arquiteturas

## Defina arquiteturas de software

Arquitetura de software é definido como os vários **componentes de software se organizam, interagem entre si** e como **juntos formam um sistema**.

## Que tipos de arquiteturas de software existem? Defina-as.

Primeiramente, a **arquitetura por camadas**. É organizada em componentes hierárquico, sendo a cada um deles possível invocar as camadas inferiores e, em casos excecionais, as camadas superiores em termos de hierarquia. Existe três tipos de abordagens:

1. **Implementação pura**, chamada é sempre feita ao componente inferior e assim sucessivamente
2. **Mixed Implementation**, chamada é feita a qualquer componente, desde que inferior
3. **Implementação com upcalls** – chamada é também feita a camadas superiores.

Segundo esta arquitetura, existirão em várias máquinas camadas correspondentes, que juntas oferecem um serviço, acedido através de interface, que implementam um protocolo. De forma geral, um sistema pode ser divido em três grandes camadas. Camada de interface, dados e processamento.

Segundo, existe a **arquitetura baseada em objetos e orientada a serviços**. Esta arquitetura trata os componentes abordados anteriormente como objetos, que estão conectados por mecanismos, procedimentos que podem ocorrer sobre uma rede. Quando o cliente faz um **bind** a um objeto distribuído é **carregada uma implementação da sua interface**, denominada por **proxy**, surgindo do lado do servidor uma entidade complementar, o **sketelon**.

Terceiro, a **arquitetura baseada em recursos**. Abordagem representada por **REST** (Representational State Transfer) permite que as aplicações adicionem ou modifiquem recursos facilmente, caracterizando-se por:

1. Os recursos têm um id único
2. Todos os recursos têm a mesma interface
3. Todas as mensagens são auo-contidas
4. Após uma operação, o componente esquece tudo

Por fim, a **Publish-Subscribe**. Admitindo um acoplamento temporal, mas não referencial, existe uma **abordagem baseada em eventos**. Os processos apenas podem executar duas operações: publicar e subscrever notificações. Quando um processo publicar uma notificação de determinado tipo, todos os processos já subscritos irão ser notificados através do Middleware (sendo necessário estarem todos em execução no momento, daí o acoplamento temporal).

**Sem acoplamento temporal e referencial** estamos perante um espaço de **memória partilhado.** Os processos comunicam através de **tuplos**, podendo escrever ou obtê-los. Permite ainda a subscrição de tuplos, sendo este verificados quanto à existência de novos.

## Que tipo de organizações de middleware existem? Defina-os.

**Wrappers**. Solução para interfaces que não são adequadas a determinada aplicação. Para reduzir o número de wrappers, foram criados os brokers, que é responsável por gerir os acessos entre as várias aplicações.

**Interceptors**. Quebra o fluxo normal de controlo e permite a execução de outro código. Sem perceção do midlleware é realizado um request-level interceptor antes do pedido chegar ao middleware e antes de chegar ao sistema operativo é realizado um message-level interceptor.

## Defina arquitetura de sistema

Arquitetura de sistema é a **interação entre componentes de software** e onde são **colocados em máquinas reais**.

## Defina os dois tipos de organizações centralizadas

**Cliente/Servidor** e **arquitetura multi-camadas** (interface do utilizador, processamento e dados).

## Defina os tipos de organizações descentralizadas(P2P).

**Sistemas estruturados** (Organização através de **distributed hash tables**).

**Sistemas não estruturados** (Grafos aleatórios**). Atualização permanente da lista de nós conhecidos através de flooding** (encaminhar pedido para todos os pedidos e aguardar resposta), ou **random walks**(enviar pedido a um dos vizinhos, menos recursos, mais tempo).

**Redes organizadas hierarquicamente** (Rede de super peers, cada um com weak peer (mediadores de comunicação)).

## Defina os tipos de arquitetura híbrida

**Sistemas edge-server**. Coloca servidores no limite da rede (entre as redes locais e a internet) aos quais os utilizadores se ligam.

**Sistema de distribuição colaborativa** (BitTorrent). Para garantir a colaboração, um nó só pode descarregar um ficheiro se estiver a disponibilizar ficheiros.

## Defina o Protocolo Chord

Chord é um protocolo para uma **P2P distributed hash table**. Especifica como é que as chaves estão **conectadas com os nós** e como u**m nó consegue descobrir o valor de determinada chave**.

# Processos

## Diferencie Processos de Threads.

Os **processos** são definidos, **em sistemas operativos**, como **programa em execução**. Estes são **criados pelo próprio sistema operativo**. Recorrem a muitos mais recursos e **não têm memória partilhada.**

As **threads** são “**mini-processos**” dentro de um processo com **execução independente**, **partilham memória** e **aumentam a performance de um multiprocessador**.

## Distinga o modelo many-to-one, one-to-one e many-to-many.

Modelo **many-to-one** – **Várias threads do utilizador** são mapeadas numa **única thread do núcleo**, sendo as chamadas de uma qualquer thread então bloqueantes.

Modelo **one-to-one** – Para **cada thread do utilizador**, há **uma thread do núcleo**. Aumenta o custo da implementação, uma vez que a mudança de contexto de cada thread pode ser tão exigetnte quanto a de um processo.

Modelo **many-to-many** – Mapeia **várias threads do utilizador em várias threads do núcleo**. Assim quando uma thread bloqueia, o núcleo pode agendar outra para execução.

## Defina LightWeight Process.

Este modelo consiste em dois espaços, o **lado do utilizador e o lado do Kernel**.

Do lado do utilizador, o sistema implementa uma biblioteca que permite a criação e a gestão de threads sem contexto e sem recurso ao núcleo. Cada biblioteca implementa vários LWP, sendo de forma escondida ao utilizador.

## Qual a diferença entre dispatcher e workers?

Dispatcher é a **thread responsável por receber os pedidos e encaminhá-los para os workers** **ativos**. Os workers, mesmo sendo bloqueantes, não interferem na execução do sistema. Estes são os responsáveis por **processar a informação necessária**.

## O que é a virtualização?

A virtualização define-se como a **extensão ou substituição de uma interface existente**, **oferecendo uma de outro sistema**. Permite que **um único computador** possa desempenhar a **função de vários através da partilha de recursos de hardware por vários ambientes**.

## Diferencie Processo Virtual Machine, Virtual Machine Nativa e Virtual Machine Hosted.

**Processo Virtual Machine** -> Aplica-se **apenas a um processo**. Sistema em tempo de execução que fornece um conjunto de instruções para ser utilizado pelas aplicações.

**Virtual Machine Nativa** -> Implementada **sobre o hardware**. Responsável pelo acesso à memória externa. Permite a execução de vários sistemas operativos na mesma máquina.

**Virtual Machine Hosted** -> Semelhante à nativa, mas **corre sobre um sistema operativo anfitrião**. Não é então responsável pelo acesso à memória externa. Executada apenas com permissões especiais.

## Disitinga a ligação daemon de implementação de um super server?

1. Colocar um daemon em execução em cada servidor, que contém o registo da porta atual dos vários serviços.
2. Implementar um superserver

## Defina os três níveis de clusters de servidores.

Logical Switch -> encaminha os pedidos do cliente

Application/Compute Servers

Distributes file/Database system

## Distinga TCP Handoff e NAT.

Em NAT é feita duas conexões TCP. Uma do cliente ao switch e outra do switch ao servidor. Switch funciona apenas como intermediário.

Com TCP Handoff, a conexão TCP cliente-switch é entregue ao servidor, sendo que este a utiliza para responder diretamente ao cliente. O servidor na resposta envia ainda um aclnowledge com o IP do switch para garantir integridade da conexão.

## Quais são as diferenças entre Round Robin e Content-aware (distribuição dos trabalhos do switch pelos servidores)?

Round Robin, o switch tem uma lista de servidores e entrega os pedidos de forma sequencial.

Content-aware, analisa os pedidos para saber a que serviço correspondem antes de os encaminhar.

## Determine as razões para migração de código.

Migração cliente-servidor -> processamento intenso dos dados

Migração servidor-cliente -> validação de formulários

## Que modelos de migração existem?

Iniciada pelo emissor, iniciada pelo recetor, de mobilidade fraca(apenas enviado o segmento de código) e de mobilidade forte(enviado o segmento de código e o de execução).

## StateFull vs StateLess ?

StateFul -> Guardam informação de sessão e permitem maior eficiência de comunicação, pois mantém informação de contexto.

StateLess -> Não guardam informação entre pedidos do cliente e evitam sobrecarga do servidor por necessidade de guardar informação e recuperar a mesma.

# Comunicações

## Quais são as camadas do modelo de referência da ISO (Pilha OSI)?

Application

Presentation

Session

Transport

Network

Data link

Physical

## Distinga mensagens persistentes de mensagens transientes?

Persistente -> É armazenado no middleware de comunicação até ser entregue ao recetor.

Transiente -> É armazenada no sistema de comunicação apenas enquanto ambos os emissor e recetor estiverem em execução.

## Defina Remote Procedure Call.

Tecnologia de comunicação que permite a programas invocar procedimentos localizados noutras máquinas, ocultando a troca de mensagens ao programador. Este processo é bloqueante, enquanto o processo invocador não receber resposta não realiza qualquer ação. Para garantir a transparência o cliente recorre a um Client Stub (encapsula os parâmetros passados como argumentos numa mensagem que é enviada para o servidor). No lado do servidor, existe um Server Stub, que recebe a mensagem e converte-a numa chamada a um método local. De seguida retorna ao stub, que por sua vez envia ao cliente stub, bloqueado até então.

Passos:

1. Cliente chama o Cliente Stub
2. Client Stub constói mensagem e chama o SO
3. O SO envia a mensagem para o SO remoto
4. O SO remoto entrega ao server stub
5. O Server Stub retira os parâmetros e invoca o método do servidor
6. O servidor executa o método e retorna ao server stub
7. O server stub contrói mensagem de retorno e chama o seu SO
8. O SO envia a mensagem ao SO do cliente
9. O SO do cliente entrega ao Client Stub
10. O Client Stub processa a mensagem e retorna ao cliente

## Na passagem de parâmetros de métodos remotos, que possíveis problemas existem? Quais as soluções para esses problemas?

A codificação dos parâmetros, a representação dos dados (little endian ou big endian) e o envio de ponteiros locais. A solução passa por tornar a informação independente da máquina a enviar.

## Diferencie RPCs síncronos de assíncronos de derrefer synchronous RPC e de one-way RPC.

RPCs assíncronos refere-se aquando do método é invocado pelo cliente não implicar uma resposta do servidor após o seu processamento.

RPC síncronos indica que o cliente bloqueia depois de enviar um pedido ao servidor até obter resposta deste.

RPCs síncronos adiado o cliente recebe uma mensagem de acknowledge do servidor e desbloqueia. Assim que terminar o processamento, o servidor envia uma nova mensagem, agora com o retorno da execução do método.

One-way RPC são assíncronos, mas com a particularidade de não esperarem pelo acknowledge do servidor.

## Em que casos é que a comunicação orientada a mensagens pode ser a solução?

Quando não pode ser assumido que ambos o emissor e o recetor estão ativos em simultâneo aquando da comunicação.

## O que são sockets? Como funcionam?

Sockets são definidos como um end point de comunicação, sendo reservados recursos para a troca de mensagens no protocolo especificado.

As operações possíveis são:

1. Socket -> cria um end point de comunicação, reservando recursos
2. Bind -> associa um endereço local à socket
3. Listen -> informa o sistema operativo do número de buffers a reservar
4. Accept -> bloqueia o servidor até que seja feito um pedido, momento em que é criada uma nova socket para o processar

Para o cliente é usado:

1. Connect -> bloqueia o cliente e tenta estabelecer conexão ao endereço fornecido
2. Send -> envia dados através da ligação
3. Receive -> recebe dados através da ligação
4. Close -> liberta a ligação

## Qual o conceito de ZeroMQ?

Biblioteca de troca de mensagens assíncronas de ato nível orientada à conexão que fornece uma camada de abstração para sockets.

Permite que uma socket seja associada a várias portas da máquina. Suporta many-to-one e ainda one-to-many e multicasting.

## O que é e para que é usado o sistema de filas?

O sistema de filas foi criado para garantir as comunicações persistentes assíncronas. Oferecem capacidade de armazenamento de mensagens sem a necessidade de o emissor ou o recetor estarem ativos. Os servidores têm uma fila para cada aplicação cuja leitura é exclusiva, mas a submissão de mensagens é acessível a todas.

A transparência é assiom garantida.

Put -> adiciona uma mensagem à fila especificada

Get -> Bloqueia até que haja uma mensagem na fila, remove a primeira

Poll -> Verifica a existência de mensagens numa dada fila, remove a primeira, mas não é bloqueante

Notify -> É chamada quando uma mensagem for colocada na fila

## Qual a definição de um message broker?

Um message broker essencialmente simplifica as conversões entre mensagens, para que o destinatário as possa compreender. Para além disto, pode ainda atuar como um application gateway, ou ter associadas funcionalidades de roteamenteo ao disponibilizar um modelo publish-subscribe.

## Quais são os tipos de organização de rede em Multicast ao nível da aplicação?

Em árvore -> Caminho único entre cada par de nós

Em teia (mesh) -> Cada nós tem vários vizinhos, requerendo algum tipo de routing

## Quais são os três parâmetros de avaliação dessa rede?

Link stress -> consiste na contagem do número de pacotes com o mesmo conteúdo que passam pela mesma ligação

Stretch ou Relative Delay Penalty -> relaciona o atraso (custo) entre dois nós na rede overlay e na rede subjacente

Tree cost -> métrica global fortemente relacionada com a minimização dos custos agregados

## Qual é a diferença entre Flooding e Flooding probabilístico?

Flooding trata-se de encaminhar cada mensagem para todos os nós vizinhos. Flooding probabilitico trata-se de encaminhar a mensagem para os vizinhos com uma determinada probabilidade.

## Defina os dois modelos de disseminação de informação, anti-entropy e rumor spreading?

Anti-entropy -> Um nó P escolhe um nó Q aleatoriamente para trocar informação

Rumour Spreading -> P atualizado escolhe um nó Q para informar da atualização e se este já tiver sido atualizado por outro, P para de propagar a atualização com uma determinada probabilidade.

## 

# Designação (Naming)

## Defina nome, endereço e identificador.

Um nome refere-se a uma entidade. Um endereço refere-se a um nome utilizado para referência de um ponto de acesso de uma entidade. Um identificador é um valor associado a uma e só uma entidade.

## O que são o sistema de nomes: nomes planos?

São identificadores strings de tamanho fixo, também com o nome de nomes não estruturados. Podem ser manipulados eficientemente por máquinas.

## Explique como funciona o broadcasting neste sistema de nomes.

É enviada uma mensagem em Broadcast para todas as máquinas. A que corresponder responde com o endereço do acess point. Porém esta solução não é escalável, quanto mais a rede crescer mais distúrbios na rede irá causar. Uma alternativa passa por multicasting, criação de determinados grupos para recebrrem essas mensagens.

## Defina os Forwarding Pointers.

É definido com base no princípio em que quando uma entidade se move, deixa uma referência à sua nova localização. Conhecendo o antigo endereço é possível relocalizá-lo.

## O que é o Hierarquical Location Services ?

É definido como um esquema de organização hierárquica com uma divisão de rede com os seguintes domínios: um raíz, que abrange toda a rede e se divide em múltiplos subdomínios menores até aos leaf domains (terminais). A cada domínio está associado um diretório de nós.

## Defina nomes estruturados.

Os nomes estruturados caracterizam-se pela simplicidade e facilidade de leitura para os humanos.

## O que é um name space?

Gerlamente na forma de um grafo orientado, um name space é uma maneira de organizar os nomes do sistema. Este grafo geralmente contém dois tipos de nós:

Nós folha -> representam e armazenam informação relativa a uma entidade

Nós intermédios -> Contém uma directory table com entradas (nó, etiqueta do vértice)

## Geralmente, quais são as camadas lógicas da organização hierárquica de um name space?

Global -> parte da raíz e por directory nodes próximos da mesma, carazterizados pela sua estabilidade

Administrativa -> composta por nós próximos e geridos por uma organização, representando as suas entidades

Managerial -> nós que mudam com frequência, mantidos por administradores da rede e por utilizadores finais.

## Explicite o significado de attribute based naming.

Consiste na descrição de entidades como pares (atributo,valor).

## Para que serve o resource description framework (RDF)?

Unificar a forma de descrever entidades. Estabelece a descrição de uma entidade como um tripleto (Sujeito, Predicado, Objeto).

## Explique como funciona o protocolo LDAP.

O LDAP é um protocolo para a implementação de um naming system. Este serviço tem por base um conjunto de registos, designados por uma sequência de atributos de nome, relative distinguished name (RDN). Ao listar estes RDNs de forma sequencial, é criada a directory information tree (DIT), um grafo de nomes de um diretório. Ao conjunto de entradas de um serviço LDAP, chamamos directory information base (DIB). O LDAP distingue-se do DNS por fornecer opções de pesquisa muito mais abrangentes e pouco específicas.

# Coordenação

## Em que consiste o Universal Coordinated Time?

Consiste num standard mundial para a definição do tempo real e é implementado através de 40 antenas de sinal rádio que transmitem um pulsar a cada segundo de UTC.

## O que entende por Clock Drift?

Clock drift são erros externos de sincronização de relógio, como a temperatura.

Precisão (Sincronização Interna) -> desvio entre o relógio de duas máquinas dentro de um limite pi

Exatidão (Sincronização Externa) -> desvio entre o relógio e o tempo UTC

## O que é o Network Time Protocol?

É um protocolo de troca de mensagens entre o servidor de tempo e o cliente.

## Como funciona o algoritimo de Berkeley?

Um algoritimo de sincronização de relógios em máquinas sem recetor UTC. O servidor atua como entidade ativa, que contacta todas as máquinas para saber o seu tempo, faz a média e responde a cada um informando a quantidade de atraso ou adiantamento.

## Como funciona a sincronização por difusão?

Este algoritimo também não tem noção do tempo real, apenas máquinas do sistema. Procura apenas sincronizar os recetores com o tempo do emissor. Um dos protocolos que implementa é o RBS (Reference Broadcast Syncronization) que assenta no princípio da não medição dos tempos de processamento. O servidor envia uma mensagem a cada nó, sendo gravado o tempo em que recebe a mensagem. Os nós comparam os tempos e determinam o seu offset usando a média. A ideia é comparar a diferença de tempo de receção entre as máquinas e não a hora do servidor.

## Explique o funcionamento dos relógios de Lamport.

Os relógios de Lamport são relógios lógicos, isto é, abstraem-se da noção do tempo e focam-se apenas na ordem pela qual os eventos ocorrem. A sincronização destes relógios é feita com base no princípio de que nenhum processo pode receber mensagens do futuro. O contador local do processo incrementa x unidades a cada intervalo de tempo e sempre que recebe uma mensagem com um valor superior ao seu contador atual avança para o valor recebido mais uma unidade.

## Em que cenários os relógios lógicos não garantem consistência?

Em cenário de servidores implementados de forma distribuída, caso haja acesso a dois servidores distintos, sendo estes réplicas, as operações feitas pelo cliente serão assumidas como anteriores causando inconsistência.

## Qual o problema dos relógios de Lamport em que os relógios vetoriais são a solução?

Devido às disparidades nos incrementos dos valores dos seus relógios, nada nos garante acerca da ordem real em que estas foram enviadas e por consequência se existe uma relação de causalidade. Problemas que desaparecem com os relógios vetoriais.

## O que são relógios vetoriais?

Os relógios vetoriais consistem em pares ordenados onde são armazenados os valores dos tempos dos demais processos, fazendo corresponder um índice a cada processo

## A nível de exclusão mútua, que algoritmos existem? Explique o funcionamento de cada um deles.

Algoritimo centralizado -> Apenas um coordenador que gere o acesso aos recursos de todos os processos. Quando um processo quer aceder envia um REQUEST e recebe um OK caso o recurso esteja livre ou não recebe resposta caso esteja a ser utilizado. Quando um processo não está a utilizar mais um recurso envia um RELEASE.

Algoritmo distribuído -> Com base nos relógios de Lamport, este algoritmo não contém nenhum coordenador. Quando um processo quer aceder a um recurso envia um pedido a todos os processos. Caso determinado processo não esteja a utilizar envia um OK. Caso esteja a utilizar, não responde armazenado o pedido na fila. Caso esteja à espera do acesso, responde OK ou não responde dependendo do timestamp da mensagem. Só com um OK de todos os processos é que o processo pode aceder ao recurso. Pode gerar problemas caso um processo crashe, comprometendo todos os processos a partir daí.

Token Ring -> É formado um anel entre todos os processos. Quando é inicializado um token é entregue ao primeiro processo e caso este necessite do acesso acede, caso contrário passa para o próximo processo e assim sucessivamente. Caso um processo crashe quando está com um token pode gerar problemas.

## Quais são os algoritimos de eleição de um coordenador? Defina cada um deles.

Eleição por bullying -> Quando um processo inicia uma eleição envia um ELECTION para todos os processos com identificadores maiores. Caso nenhum responda o processo emissor passa a ser coordenador. Caso haja resposta, o processo que respondeu passa a assumir o processo de eleição.

Eleição num anél -> Consiste também na criação de um anél lógico entre todos os processos. Quando inicia um processo encia uma ELECTION para o seu sucessor com o seu PID e assim sucessivamente. Quando a mensagem regressa ao emissor, este identifica o coordenador e muda o tipo de mensagem para COORDINATOR e volta a encaminhar para o seu sucessor. Quando regressar novamente a mensagem é descartada. Pode ocorrer vários processos de eleição em paralelo e existe a vantagem de não existir a necessidade de todos os processos se conhecerem.

Eleições em sistemas de grande escala -> Consiste em criar partições na rede, tornando-la mais robusta e menos suscetível a falhas. Existe super peers distribuídos uniformemente pela rede. A eleição é realizada baseada nas leis da termodinâmica, nomeadamente as forças de repulsão. OS super nós contactam-se entre si enviando mensagem com a latência associada.

Caso a latência seja baixa, os nós estão muito próximos e o que tiver o ID mais baixo deve de procurar um nó que ainda não é super nó para lhe passar a sua função.

## Que soluções para sistemas de localização existem?

GPS -> Sistema distribuído sustentado por 72 satélites em órbitra que fazem Broadcast contínuo da sua posição e tempo.

Alternativas Indoor -> Múltiplos pontos de acesso WiFi disponíveis ou Bluetooth

Posição lógica -> Espaço geométrico m-dimensional, onde a distância entre dois nós reflete a latência entre ambos.

# Consistência e Replicação

## Indique razões para replicação do sistema. Que problemas podem trazer? Como resolver esses problemas?

Uma das razões para a replicação do sistema é para aumentar a fiabilidade (prevenção de falhas em caso de avaria ou de ficheiros corrompidos), sendo outra o aumento da performance (aumenta com a divisão do trabalho pelas várias máquinas).

Um dos maiores problemas com a replicação é a consistência dessas mesmas réplicas. Para garantir consistência é necessário garantir que as operações de escritas são executadas como sendo uma única operação atómica em todos os nós do sistema.

## Que modelos de consistência Data-Centric existem? Defina cada um deles.

Consistência contínua -> Vários graus de consistência – Valor numérico, estabilidade entre réplicas e ordenação das atualizações. Medição de consistência é feita com a conit. O número de operações com commit pendentes chama-se order deviation.

Consistência sequencial -> Quando o resultado de qualquer execução é o mesmo se todas as operações em todos os processos forem executadas na mesma ordem sequencial.

Consistência causal -> Escritas que estão relacionadas causisticamente têm de ser vistas em todos os processos pela mesma ordem. Escritas concorrentes não têm de seguir qualquer ordem.

## Que modelos de consistência Client-Centric existem? Defina cada um deles.

Leituras monotónicas -> Quando um processo ler um valor de uma variável, qualquer operação de leitura posterior pelo mesmo processo devolve o mesmo valor ou um mais recente.

Escritas monotónicas -> Quando uma operação de escrita por um processo numa variável é completada antes de qualquer outra escrita nessa variável pelo mesmo processo.

Leitura após escrita -> O efeito da operação de escrita por um processo num determinado dado, será sempre visto pelas operações de leitura sucessivas nesse dado nesse processo.

Escrita após leitura -> Um processo que escreva num determinado dado e o tenha lido anteriormente, faz essa escrita sobre o mesmo valor lido ou um mais recente.

## Quais são as abordagens para descobrir quais os melhores locais para instalar os servidores?

1. Selecionar a melhor localização entre N possíveis, para a qual a média de distância dos clientes é mínima
2. Selecionar os K maiores sistemas autónomos e colocar um servidor em cada um
3. Posicionar os nós num espaço d-dimensional, onde a distância reflete a latência, identificando as K regiões com densidade mais elevada e colocando um servidor em cada uma

## A nível de replicação de conteúdo, quais são as escolhas possíveis de um servidor alocar o conteúdo?

Permanente -> É escolhido um servidor de forma estável e invariável no tempo

Iniciada pelo servidor -> Quando um processo hospeda dinamicamente uma réplica a pedido do servidor.

Iniciada pelo cliente -> Quando um processo hospeda dinamicamente uma réplica a pedido do cliente.

## Que formas existem para a distribuição de conteúdos?

Propagação apenas da notificação, invalidando uma atualização

Replicação passiva -> Transferir dados de uma cópia para a outra

Replicação ativa -> Propagar operações de atualização para outras cópias

## Justifique esta afirmação: “O tempo de leases deve ser dependente do comportamento do sistema”.

Os tempos de leases devem ser baseados em três critérios:

1. Idade -> Se um objeto é alterado ou não há algum tempo. Se já não for atualizado há muito tempo o tempo de lease deve ser longo
2. Frequência -> Se um cliente pede constantemente um objeto ou não. Se pede, o tempo de lease deve ser longo
3. Estado -> Se existe muita carga do servidor. Quanto mais carga, menor deve ser o tempo.

## Que tipo de protocolos de consistência existem? Explicite cada uma delas.

Protocolos baseados num primário -> Contém uma réplica primária que centraliza todas as operações de escrita e as trata sequencialmente. Solução simples e fácil de implementar.

Protocolos baseados num primário com escritas locais -> Otimizando o anterior, este protocolo permite a migração de cópias primárias para a réplica mais próxima tornando assim possível as escritas de forma local. Assume-se que o cliente faz mais escritas.

Protocolos de escrita replicada -> Todas as réplicas voltam para validar operações de leitura e escrita, sendo necessário um quórum para serem efetivadas. Valor de quórum:

Nr + Nw > N

Nw > N/2

# Tolerância a Falhas

## Defina fiabilidade, disponibilidade, segurança e manutabilidade.

Fiabilidade -> Disponibilidade contínua de um serviço, isto é, probabilidade condicional do sistema funcionar corretamente num intervalo, sabendo que no início estava a funcionar corretamente.

Disponibilidade -> Prontidão para ser utilizado.

Segurança -> Baixa probabilidade de catástrofes associadas à dependência

Manutabilidade -> Facilidade de reparar o sistema

## Distinga falha por omissão de falha por comissão.

Falha por omissão -> componente não realizou uma ação que devia ter realizado.

Falha por comissão -> componente realizou uma ação que não devia ter realizado.

## O que entende por k-fault tolerant?

Dizem-se k-fault tolerant processo que conseguem mascarar falhas em k componentes, sendo k o grau de tolerância a falhas.

Para falhas terminais precisamos de k+1 membros.

Para falhas arbitrárias precisamos de 2k+1 membros.

## Como funciona o consenso por flooding?

Um grupo de processos com deteção fiável de falhas comunica através de flooding, mantendo uma lista de comando propostos e atingindo consenso assim:

1. Processo envia por multicast o seu conjunto de comandos propostos até ao momento
2. Depois de receber os conjuntos de todos os outros processos, cada processo funde todos os comandos recebidos num novo conjunto
3. O próximo comando é selecionado por uma função determinística partilhada globalmente

## Explicite o funcionamento do consenso realista. Qual a sua vantagem em relação ao consenso por flooding?

O consenso realista é feito pelo algoritmo de paxos, que tem por base um sistema parcialmente síncrono, cuja comunicação pode ser instável, mas com a possibilidade de deteção de mensagens corrompidas. Cada processo assume os papéis proposer, learner e acceptor.

Primeiramente existem clientes que solicitam pedidos de uma operação. No lado do servidor, cada cliente é representado por um proposer que tenta convencer que a solitiação seja aceite. O objetivo é estabelecer que uma proposta seja aceite por um acceptor. Se a maioria dos acceptors aceitar, a proposta avança. No entanto é ainda preciso a proposta ser “aprendida”. Para isso, teremos vários processos denominados de learners, cada um dos quais executará uma proposta escolhida assim que for informado pela maioria dos acceptors.

Neste algoritmo todas as operações são determinísticas e os processos podem exibir falhas por crash, mas nunca arbitrárias.

O flooding não é uma solução realista para o consenso, não só devido às exigências de troca intensa de mensagens, mas porque temos decisões que não são tomadas porque nem todos recebem a mesma comunicação.

## Explique o que é e para que foi criado o acordo Bizantino.

O acordo Bizantino foi criado para garantir a consistência em falhas arbitrárias. O acordo Bizantino preza os seguintes requisitos:

1. Todos os backups que não estejam a falhar armazenam o mesmo valor
2. Se o primário não estiver em falha então todos os backups que não estejam a falhar armazenam exatamente o que o primário enviar.

## Quais são os requisitos formais para consenso?

Processos produzem o mesmo valor

Todos os resultados são válidos

Todos os processos têm que eventualmente fornecer um resultado

## Explique o teorema CAP.

O teorema CAP diz-nos que qualquer sistema que partilhe informação só pode ter duas das três propriedades:

1. Consistência -> qualquer informação partilhada e replicada aparece com um único valor
2. Disponibilidade -> garante que uma atualização será sempre executada
3. Partição do grupo será sempre tolerada

O corolário diz-nos que, numa rede com falhas de comunicação é impossível realizar uma operação em memória partilhada que garanta uma resposta a todos os pedidos.

## Qual a finalidade do módulo de deteção de falhas?

Este módulo server para detetar se um processo crashou ou não.

## Quais são as abordagens existentes para lidar com a perda de mensagens?

1. Desenhar o servidor com operações idempotentes (leituras puras e operações de substituição restritas)
2. Incluir número sequencial nas mensagens dos clientes
3. Adicionar um bit para distinguir pedidos originais de repetidos

## Quais são as abordagens para um crash do cliente?

1. Matar o órfão pelo cliente quando recupera
2. Quando servidor recebe novo número de época do cliente quando este recupera, mata todos os seus órfãos
3. Estabelecer tempo máximo para computações, sendo removidas as que ultrapassem este limite.

## Como funciona o protocolo 2PC?

1. O coordenador envia VOTE-REQUEST aos participantes
2. Os participantes respondem com VOTE-COMMIT ou VOTE-ABORT
3. Se recebeu commit de todos, o coordenador faz um GLOBAL-COMMIT, caso contrário faz um GLOBAL-ABORT
4. Caso haja GLOBAL-COMMIT, os participantes efetivam a operação.

## Que tipo de checkpoints existem e como funciona cada um deles?

Checkpoint coordenado:

1. Coordenador envia em multicast mensagem com pedido de checkpoint
2. Participantes fazem checkpoint e respondem com acknowledge
3. Coordenador quando receber mensagem de todos os participantes envia checkpoint done

Checkpoint independente:

1. Processo faz um checkpoint
2. Quando envia mensagem, envia com o intervalo onde fez
3. O processo recetor, recebe no seu intervalo e cria as dependências de intervalo
4. O processo recetor cria um checkpoint associado à dependência

# Perguntas extra

## O que é o Jinja2?

Mecanismo de template em Python para front-end

## O que é o Flask?

Framework em python para a web.

## O que é o Docker?

Plataforma que permite a criação, o empacotamento e a execução de aplicações em ambiente isolados, denominados de containers.

## O que é o Map Reduce?

Paradigma de programação que permite processar grandes quantidades de informação de forma paralela.

## O que é o RabbitMQ?

Software de message-broker de código aberto.

## O que é o Redis?

Projeto de estrutura de dados em memória de forma distribuída.

## O que é HAProxy?

Load balance rand proxy server para aplicações TCP e HTTP-based.