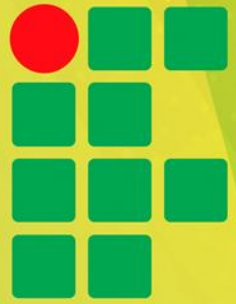
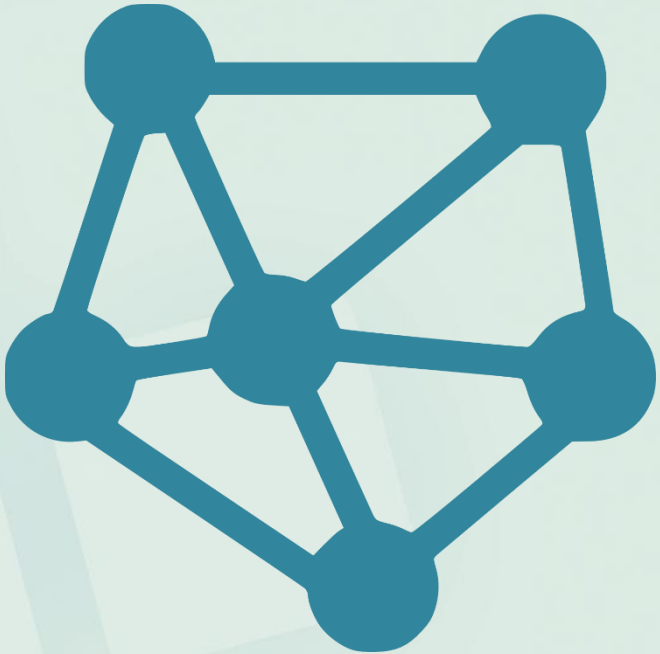


INSTITUTO FEDERAL

Sertão Pernambucano

Sistemas Distribuídos

Compartilhamento de Recursos e Desafios

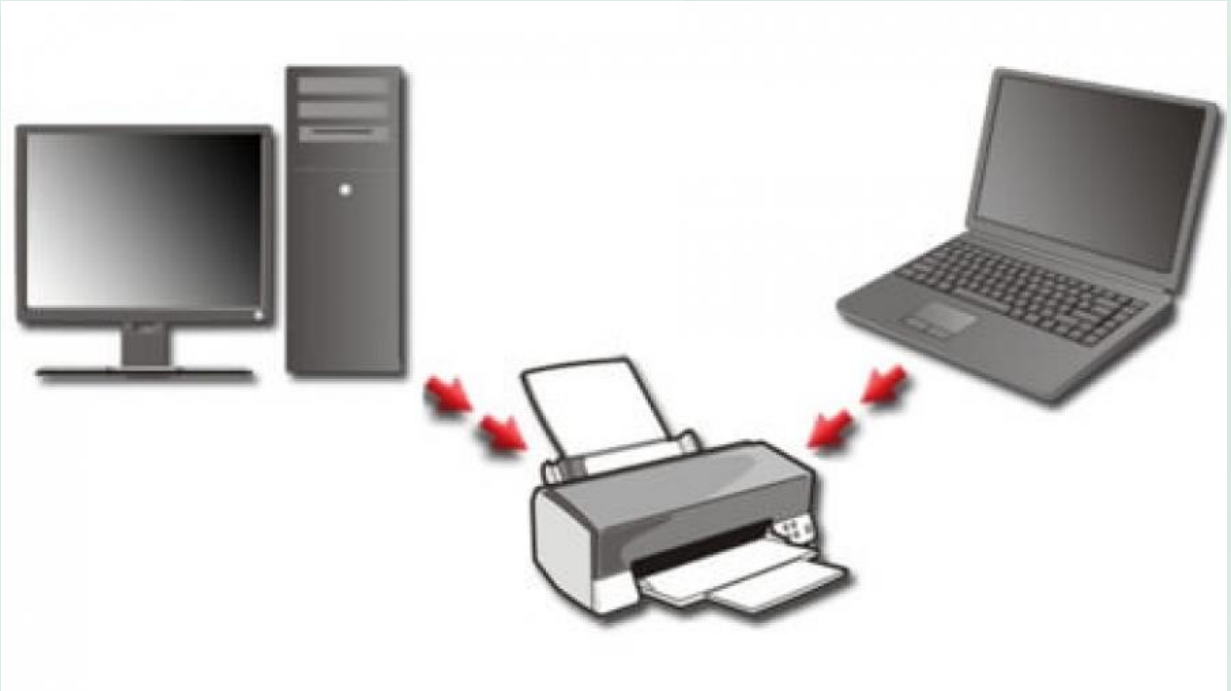


Prof. Heraldo Gonçalves Lima Junior

INSTITUTO FEDERAL
Sertão Pernambucano

1. Enfoque no compartilhamento de recursos

1.1. Compartilhamento de recursos



1.1. Compartilhamento de recursos

1.1. Compartilhamento de recursos

1.2. Serviço

- O termo serviço é usado para designar uma parte distinta de um sistema computacional que gerencia um conjunto de recursos relacionados e apresenta sua funcionalidade para usuários e aplicativos.

1.2. Serviço

- Em um sistema distribuído, os recursos são fisicamente encapsulados dentro dos computadores e só podem ser acessados a partir de outros computadores por intermédio de mecanismos de comunicação.

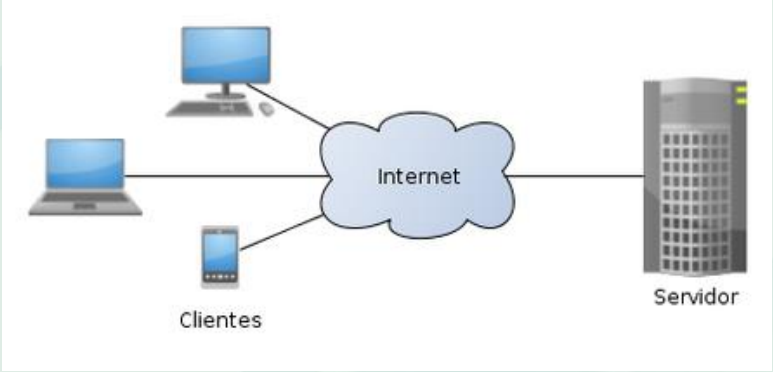
1.3. Servidor

- O termo servidor se refere a um programa em execução (um processo) em um computador interligado em rede, que **aceita pedidos** de programas em execução em outros computadores para **efetuar um serviço e responder apropriadamente**.

1.3. Servidor

1.3. Clientes

- Os processos que realizam os **pedidos** são referidos como clientes e a estratégia geral é conhecida como computação cliente-servidor.



1.3. Clientes

- Quando o cliente envia um pedido para que uma operação seja efetuada, dizemos que o cliente **requisita uma operação no servidor**.
- Uma interação completa entre um cliente e um servidor chamada de **requisição remota**.

2. Desafios

2.1. Heterogeneidade

- A Internet permite aos usuários acessarem serviços e executarem aplicativos por meio de um conjunto heterogêneo de computadores e redes.

2.1.1. Heterogeneidade: Hardware

- Os tipos de dados, como os inteiros, podem ser representados de diversas maneiras em diferentes tipos de hardware;

2.1.2. Heterogeneidade: Sistemas Operacionais

- Embora os sistemas operacionais de todos os computadores na Internet precisem incluir uma implementação dos protocolos Internet, nem todos fornecem, necessariamente, a mesma interface de programação de aplicativos para esses protocolos.

2.1.3. Heterogeneidade: Redes

- Embora a Internet seja composta de muitos tipos de redes, suas diferenças são mascaradas pelo fato de que todos os computadores ligados a elas utilizam protocolos Internet para se comunicar.

2.1.4. Heterogeneidade: Linguagens de Programação

- Diferentes linguagens de programação usam diferentes representações para caracteres e estruturas de dados, como vetores e registros. Essas diferenças devem ser consideradas, caso programas escritos em diferentes linguagens precisem se comunicar.

INSTITUTO FEDERAL
Sertão Pernambucano

2.1.5. Heterogeneidade: Middleware

- O termo middleware se aplica a uma camada de software que fornece uma abstração de programação, assim como o mascaramento da heterogeneidade das redes, do hardware, dos sistemas operacionais e das linguagens de programação subjacentes.

2.1.6. Heterogeneidade: Implementação de Diferentes Desenvolvedores

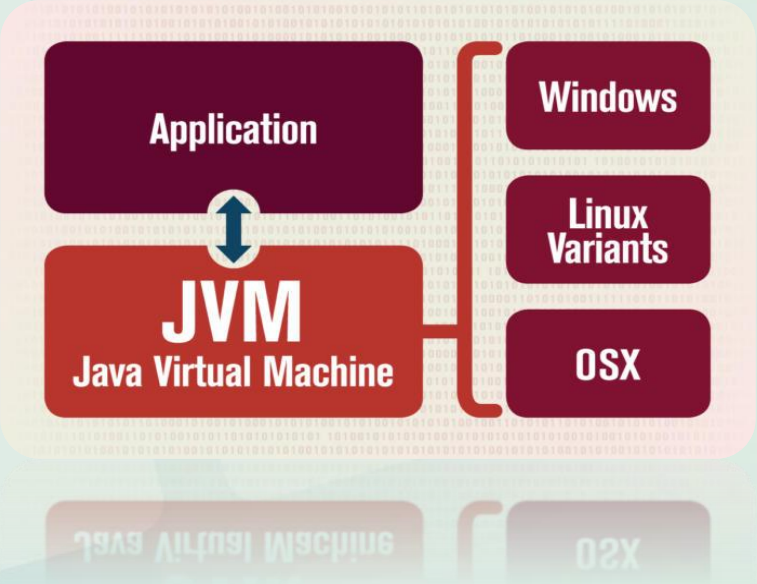
- Os programas escritos por diferentes desenvolvedores não podem se comunicar, a menos que utilizem padrões comuns.

2.1.7. Heterogeneidade: Migração de Código

- O termo migração de código, ou ainda, código móvel, é usado para se referir ao código de programa que pode ser transferido de um computador para outro e ser executado no destino.

2.1.7. Heterogeneidade: Migração de Código

- Sua aplicação roda sem nenhum envolvimento com o sistema operacional, sempre conversando apenas com a **Java Virtual Machine (JVM)**.



2.1.7. Heterogeneidade: Migração de Código

- Atualmente, a forma mais usada de código móvel é a inclusão de programas Javascript em algumas páginas Web carregadas nos navegadores clientes.

2.2. Sistemas Abertos

- Diz-se que um sistema computacional é aberto quando ele pode ser estendido e reimplementado de várias maneiras.

2.2. Sistemas Abertos

- Os sistemas abertos são caracterizados pelo fato de **suas principais interfaces serem publicadas**;
- Os sistemas distribuídos abertos são baseados na estipulação de um mecanismo de **comunicação uniforme** e em **interfaces publicadas para acesso aos recursos com partilhados**.

2.2. Sistemas Abertos

- Os sistemas distribuídos abertos podem ser construídos a partir de **hardware e software heterogêneo**, possivelmente de diferentes fornecedores. Para que um sistema funcione corretamente, a compatibilidade de cada componente com o padrão publicado deve ser cuidadosamente testada e verificada.

2.2. Sistemas Abertos

- O maior desafio para os projetistas é encarar a complexidade de sistemas distribuídos compostos por muitos componentes e elaborados por diferentes pessoas.

2.3. Segurança

- A segurança de recursos de informação tem três componentes:

CONFIDENCIALIDADE

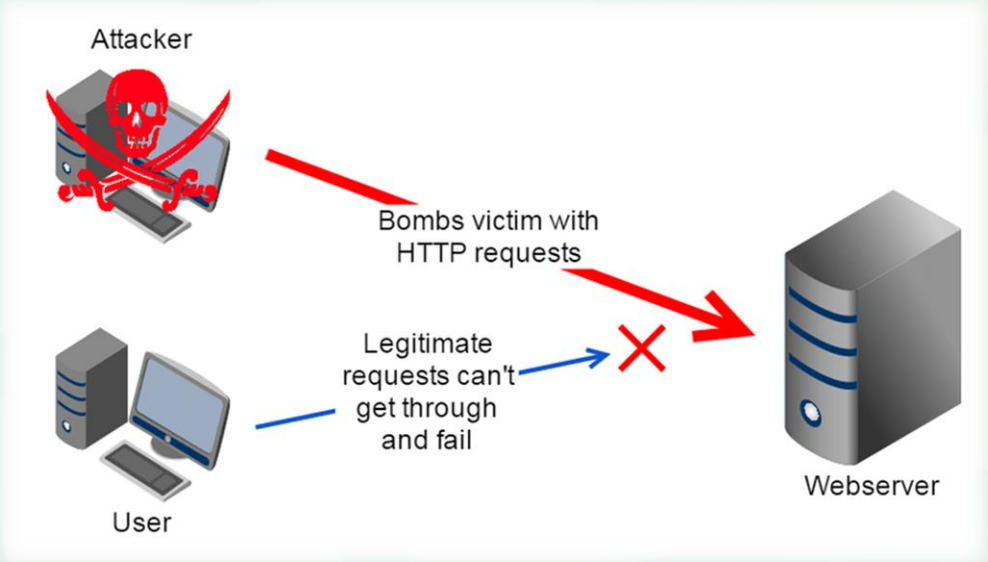
INTEGRIDADE

DISPONIBILIDADE

2.3.1. Desafios de Segurança: Ataque de negação de serviço (Denial of Service)

- Ocorre quando um usuário interrompe um serviço por algum motivo. Isso pode ser conseguido bombardeando o serviço com um número tão grande de pedidos sem sentido, que os usuários sérios não são capazes de utilizá-lo.

2.3.1. Desafios de Segurança: Ataque de negação de serviço (Denial of Service)



2.3.1. Desafios de Segurança: Segurança de código móvel:

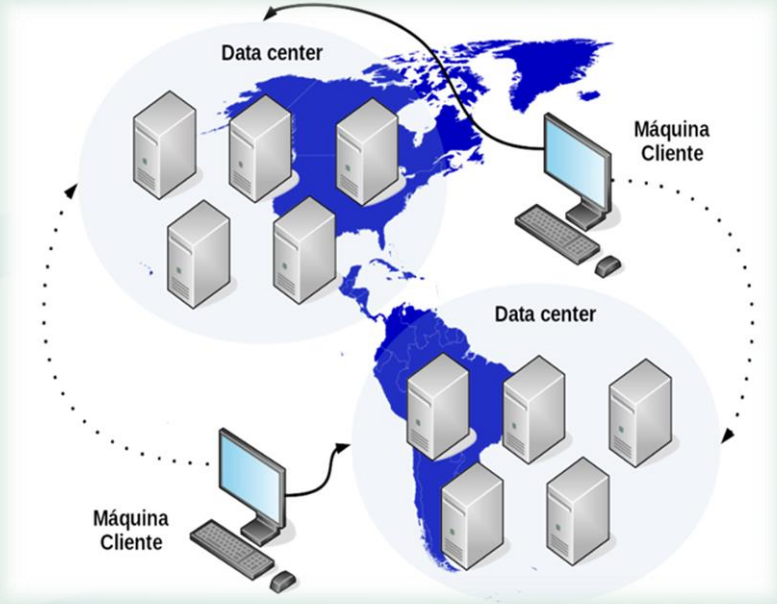
- Um código móvel precisa ser manipulado com cuidado. Considere alguém que receba um programa executável como um anexo de correio eletrônico: os possíveis efeitos da execução do programa são imprevisíveis.

2.4. Escalabilidade

- Os sistemas distribuídos funcionam de forma efetiva e eficaz em **muitas escalas diferentes**, variando desde uma pequena intranet até a Internet.

2.4. Escalabilidade

- Um sistema é descrito como escalável se **permanece eficiente** quando há um aumento significativo no número de recursos e no número de usuários.



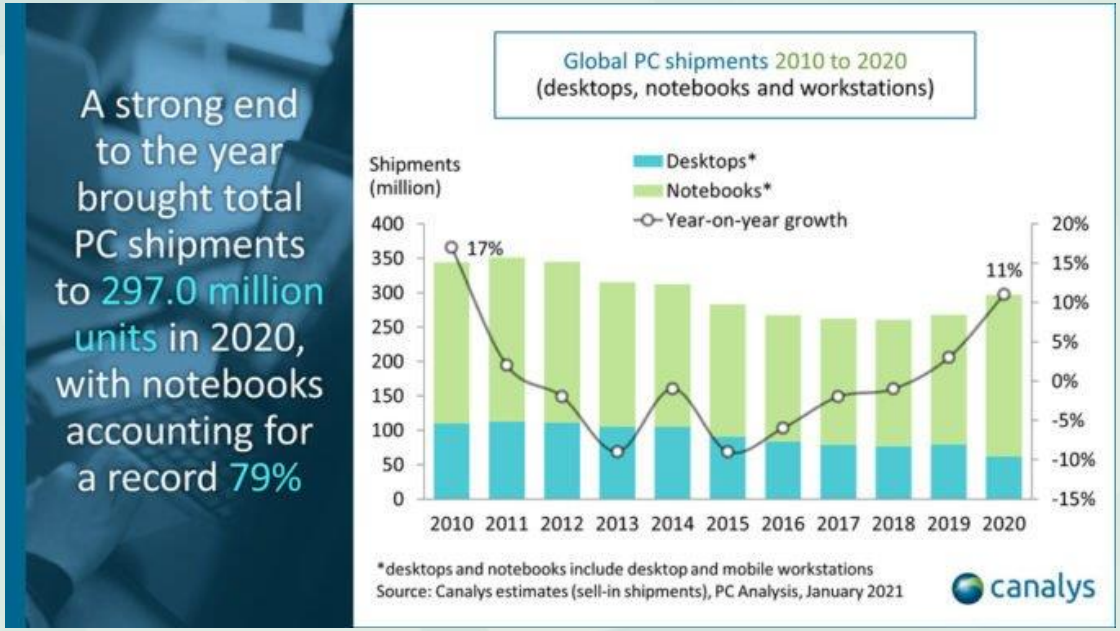
2.4.1 Desafios da Escalabilidade: Controlar o custo dos recursos físicos

- À medida que a demanda por um recurso aumenta, deve ser possível, a um custo razoável, ampliar o sistema para atendê-la. Por exemplo, a frequência com que os arquivos são acessados em uma intranet provavelmente vai crescer à medida que o número de usuários e de computadores aumentar.

2.4.1 Desafios da Escalabilidade: Controlar o custo dos recursos físicos

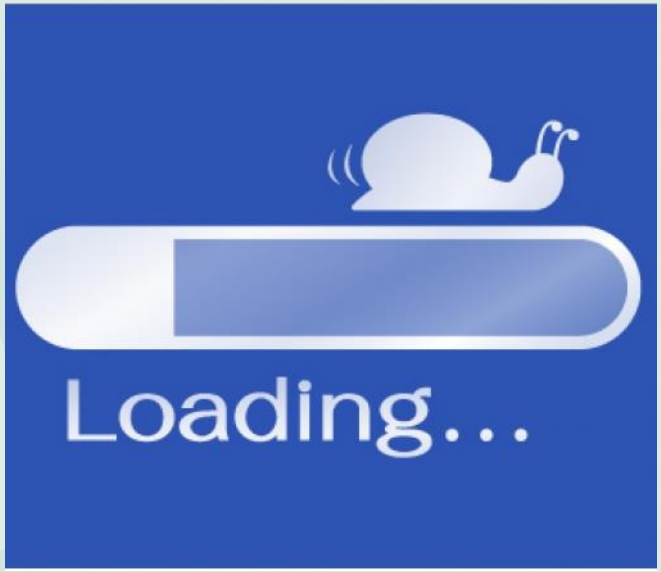
- Deve ser possível adicionar servidores de arquivos de forma a evitar o gargalo de desempenho que haveria caso um único servidor de arquivos tivesse de tratar todos os pedidos de acesso a arquivos.

2.4.1 Desafios da Escalabilidade: Controlar o custo dos recursos físicos



2.4.2 Desafios da Escalabilidade: Controlar a perda de desempenho

- Considere o gerenciamento de um conjunto de dados, cujo tamanho é proporcional ao número de usuários ou recursos presentes no sistema. Um aumento no tamanho resulta em alguma perda de desempenho.



2.4.3 Desafios da Escalabilidade: Impedir que os recursos de software se esgotem:

- Um exemplo de falta de escalabilidade é mostrado pelos números usados como endereços IP (endereços de computador na Internet).

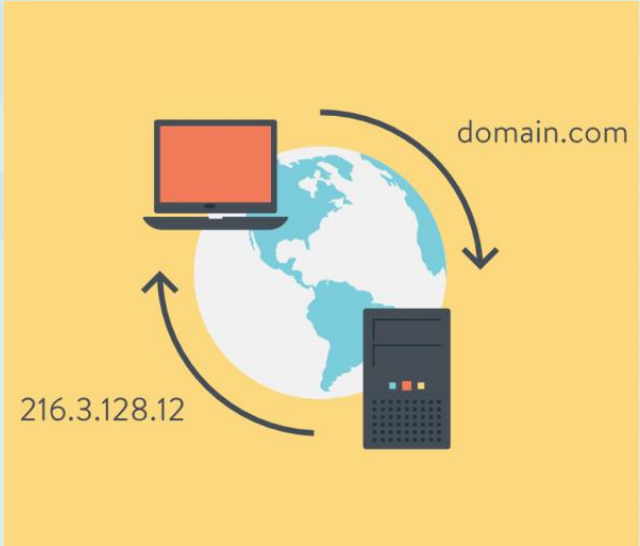
IPv4	IPv6
Implantado em 1981	Implantado em 1998
Endereço IP de 32-bit	Endereço IP de 128-bit
4,3 bilhões de endereços Endereços precisam ser reutilizados e mascarados	**340 undecilhões de endereços** Cada dispositivo tem um endereço exclusivo
Notação numérica decimal com ponto 192.168.5.18	Notação hexadecimal alfanumérica 50b2:6400:0000:0000:6c3a:b17d:0000:10a9 (Simplificado - 50b2:6400::6c3a:b17d:0:10a9)
DHCP ou configuração manual	Compatível com configuração automática

2.4.3 Desafios da Escalabilidade: Impedir que os recursos de software se esgotem

- **Superestimar o crescimento futuro pode ser pior do que se adaptar para uma mudança quando formos obrigados a isso.**
- Por exemplo, endereços IP maiores ocupam espaço extra no armazenamento de mensagens e no computador.

2.4.4 Desafios da Escalabilidade: Evitar gargalos de desempenho

- Em geral, os algoritmos devem ser descentralizados para evitar a existência de gargalos de desempenho.
- Exemplo: **DNS**



2.4.4 Desafios da Escalabilidade: Evitar gargalos de desempenho

- A tabela de correspondência entre endereços IP e nomes era mantida em um único arquivo central, cujo download podia ser feito em qualquer computador que precisasse dela.

2.4.4 Desafios da Escalabilidade: Evitar gargalos de desempenho

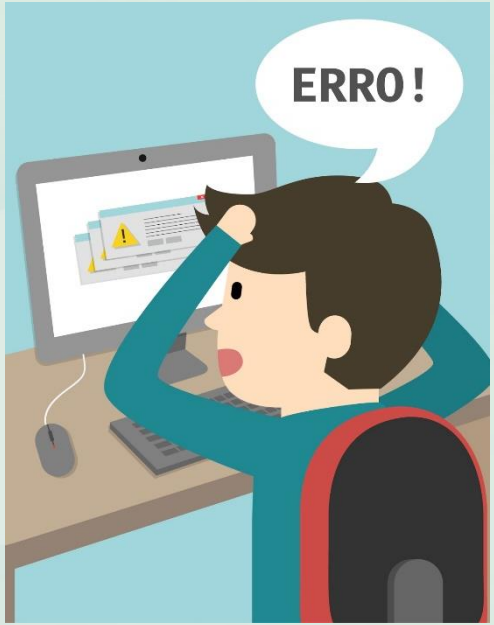
- O Domain Name System eliminou esse gargalo, particionando a tabela de correspondência de nomes entre diversos servidores localizados em toda a Internet e administrados de forma local.
- O uso da cache pode evitar gargalos de desempenho?

2.4.5 Desafios da Escalabilidade

- De preferência, o software de sistema e de aplicativo não deve mudar quando a escala do sistema aumentar, mas isso é difícil de conseguir. O problema da escala é um tema central no desenvolvimento de sistemas distribuídos.

2.5. Tratamento de falhas

- Às vezes, os sistemas de computador falham. Quando ocorrem falhas no hardware ou no software, os programas podem produzir resultados incorretos ou podem parar antes de terem concluído a computação pretendida.



2.5. Tratamento de falhas

- Quando um dos componentes de um sistema distribuído falha, **apenas o trabalho que estava usando o componente defeituoso é afetado.** Um usuário pode passar para outro computador, caso aquele que estava sendo utilizado falhe; um processo servidor pode ser iniciado em outro computador.

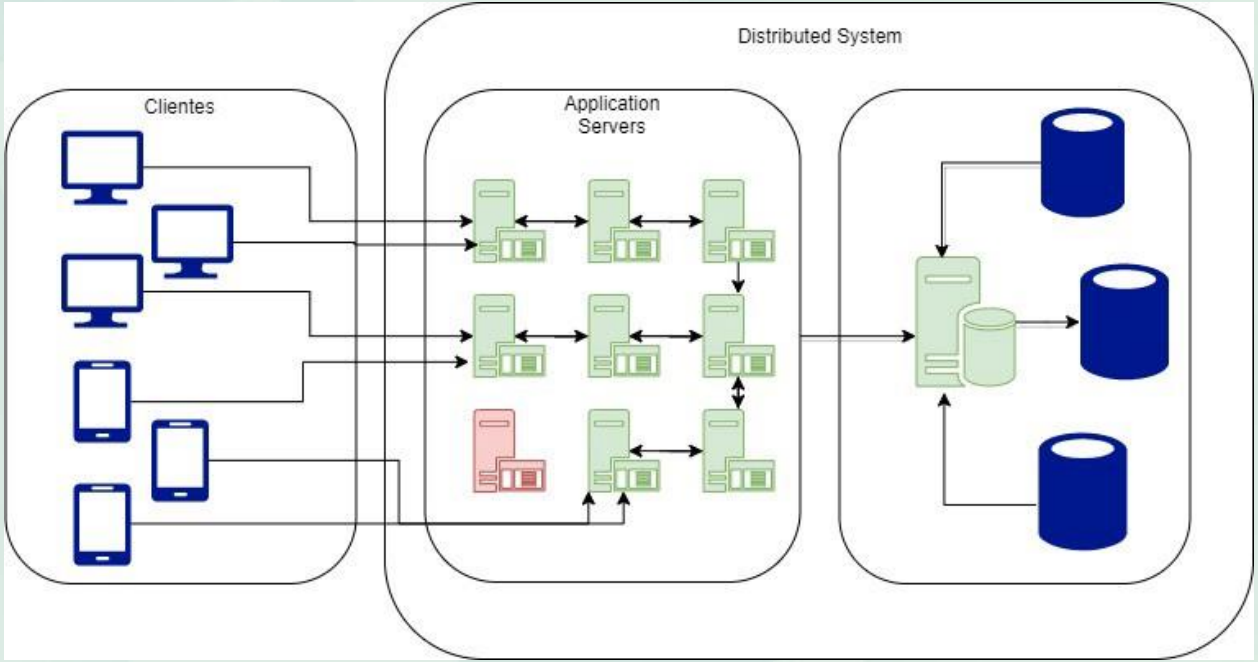
2.5. Tratamento de falhas

- As falhas em um sistema distribuído são parciais – isto é, alguns componentes falham, enquanto outros continuam funcionando. Portanto, o tratamento de falhas é particularmente difícil.
- As técnicas para tratamento de falhas serão discutidas a seguir.

2.5.1. Técnicas para tratamento de falhas

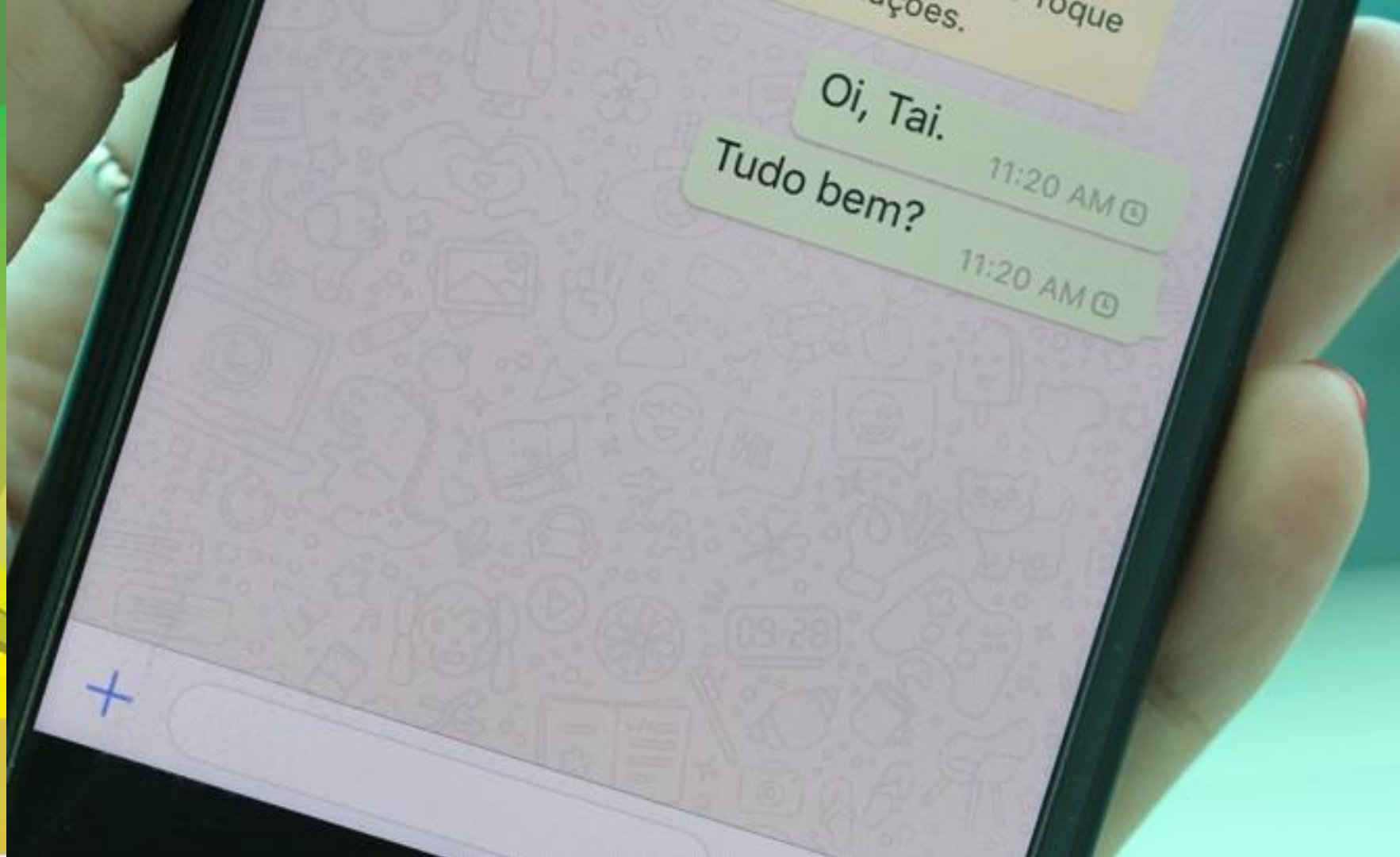
- **Detecção de falhas:** algumas falhas podem ser detectadas, porém o desafio é gerenciar a ocorrência de falhas que não podem ser detectadas, mas que podem ser suspeitas.
- Exemplo: um servidor remoto danificado na Internet.

2.5. Tratamento de falhas



2.5.1. Técnicas para tratamento de falhas

- **Mascaramento de falhas:** algumas falhas detectadas podem ser ocultas ou se tornar menos sérias. Dois exemplos de ocultação de falhas:
- 1. Mensagens podem ser retransmitidas quando não chegam.
- 2. Dados de arquivos podem ser gravados em dois discos, para que, se um estiver danificado, o outro ainda possa estar correto.


Oi, Tai.
11:20 AM
Tudo bem?
11:20 AM

2.5.1. Técnicas para tratamento de falhas

- **Tolerância a falhas:** a maioria dos serviços na Internet apresenta falhas – não seria prático para eles tentar detectar e mascarar tudo que possa ocorrer em uma rede grande assim, com tantos componentes.

2.5.1. Técnicas para tratamento de falhas

- **Tolerância a falhas (continuação):** Por exemplo, quando um navegador não consegue contatar um servidor Web, ele não faz o usuário esperar indefinidamente, enquanto continua tentando – ele informa o usuário sobre o problema, deixando-o livre para tentar novamente.


https://apps.correios.com.br/idC
apps.correios.com.br/idCorreios/inicio/inicio.jsf?nomeSistema=idCorreios&ticket=ST-9285-EN4LiOpXhhrnIAkiNW
Produção
Contraste
A · Texto no tamanho padrão
Correios
Sobre o IdCorreios
idCorreios
Ocorreu um erro!
Aviso!
Estamos com problemas para realizar o acesso ao sistema. Por favor, tente novamente em 10 minutos.
Código do erro: 500

2.5.1. Técnicas para tratamento de falhas

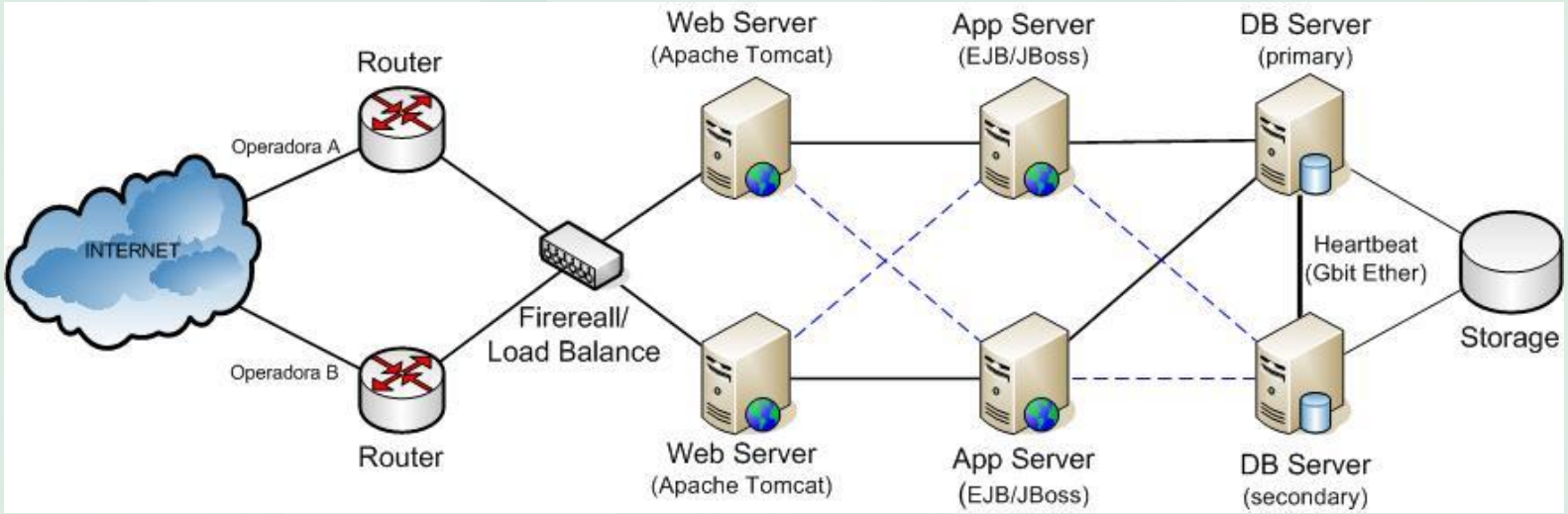
- **Recuperação de falhas:** a recuperação envolve projetar software de modo que o estado dos dados permanentes possa ser recuperado ou “retrocedido” após a falha de um servidor.

2.5.1. Técnicas para tratamento de falhas

- **Redundância:** os serviços podem se tornar tolerantes a falhas com o uso de componentes redundantes.
- Os servidores podem ser projetados de forma a detectar falhas em seus pares; quando uma falha é detectada em um servidor, os clientes são redirecionados para os servidores restantes.

2.5.1. Técnicas para tratamento de falhas

- **Redundância:**



2.6. Concorrência

- Tanto os serviços como os aplicativos fornecem recursos que podem ser compartilhados pelos clientes em um sistema distribuído. Portanto, existe a possibilidade de que vários clientes tentem acessar um recurso compartilhado ao mesmo tempo.

2.6. Concorrência

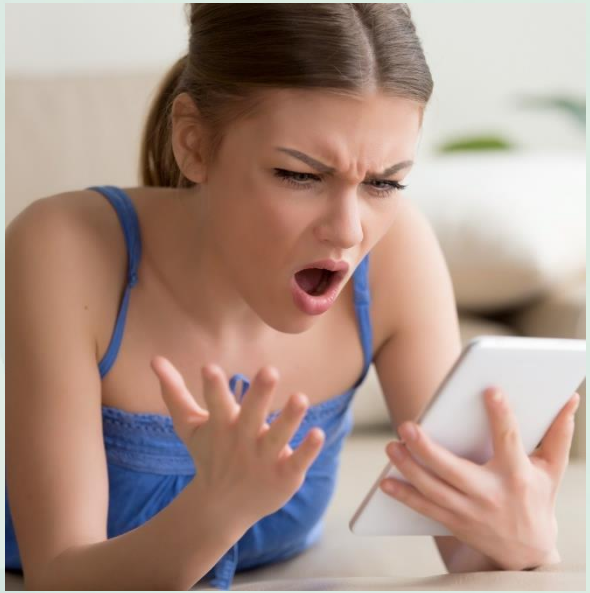
2.6. Concorrência

- O processo que gerencia um recurso compartilhado poderia aceitar e tratar um pedido de cliente por vez. Contudo, essa estratégia limita o desempenho do tratamento de pedidos. Portanto, os serviços e aplicativos geralmente permitem que vários pedidos de cliente sejam processados concorrentemente.

2.6. Concorrência

2.6. Concorrência

- Para que um objeto mantenha coerência em um ambiente concorrente, suas operações devem ser sincronizadas de tal maneira que seus dados permaneçam consistentes.



2.7. Transparência

- A transparência é definida como a **ocultação**, para um usuário final ou para um programador de aplicativos, da **separação dos componentes** em um sistema distribuído, de modo que o sistema **seja percebido como um todo**, em vez de como uma coleção de componentes independentes.

2.7.1. Formas de transparência

- **Transparência de acesso:** permite que recursos locais e remotos sejam acessados com o uso de operações idênticas.
- **Transparência de concorrência:** permite que vários processos operem concorrentemente, usando recursos compartilhados sem interferência entre eles.

2.7.1. Formas de transparência

- **Transparência de localização:** permite que os recursos sejam acessados sem conhecimento de sua localização física ou em rede (por exemplo, que prédio ou endereço IP).

2.7.1. Formas de transparência

- **Transparência de replicação:** permite que várias instâncias dos recursos sejam usadas para aumentar a confiabilidade e o desempenho, sem conhecimento das réplicas por parte dos usuários ou dos programadores de aplicativos.

2.7.1. Formas de transparência

- **Transparência de falhas:** permite a ocultação de falhas, possibilitando que usuários e programas aplicativos concluam suas tarefas, a despeito da falha de componentes de hardware ou software.

2.7.1. Formas de transparência

- **Transparência de mobilidade:** permite a movimentação de recursos e clientes dentro de um sistema, sem afetar a operação de usuários ou de programas.

2.7.1. Formas de transparência

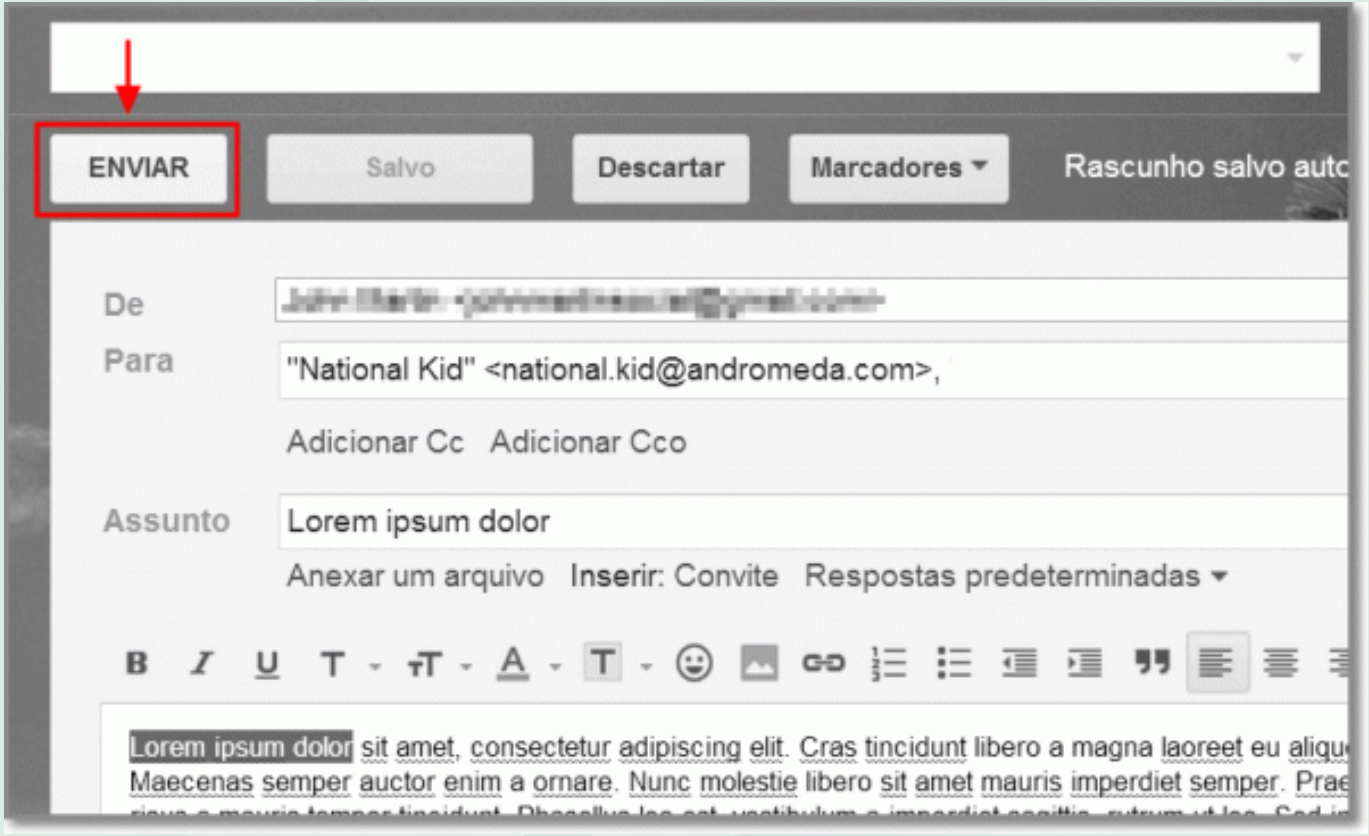
- **Transparência de desempenho:** permite que o sistema seja reconfigurado para melhorar o desempenho à medida que as cargas variam.
- **Transparência de escalabilidade:** permite que o sistema e os aplicativos se expandam em escala, sem alterar a estrutura do sistema ou os algoritmos de aplicação.

2.7.1. Formas de transparência

- **As duas transparências mais importantes são a de acesso e a de localização; sua presença ou ausência afeta mais fortemente a utilização de recursos distribuídos. Às vezes, elas são referidas em conjunto como transparência de rede.**

INSTITUTO FEDERAL
Sertão Pernambucano

Sist


ENVIAR
Salvo
Descartar
Marcadores
Rascunho salvo auto
De
Para
"National Kid" <national.kid@andromeda.com>,
Adicionar Cc Adicionar Cco
Assunto
Lorem ipsum dolor
Anexar um arquivo Inserir: Convite Respostas predeterminadas
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras tincidunt libero a magna laoreet eu
Maecenas semper auctor enim a ornare. Nunc molestie libero sit amet mauris imperdiet semper. Prae

2.7.1. Formas de transparência

- O uso dos diferentes tipos de transparência oculta e transforma em anônimos os recursos que não têm relevância direta para a execução de uma tarefa por parte de usuários e de programadores de aplicativos.

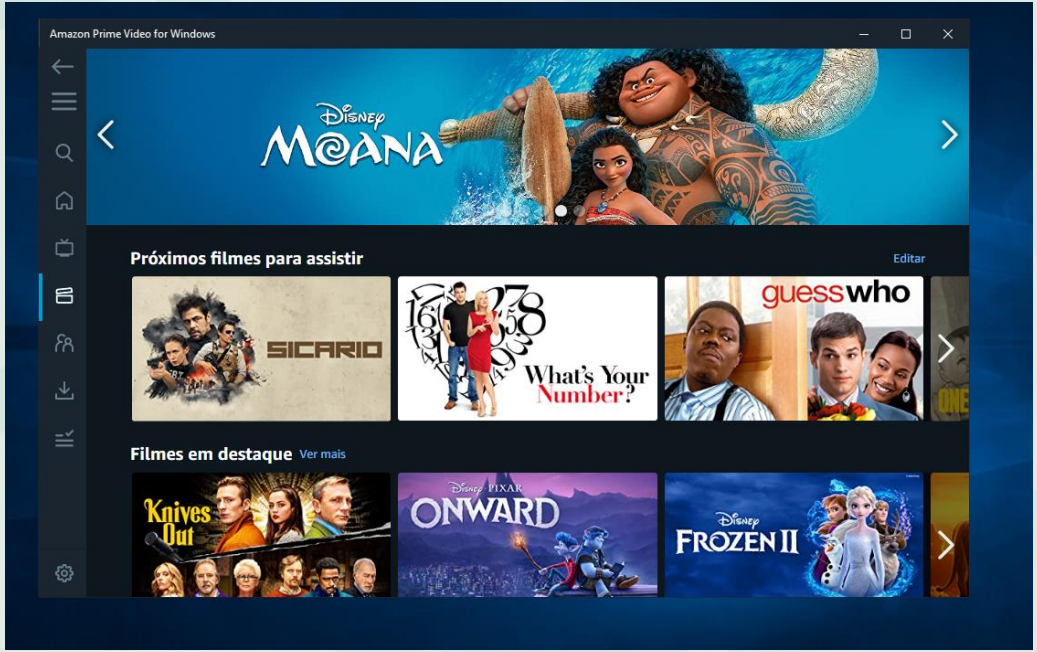
2.8. Qualidade de serviço

- As principais propriedades não funcionais dos sistemas que afetam a qualidade do serviço experimentada pelos clientes e usuários são a **confiabilidade**, a **segurança** e o **desempenho**. A adaptabilidade para satisfazer as configurações de sistema variáveis e a disponibilidade de recursos tem sido reconhecida como um aspecto muito importante da qualidade do serviço.

2.8. Qualidade de serviço

- Os problemas de confiabilidade e de segurança são fundamentais no projeto da maioria dos sistemas de computador. O aspecto do desempenho da qualidade de serviço foi definido originalmente em termos da **velocidade de resposta e do rendimento computacional**, mas foi redefinido em termos da **capacidade de satisfazer garantias de pontualidade**.

2.8. Qualidade de serviço



Obrigado!
Vlw! Flw!

INSTITUTO FEDERAL
Sertão Pernambucano