Unichristus
Centro Universitário Christus

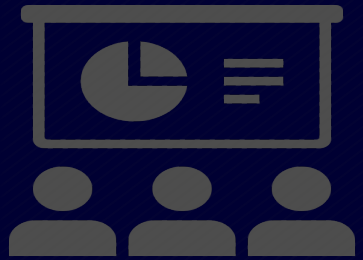
SISTEMAS DISTRIBUÍDOS

API de Protocolos

Lattes - linkedin
euristenhojr@gmail.com
http://www.unichristus.edu.br/

Euristenho Queiroz de Oliveira **Júnior**
Especialista em Engenharia de Software
MSc em Engenharia de Software

AGENDA



1. Apresentação
2. Livros
3. API
4. Multicast
5. Virtualização
6. Próxima Aula
7. Referências

FORMAÇÃO ACADÊMICA

- Graduado em Telemática/Telecomunicações - IFCE (2002 - 2008)
- Especialista em Engenharia de Software - FA7 (2011 - 2013)
- MSc em Engenharia de Software - UFPE (2011 - 2015)

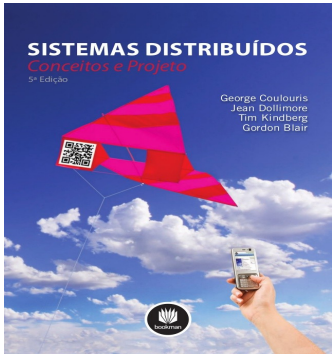
CURRÍCULO PROFISSIONAL

- Atuei 4 anos na empresa privada
- 9 anos no ambiente Público
- Atualmente Líder Técnico de 45 Projetos de Tecnologia na SEPOG/PMF

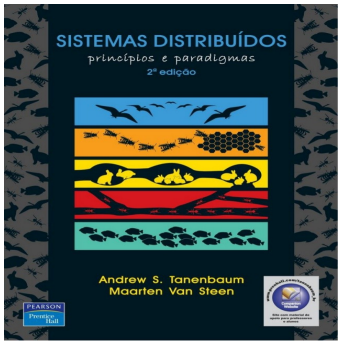
DOCÊNCIA

- Professor Substituto das Disciplinas de Sistemas de Informação – FA7 (2011 - 2012)
- Professor da Especialização em Sistemas WEB – FJN (2011 - 2012)
- Professor de Bancas de graduação em Sistemas de Informações – FA7 (2012)
- Professor dos Cursos de Tecnologia da Unifanor (2015 - ATUAL)
- Professor da Unichristus (2018 - ATUAL)

- **Sistemas Distribuídos - Conceitos e Projeto** - 5ª Ed. 2013 - George Coulouris, Tim Kindberg, Jean Dollimore



- **Sistemas Distribuídos, Princípios e Paradigmas** - 2ª Ed. 2007 - Andrew S. Tanembaum, Maarten Van Steen



OBJETIVOS

- Apresentar os conceitos básicos da computação distribuída e seus desafios como Heterogeneidade; Segurança; Tolerância a Falhas; Escalabilidade; Concorrência; Coordenação e Sincronização de processos; Comunicação interprocessos.
- Desenvolver competências e habilidades que auxiliem o profissional de Ciência da Computação a implementar os conceitos de sistemas distribuídos no desenvolvimento de sistemas de informação.
- Conhecer a aplicação desses conceitos em estudos de Casos que abordam arquiteturas e tecnologias modernas como RMI, CORBA e Web Services.

Agenda

1. **Lista de Exercícios**
2. Introdução
3. API para protocolos Internet
4. Representação externa de dados e empacotamento
5. Comunicação por Multicast (difusão seletiva)
6. Virtualização de redes: redes de sobreposição

Exercícios

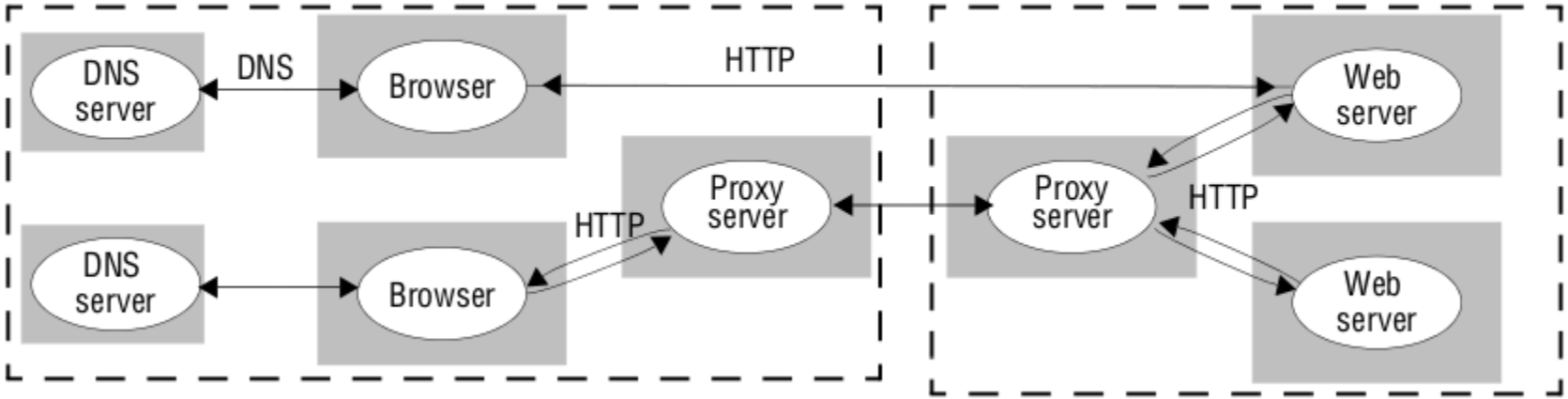
1. Dê três exemplos específicos e contrastantes dos níveis de heterogeneidade cada vez maiores experimentados nos sistemas distribuídos atuais, conforme definido na Seção 2.2
2. Descreva e ilustre a arquitetura cliente-servidor de um ou mais aplicativos de Internet importantes (por exemplo, Web, correio eletrônico ou news).

Exercícios

1. Dê três exemplos específicos e contrastantes dos níveis de heterogeneidade cada vez maiores experimentados nos sistemas distribuídos atuais, conforme definido na Seção 2.2
   - Em termos de hardware
   - Em termos de Sistemas operacionais
   - Em termos de redes

2. Descreva e ilustre a arquitetura cliente-servidor de um ou mais aplicativos de Internet importantes (por exemplo, Web, correio eletrônico ou news).

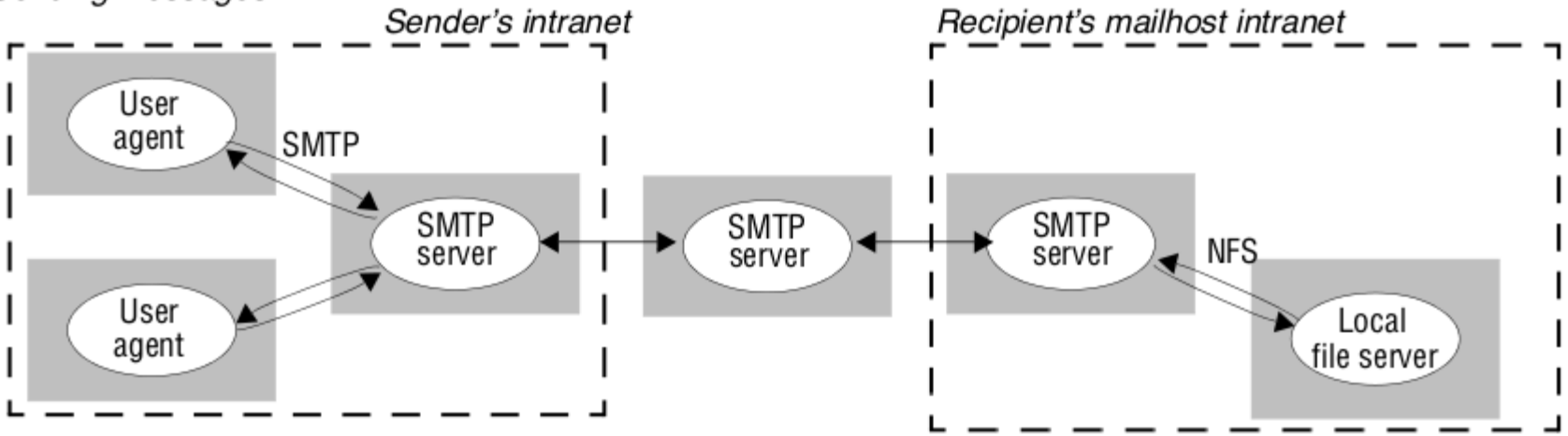
Web:



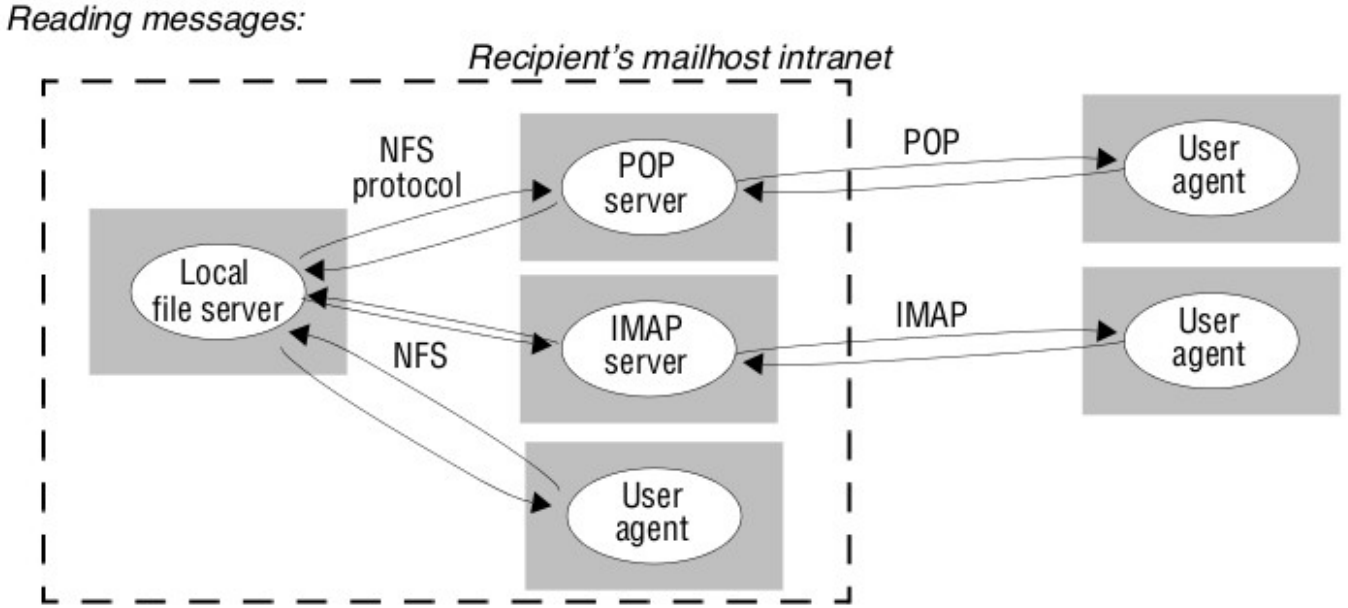
2. Descreva e ilustre a arquitetura cliente-servidor de um ou mais aplicativos de Internet importantes (por exemplo, Web, correio eletrônico ou news).

Email:

Sending messages:



2. Descreva e ilustre a arquitetura cliente-servidor de um ou mais aplicativos de Internet importantes (por exemplo, Web, correio eletrônico ou news).



Exercícios

1. Um mecanismo de busca é um servidor Web que responde aos pedidos do cliente para pesquisar em seus índices armazenados e (concomitantemente) executa várias tarefas de Web crawling para construir e atualizar esses índices. Quais são os requisitos de sincronização entre essas atividades concomitantes?
2. Frequentemente, os computadores usados nos sistemas peer-to-peer são computadores desktop dos escritórios ou das casas dos usuários. Quais são as implicações disso na disponibilidade e na segurança dos objetos de dados compartilhados que eles contêm e até que ponto qualquer vulnerabilidade pode ser superada por meio da replicação?

Agenda

1. Lista de Exercícios
2. **Introdução**
3. API para protocolos Internet
4. Representação externa de dados e empacotamento
5. Comunicação por Multicast (difusão seletiva)
6. Virtualização de redes: redes de sobreposição

Quais os dois **protocolos da camada de transporte** mais utilizados?

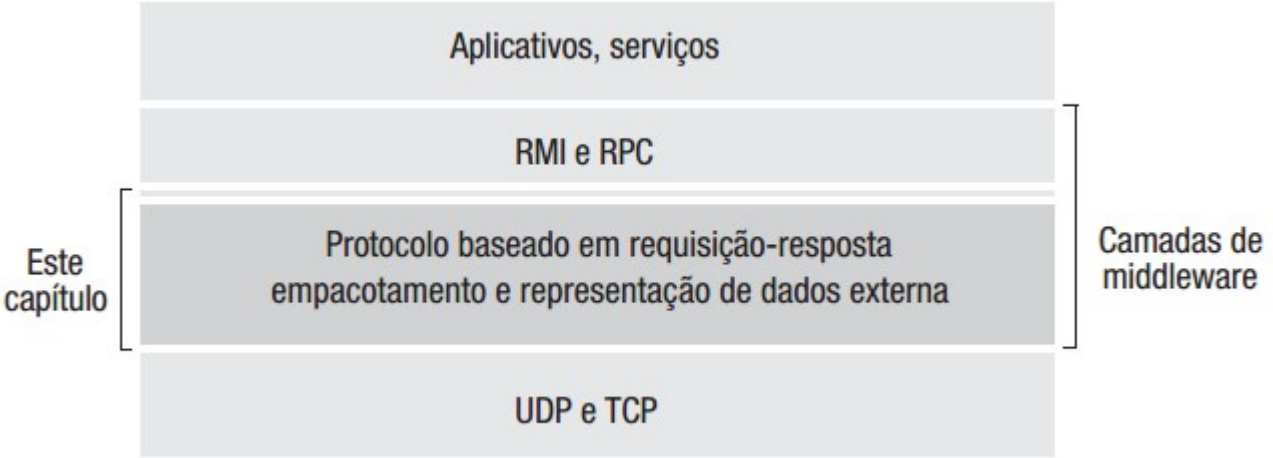
- **Protocolo UDP**:
  - Fornece uma abstração de passagem de mensagem
  - Pacotes independentes são chamados de datagramas
  - Nas APIs Java e UNIX, o remetente especifica o destino usando um **soquete (socket)**

- **Protocolo TCP**:
  - Fornece uma abstração de um fluxo (stream) bidirecional entre pares de processos
  - Comunicação **produtor-consumidor**

Introdução

Para um sistema distribuído, qual a desvantagem em se ter uma comunicação ponto-a-ponto?

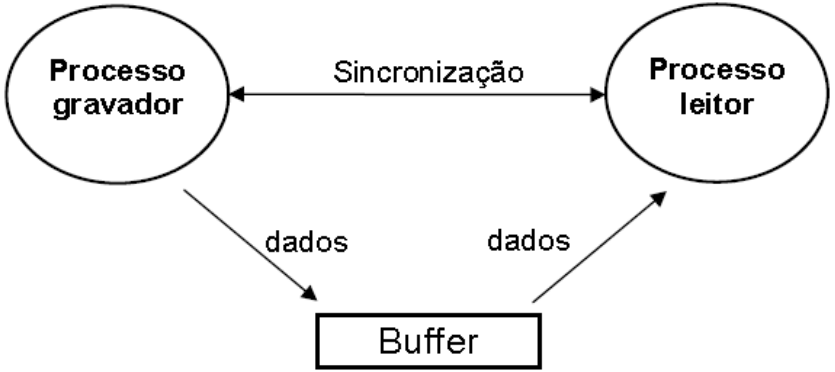
- **Assuntos a serem abordados:**



Agenda

1. Lista de Exercícios
2. Introdução
3. **API para protocolos Internet**
4. Representação externa de dados e empacotamento
5. Comunicação por Multicast (difusão seletiva)
6. Virtualização de redes: redes de sobreposição

- **Características da comunicação entre processos**:
  - **Passagem de mensagem** entre um par de processo pode ser suportada por duas operações: **send e receive**
  - Uma fila é **associada a cada destino de mensagem**.
    - Os processos origem fazem as mensagens serem adicionadas em filas remotas
    - Os processos destino removem mensagens de suas filas locais



- **Características da comunicação entre processos**:
  - Comunicação síncrona e assíncrona:
    - **Comunicação síncrona**:
      - Processos sincronizados a cada mensagem
      - As **operações send e receive são bloqueantes**
    - **Comunicação assíncrona**:
      - O uso da **operação send é não bloqueante**
      - O envio da mensagem pode ocorrer em paralelo a execução do processo
      - A recepção pode ser executada em background

- **Características da comunicação entre processos**:
  - Comunicação síncrona e assíncrona:
    - Em um **ambiente que suporta múltiplas threads**, a recepção bloqueante não tem desvantagens
    - A **comunicação não bloqueante** parece ser mais eficiente, mas ela envolve uma **complexidade extra no processo destino**. Por que?

- **Características da comunicação entre processos**:
  - Comunicação síncrona e assíncrona:
    - Em um **ambiente que suporta múltiplas threads**, a recepção bloqueante não tem desvantagens
    - A comunicação não bloqueante parece ser mais eficiente, mas ela envolve uma complexidade extra no processo destino. Por que?
      - Necessidade de ler uma mensagem recebida fora de seu fluxo normal de execução.

A API para protocolos Internet

Como o processo/aplicação de destino de uma mensagem é identificado?

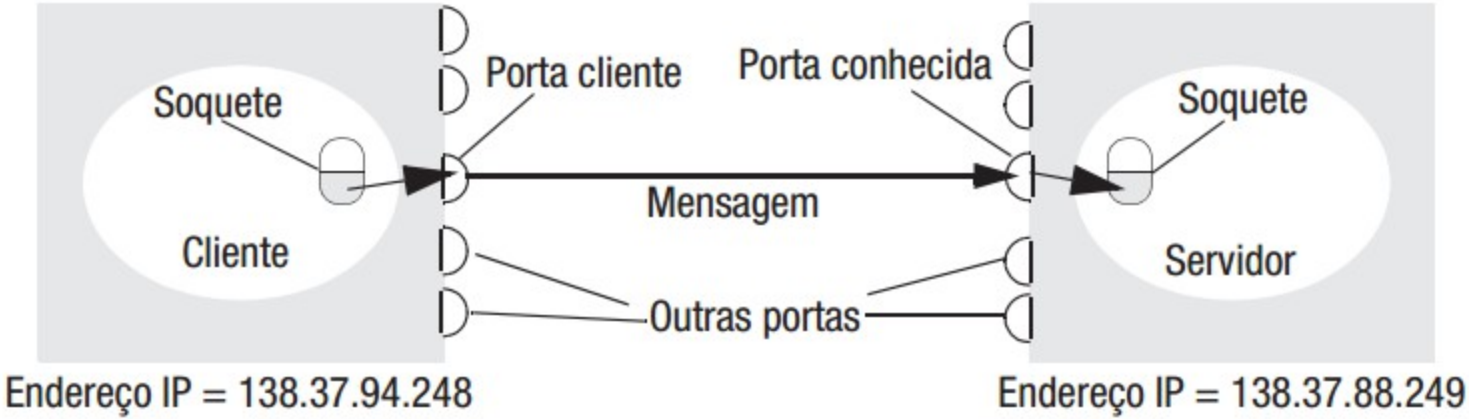
- **Características da comunicação entre processos**:
  - Destino de mensagem:
    - Destino identificado pelo par **(endereço IP, porta local)**
      - Uma porta tem somente um destino (exceto as portas multicast), mas pode ter múltiplos remetentes.
    - **Utilização de endereço IP para se referir a um serviço**: aplicação sempre na mesma máquina
    - Para proporcionar **transparência de localização**, isso pode ser evitado com o uso da seguinte estratégia:
      - Os programas clientes se referem aos serviços pelo nome e usam um **servidor de nomes ou de associação (binder)** para transformar seus nomes em localizações de servidor no momento da execução

Relembrando, quais as duas propriedades que deve haver para se ter **confiabilidade**?

- **Características da comunicação entre processos**:
  - **Confiabilidade**:
    - **Propriedade de validade**: um serviço de mensagem ponto a ponto pode ser descrito como confiável se **houver garantia de que as mensagens foram entregues**, independentemente da quantidade de pacotes que possam ter sido eliminados ou perdidos.
    - **Propriedade de integridade**: as mensagens devem chegar **não corrompidas e sem duplicação.**
  - **Ordenação:**
    - Algumas aplicações exigem que as mensagens sejam entregues na ordem de emissão
    - A entrega de mensagens fora da ordem da origem é considerada uma falha por tais aplicações.

O que vocês lembram sobre sockets?

- **Sockets**:
  - As **duas formas de comunicação (UDP e TCP)** usam a abstração de socket, um **ponto de destino** para a comunicação entre processos



Qual a diferença entre porta de rede e socket?

- **Sockets**:
  - Quantidade de portas: 216
  - Qualquer processo pode fazer uso de várias portas para receber mensagens, mas **um processo não pode compartilhar portas** com outros processos no mesmo computador
  - **Cada socket é associado a um protocolo em particular: TCP ou UDP**

Como funciona a comunicação UDP?

- **Comunicação por datagrama UDP**:
  - Um **datagrama enviado** pelo protocolo UDP é transmitido de um processo origem para um processo destino **sem a existência de confirmações ou novas tentativas de envio**
  - Antes de utilizar as **chamadas send e receive**, um processo precisa primeiro **criar uma associação entre um socket com um endereço IP e com uma porta do host local**
  - O método receive retorna, além da **mensagem, o endereço IP e a porta da origem** permitindo que o destinatário envie uma resposta a este

Quais os problemas relacionados à comunicação UDP?

- **Comunicação por datagrama UDP**:
  - Problemas relacionados:
    - **Tamanho da mensagem**: o processo destino precisa especificar um vetor de bytes de um tamanho em particular para receber as mensagens. Se a mensagem for grande demais para esse vetor, ela será truncada na chegada
    - **Bloqueio**: os sockets fornecem **operações send não bloqueantes e receive bloqueantes** para comunicação por datagrama
      - Ao chegar, a mensagem é posta em uma fila de recepção vinculada ao socket associado à porta de destino
      - O **método receive bloqueia a execução do processo até que um datagrama seja recebido**, a não ser que um tempo de espera limite tenha sido fornecido ao socket.

- **Comunicação por datagrama UDP**:
  - Problemas relacionados:
    - **Timeouts:**
      - A recepção bloqueante é **conveniente para uso por um servidor** que esteja esperando para receber requisições de seus clientes
      - Em algumas situações, não é adequado que **o processo espere indefinidamente para receber algo**, sendo definidos limites temporais para evitar isso.
      - Difícil de escolher um timeout apropriado.
    - **Recepção Anônima**:
      - O método receive **não especifica uma origem para as mensagens**.
      - A invocação ao método receive obtém uma mensagem endereçada para seu socket, independentemente da origem

Quais os falhas podem ocorrer quando o protocolo UDP é utilizado? Descreva o modelo de falhas.

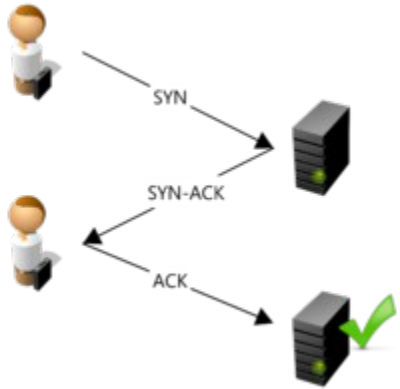
- **Comunicação por datagrama UDP**:
  - Relembrando…
    - A **propriedade da integridade** exige que as mensagens não devam estar corrompidas nem estejam duplicadas
    - A **propriedade da validade** exige que qualquer mensagem do buffer de envio é entregue ao buffer de recepção de seu destino, independentemente do tempo necessário para tal

- **Comunicação por datagrama UDP**:
  - Modelo de falhas:
    - No datagrama UDP, o **uso de soma de verificação** garante que haja uma probabilidade insignificante de que qualquer mensagem recebida esteja corrompida
    - Os datagramas sofrem das seguintes falhas:
      - Falhas por omissão
      - Ordenamento
    - É possível ter uma aplicação confiável utilizando o UDP?

Em quais situações os datagramas UDP geralmente são utilizados?

Quais as características do TCP?

- **Comunicação por datagrama TCP**:
  - A API do protocolo TCP fornece a **abstração de um fluxo de bytes** no qual dados podem ser lidos (receive) e escritos (send).
  - As características a seguir são ocultas pela abstração de fluxo:
    - Tamanho das mensagens
    - Mensagens perdidas
    - Controle de fluxo
    - Duplicação e ordenamento das mensagens
    - Destinos de mensagem (estabelecimento de conexão)



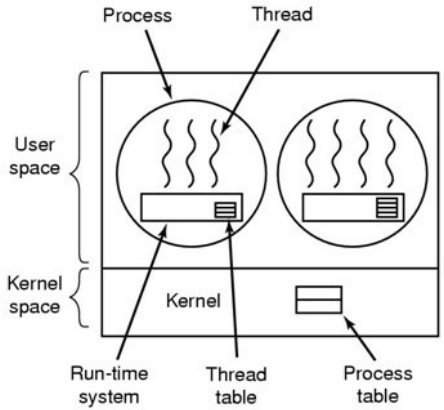
- **Comunicação por datagrama TCP**:
  - Papéis na comunicação: **cliente e servidor**
  - O **socket de “escuta”** mantém uma fila de **pedidos de conexão** recebidos.
    - Quando o servidor aceita uma conexão, **um novo socket de fluxo** é criado para que o servidor se comunique com um cliente
    - O **socket de “escuta”** na porta de serviço é mantido paralelamente para receber os pedidos connect de outros clientes.

Quais problemas podem ocorrer na comunicação por fluxo?

- **Comunicação por datagrama TCP**:
  - Problemas na comunicação por fluxo:
    - **Correspondência de itens de dados**:
      - dois processos que estejam se comunicando **precisam concordar quanto ao conteúdo dos dados transmitidos por um fluxo**.
      - Ex: se um processo escreve (envia) um valor int em um fluxo, seguido de um valor double, então o outro lado deverá ler um valor int, seguido de um valor double
      - Caso essa sincronia não aconteça, poderá ocorrer **erros na interpretação dos dados**

- **Comunicação por datagrama TCP**:
  - Problemas na comunicação por fluxo:
    - Bloqueio:
      - os dados gravados em um fluxo são mantidos em uma **fila no soquete de destino**.
      - Quando um processo tentar ler dados de um canal de entrada, **obterá dados da fila ou será bloqueado até que dados se tornem disponíveis**.
      - O **processo que escreve dados em um fluxo pode ser bloqueado** pelo mecanismo de controle de fluxo TCP, caso o soquete no outro lado já esteja armazenando o volume máximo de dados permitido pelo protocolo. (que característica é essa?)

- **Comunicação por datagrama TCP**:
  - Problemas na comunicação por fluxo:
    - **Threads**:
      - quando um servidor aceita uma conexão, ele geralmente cria uma nova thread para se comunicar com o novo cliente.
      - A vantagem de usar uma thread separada para cada cliente **é que o servidor pode bloquear quando estiver esperando por dados**, sem atrasar os outros clientes.



Quais falhas podem ocorrer quando é utilizado o TCP?

- **Comunicação por datagrama TCP**:
  - **Modelo de Falhas**:
    - A **propriedade da integridade** da **comunicação confiável**, os fluxos TCP **usam somas de verificação** e, para pacotes duplicados, são utilizados **os números de sequência**.
    - Com relação a **propriedade da validade**, os fluxos TCP usam timeout e retransmissões para lidar com pacotes perdidos.
      - Portanto, há garantia de que as mensagens sejam entregues, mesmo quando alguns dos pacotes das camadas inferiores são perdidos
      - Caso a quantidade de pacotes perdidos ultrapassar um limite, o software TCP responsável pelo envio de mensagens não receberá nenhum tipo de confirmação e, após certo tempo, declarará que a conexão está desfeita

- **Comunicação por datagrama TCP**:
  - **Modelo de Falhas**:
    - Quando **uma conexão é desfeita**, um processo, ao tentar ler ou escrever algo nela, receberá uma notificação de erro. Isso tem os seguintes efeitos:
      - os processos que estão usando a conexão **não poderão distinguir entre falha de rede e falha do processo** no outro lado da conexão;
      - os processos que estão se comunicando não poderão identificar se as mensagens que enviaram recentemente foram recebidas ou não.

Agenda

1. Lista de Exercícios
2. Introdução
3. API para protocolos Internet
4. **Representação externa de dados e empacotamento**
5. Comunicação por Multicast (difusão seletiva)
6. Virtualização de redes: redes de sobreposição

Representação externa de dados e empacotamento

Como e onde os dados de uma aplicação são armazenados?

- As informações armazenadas nos programas em execução **são representadas como estruturas de dados**
- Independente da forma de comunicação, **as estruturas de dados devem ser simplificadas** (convertidas em sequência de bytes) **antes da transmissão e reconstruídas na sua chegada**.
- Os dados transmitidos nas mensagens podem corresponder a valores de **tipos de dados primitivos diferentes**, e nem todos os computadores armazenam tipos de dados primitivos, como os inteiros, na mesma ordem.
- Outro problema é o **conjunto de códigos** usado para representar caracteres

Quais seriam as possíveis soluções para esse problema?

- Possíveis métodos:
  - **Os valores são convertidos para um formato externo**, acordado antes da transmissão e convertidos para a forma local, na recepção;
    - se for sabido que os dois computadores são do mesmo tipo, a conversão para o formato externo pode ser omitida.
  - **Os valores são transmitidos no formato do remetente**, junto a uma indicação do formato usado, e o destinatário converte os valores, se necessário.

- Dado essa abordagem, dois procedimentos devem ser realizados:
  - **Empacotamento (marshalling)** é o procedimento de pegar um conjunto de itens de dados e montá-los em uma forma conveniente para transmissão em uma mensagem.
  - **Desempacotamento (unmarshalling)** é o procedimento inverso de desmontá-los na chegada para produzir um conjunto de itens de dados equivalente no destino

Quais seriam exemplos de estratégias empregadas atualmente?

- Estratégias para a representação externa de dados e empacotamento:
  - A **representação comum de dados do CORBA**, que está relacionada a uma **representação externa dos tipos estruturados e primitivos** que podem ser passados como argumentos e resultados na invocação a métodos remotos no CORBA.
  - A **serialização de objetos da linguagem Java**, que está relacionada à **simplificação e à representação externa de dados de um objeto**, **ou de um conjunto de objetos**, que precise ser transmitida em uma mensagem ou armazenada em um disco.
  - A **XML ou Extensible Markup Language**, que define um formato textual para representar dados estruturados.

- Nos dois primeiros casos, as atividades de empacotamento e desempacotamento **se destinam a serem executadas por uma camada de middleware**, sem nenhum envolvimento por parte do programador de aplicativo
  - Além disso, os dados primitivos são empacotados em uma forma binária
- Na terceira estratégia (XML), os tipos de dados primitivos são representados textualmente. **A representação textual de um valor de dados geralmente será maior do que a representação binária equivalente**

Qual outra forma está sendo utilizada atualmente?

- Nos dois primeiros casos, as atividades de empacotamento e desempacotamento se destinam a serem executadas por uma camada de middleware, sem nenhum envolvimento por parte do programador de aplicativo
  - Além disso, os dados primitivos são empacotados em uma forma binária
- Na terceira estratégia (XML), os tipos de dados primitivos são representados textualmente. **A representação textual de um valor de dados geralmente será maior do que a representação binária equivalente**

Quais as características da XML? Onde é utilizada?

- **XML (Extensible Markup Language)**
  - A **XML é uma linguagem de marcação** que foi definida pelo World Wide Web Consortium (W3C) para uso na Web
    - **Objetivo de uso**: elaboração de documentos estruturados para a Web
  - Os itens de dados XML são rotulados com **strings de marcação (tags)**.
    - As tags são usadas para **descrever a estrutura lógica dos dados** e para associar pares atributo-valor às estruturas lógicas

Qual a motivação para você utilizar XML na troca de mensagens?

- **XML (Extensible Markup Language)**
  - A XML é usada para permitir que **clientes se comuniquem com serviços Web** e para definir as interfaces e outras propriedades desses mesmos serviços
  - A XML é **extensível**, pois os usuários podem **definir suas próprias tags**
  - Diferentemente do CORBA, **em que algumas representações externas de dados (como o CDR do CORBA) não precisam ser autodescritivas**, a XML foi projetada para ser usadas por vários aplicativos para diferentes propósitos.

- **XML (Extensible Markup Language)**
  - Exemplo:

```
<person id="123456789">
    <name>Smith</name>
    <place>London</place>
    <year>1984</year>
    <!-- a comment -->
</person >
```

- **XML (Extensible Markup Language)**
  - Na prática, a maioria dos documentos XML é **gerada e lida por software de processamento de XML**, mas a capacidade de ler código XML pode ser útil quando as coisas dão errado. Além disso, **ele independe de plataforma**.
  - Qual o impacto de se usar uma representação textual ao invés da binária na transmissão de mensagens?

- **XML (Extensible Markup Language)**
  - Na prática, a maioria dos documentos XML é **gerada e lida por software de processamento de XML**, mas a capacidade de ler código XML pode ser útil quando as coisas dão errado. Além disso, **ele independe de plataforma**.
  - Qual o impacto de se usar uma representação textual ao invés da binária na transmissão de mensagens?
    - O uso de uma representação textual, em vez de binária, junto com o uso de tags, torna **as mensagens muito maiores, o que faz com que elas exijam tempos de processamento e transmissão maiores, assim como mais espaço de armazenamento**

- **XML (Extensible Markup Language)**
  - Um elemento na XML **consiste em um conjunto de dados do tipo caractere delimitados por tags de início e de fim correspondentes**
    - A capacidade de um elemento de incluir outro permite a **representação de dados hierárquicos** – um aspecto muito importante da XML. Uma tag vazia não tem conteúdo e é terminada com />, em vez de >

```
<person id="123456789">
    <name>Smith</name>
    <place>London</place>
    <year>1984</year>
    <!-- a comment -->
</person >
```

- **XML (Extensible Markup Language)**
  - Todas as informações nos elementos XML devem ser expressas com dados do tipo caractere, mas a questão é: **como representamos elementos criptografados ou hashing de códigos de segurança**?

- **XML (Extensible Markup Language)**
  - Todas as informações nos elementos XML devem ser expressas com dados do tipo caractere, mas a questão é: **como representamos elementos criptografados ou hashing de códigos de segurança**?
    - Podem ser representados na notação base64, que utiliza apenas os caracteres alfanuméricos, junto a +, / e =, que têm significado especial.

- **XML (Extensible Markup Language)**
  - Análise (parsing) e documentos bem formados
    - Um **documento XML deve ser bem formado** – isto é, ele deve obedecer às regras sobre sua estrutura.
      - Regras básicas:
        - Toda tag de início tem uma tag de fim correspondente.
        - Todas as tags devem ser corretamente aninhadas, por exemplo <x>..<y>..</y>..</x>está correto, enquanto <x>..<y>....</x>.. </y>, não.
        - Todo documento XML deve ter um único elemento-raiz que englobe todos os outros elementos

Representação externa de dados e empacotamento

Como funciona a referência a objetos remotos?

- **Referências a Objetos Remotos**
  - Quando um **cliente invoca um método em um objeto remoto, uma mensagem de invocação é enviada** para o processo servidor que contém o objeto remoto.
  - Essa mensagem precisa **especificar qual objeto em particular deve ter seu método executado**.
    - Uma referência de objeto remoto é o identificador de um objeto remoto, válido em todo um sistema distribuído.
  - A referência de objeto remoto é passada na mensagem de invocação para especificar qual objeto deve ser ativado

- **Referências a Objetos Remotos**
  - As **referências de objeto remoto** devem ser geradas de uma forma que **garanta sua exclusividade no espaço e no tempo**.
    - Em geral, podem existir muitos processos contendo objetos remotos; portanto, **as referências de objeto remoto devem ser únicas entre todos os processos, nos vários computadores de um sistema distribuído**.
  - Mesmo após um objeto remoto, associado a uma determinada referência, ter sido excluído, **é importante que a referência de objeto remoto não seja reutilizada**, pois seus invocadores em potencial podem manter referências obsoletas

Então, como poderíamos identificar um objeto de forma exclusiva em um sistema distribuído?

- **Referências a Objetos Remotos**
  - Uma das formas é construir uma referência concatenando o **endereço IP de seu computador e o número de porta do processo que a criou, com a hora de sua criação e um número de objeto local**
    - Juntos, o número de porta e a hora produzem um identificador de processo exclusivo nesse computador

32 bits	*32 bits*	*32 bits*	*32 bits*	
Endereço IP	Número de porta	Hora	Número do objeto	Interface do objeto remoto

O que você sabe sobre o CORBA?

- **CDR - CORBA:**
  - CORBA (abreviado de Common Object Request Broker Architecture) é a arquitetura padrão criada pelo Object Management Group para **estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos**.
  - Para a representação de dados (primitivos e estruturados) na invocação a objetos distribuídos foi definida a CDR (Common Data Representation)
    - permite clientes e servidores escritos em diferentes linguagens se comunicarem

- **CDR - CORBA:**
  - Tipos compostos

Tipo	*Representação*
sequence	comprimento (*unsigned long*) seguido de seus elementos, em ordem
string	comprimento (*unsigned long*) seguido pelos caracteres que o compõem (um caractere pode ocupar mais de um *byte*)
array	elementos de vetor, fornecidos em ordem (nenhum comprimento especificado, pois é fixo)
struct	na ordem da declaração dos componentes
enumerated	*unsigned long* (os valores são especificados pela ordem declarada)
union	identificador de tipo seguido do membro selecionado

- **CDR - CORBA:**
  - Operações realizadas durante o envio:
    - **Tipos primitivos**: o CDR define uma representação para as ordens **big-endian e little-endian**. Os valores são transmitidos na ordem do remetente, que é especificada em cada mensagem. Se exigir uma ordem diferente, o destinatário a transforma.
    - **Tipos construídos ou compostos**: os valores primitivos que compreendem cada tipo construído são adicionados a uma sequência de bytes, em uma ordem específica

- **CDR - CORBA:**
  - **Exemplo**

Índice na sequência de bytes	← 4 bytes →	*Observações sobre a representação*
0–3	5	*Comprimento do string*
4–7	"Smit"	*'Smith'*
8–11	"h___"	
12–15	6	*Comprimento do string*
16–19	"Lond"	*'London'*
20–23	"on__"	
24–27	1984	*unsigned long*

Agenda

1. Lista de Exercícios
2. Introdução
3. API para protocolos Internet
4. Representação externa de dados e empacotamento
5. **Comunicação por Multicast (difusão seletiva)**
6. Virtualização de redes: redes de sobreposição

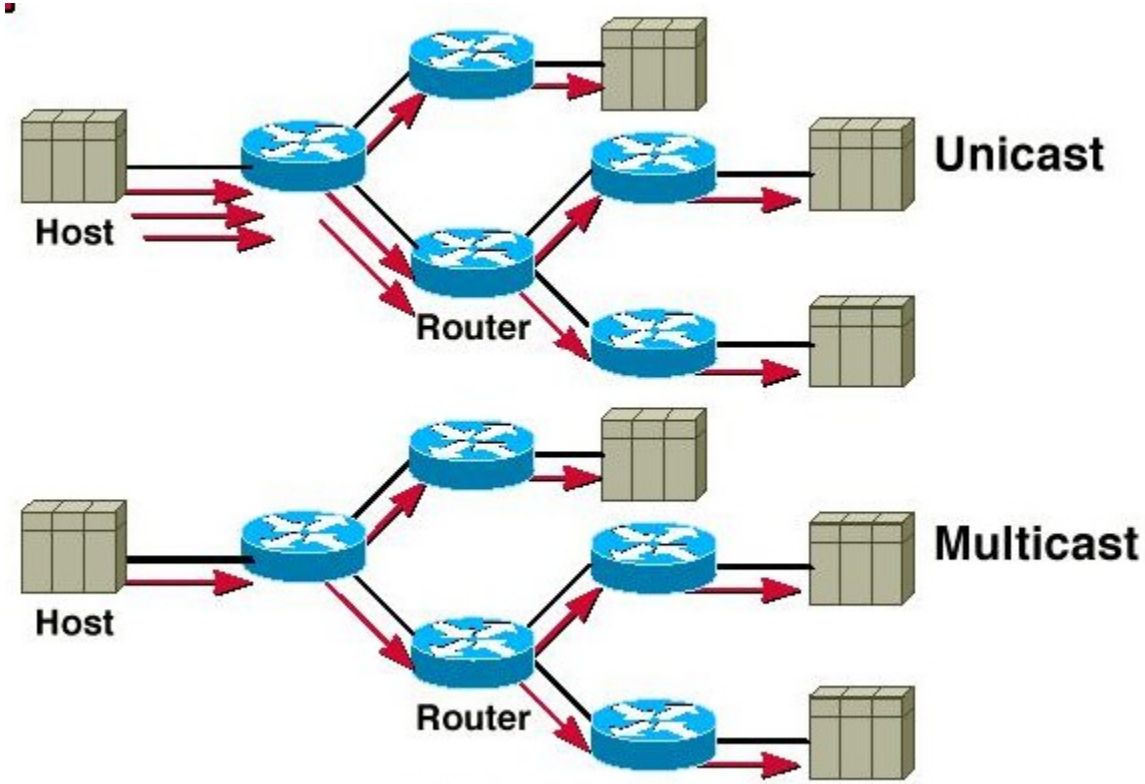
Comunicação por multicast

O que seria multicast?

Comunicação por multicast

- A troca de mensagens aos pares não é o melhor modelo para a comunicação de um processo com um grupo de outros processos
- O emprego de multicast é mais apropriado:
  - Permite o envio de **uma única mensagem** para cada um dos membros de um **grupo de processos** de tal forma que membros participantes do grupo ficam **totalmente transparentes** para o remetente

Comunicação por multicast



- As mensagens multicast fornecem uma infraestrutura útil para a construção de sistemas distribuídos com as seguintes características:
  - **Tolerância à falha baseada em serviços replicados**:
    - As requisições do cliente são difundidas para todos os membros do grupo, cada um dos quais executando uma operação idêntica.
  - **Localização de servidores de descoberta na interligação em rede espontânea**:
    - Mensagens multicast podem ser usadas por servidores e clientes para localizar os **serviços de descoberta disponíveis**, para registrar suas interfaces ou para pesquisar as interfaces de outros serviços no sistema distribuído.

- As mensagens multicast fornecem uma infraestrutura útil para a construção de sistemas distribuídos com as seguintes características:
  - **Melhor desempenho através da replicação de dados:**
    - Sempre que os dados mudam, o novo valor é enviado por multicast para os processos que gerenciam as réplicas.
  - **Propagação de notificações de evento:**
    - o multicast para um grupo pode ser usado para notificar os processos de quando algo acontece

Comunicação por multicast

O que você sabe sobre o IP de Multicast?

- **Multicast IP**:
  - O multicast IP permite que o remetente transmita um **único datagrama IP para um conjunto de computadores** que formam um grupo de multicast.
  - O remetente não conhece as identidades dos destinatários individuais nem o tamanho do grupo.
  - Um grupo multicast é especificado por um **endereço IP classe D**
  - Para a programação de aplicativos, o multicast IP está disponível apenas por meio de UDP.

- **Multicast IP**:
  - Classes IP

Classes IPv4 e Máscara de Rede

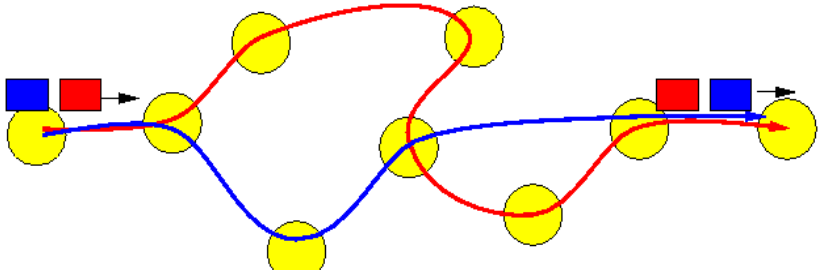
Classe	Início	Fim	Máscara de Rede Padrão	Notação CIDR	Nº de Endereços por Rede
A	1.0.0.0	126.255.255.255	255.0.0.0	/8	16 777 216
	127.0.0.0	127.255.255.255	255.0.0.0	/8	Localhost
B	128.0.0.0	191.255.255.255	255.255.0.0	/16	65 536
C	192.0.0.0	223.255.255.225	255.255.255.0	/24	256
D	224.0.0.0	239.255.255.255			Multicast
E	240.0.0.0	255.255.255.255			Uso futuro; atualmente reservada a testes pela IETF

- **Multicast IP**:
  - O fato de ser membro de um grupo multicast permite a um computador receber datagramas IP enviados para o grupo.
  - A **participação como membro de grupos multicast é dinâmica**, permitindo aos computadores entrarem ou saírem a qualquer momento e participarem de um número arbitrário de grupos

- **Multicast IP**:
  - Modelo de Falhas:
    - No IP, o envio de datagramas multicast tem as mesmas **características de falhas dos datagramas UDP** – isto é, **sofre de falhas por omissão**.
    - **Como não há garantias** de que uma mensagem será entregue para um membro de um grupo, esse tipo de comunicação é denominado de **multicast não confiável**

- **Confiabilidade e Ordenação**
  - O modelo de falhas do multicast IP, isto é, ele sofre de **falhas por omissão**
  - Outro fator que afeta o multicast IP é que qualquer **processo pode falhar**:
    - Se um roteador multicast falhar, os membros do grupo que estiverem além desse roteador não receberão a mensagem, embora os membros locais possam receber

- **Confiabilidade e Ordenação**
  - A ordenação das mensagens é outros problema:
    - Os datagramas IP enviados por várias redes interligadas não chegam necessariamente na ordem em que foram emitidos, com o possível efeito de que alguns membros do grupo recebam os datagramas de um único remetente em uma ordem diferente dos outros membros.
    - Além disso, as mensagens enviadas por dois processos diferentes não chegarão necessariamente na mesma ordem em todos os membros do grupo.



- **Confiabilidade e Ordenação**
  - Alguns exemplos dos efeitos da confiabilidade e da ordenação:
    - **Tolerância a falhas baseada em serviços replicados**:
      - considere um serviço replicado que consiste nos membros de um grupo de servidores
      - Essa aplicação multicast **impõe que todas as réplicas, ou nenhuma delas, devam receber cada pedido para executar uma operação** – se uma perder um pedido, ela se tornará inconsistente com relação às outras

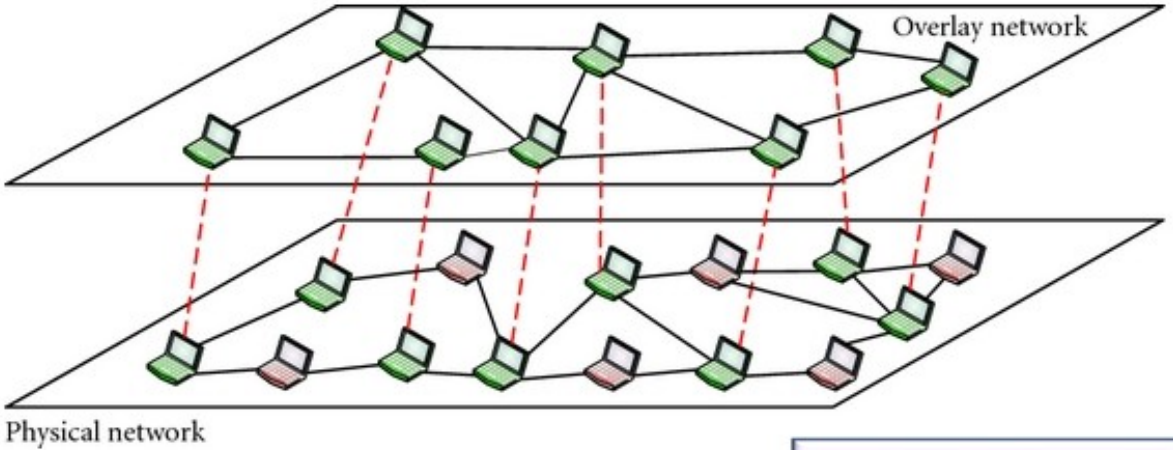
Agenda

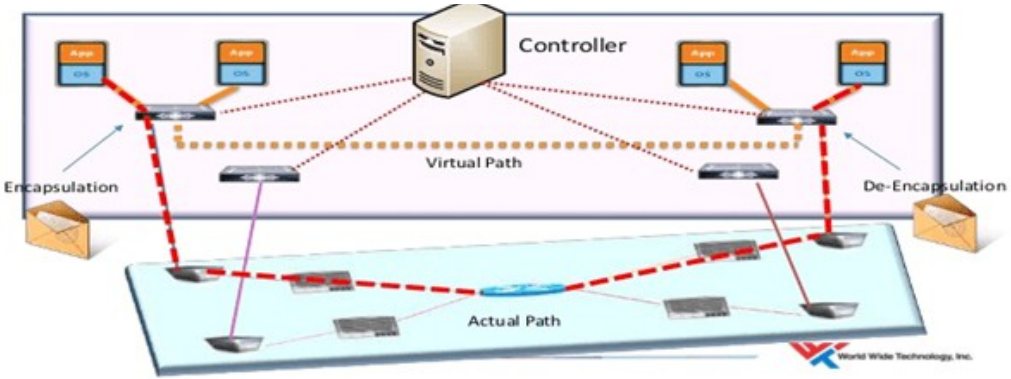
1. Lista de Exercícios
2. Introdução
3. API para protocolos Internet
4. Representação externa de dados e empacotamento
5. Comunicação por Multicast (difusão seletiva)
6. **Virtualização de redes: redes de sobreposição**

O que seria uma rede de sobreposição?

- Uma **rede de sobreposição (overlay)** é uma rede virtual consistindo em nós e enlaces virtuais, a qual **fica sobre uma rede subjacente (como uma rede IP) e oferece algo que de outro modo não é fornecido**:
  - um serviço personalizado
  - funcionamento mais eficiente em determinado ambiente interligado em rede
  - um recurso adicional; por exemplo, comunicação por multicast ou segura.

Virtualização de redes





- **Vantagens das redes de sobreposição:**
  - Permitem a definição de novos serviços de rede **sem exigir mudanças na rede subjacente**
  - Elas estimulam a **experimentação com serviços de rede** e a personalização de serviços para tipos de aplicativo específicos.
  - Várias sobreposições podem ser definidas e coexistir, sendo o resultado final uma arquitetura de rede mais aberta e extensível.

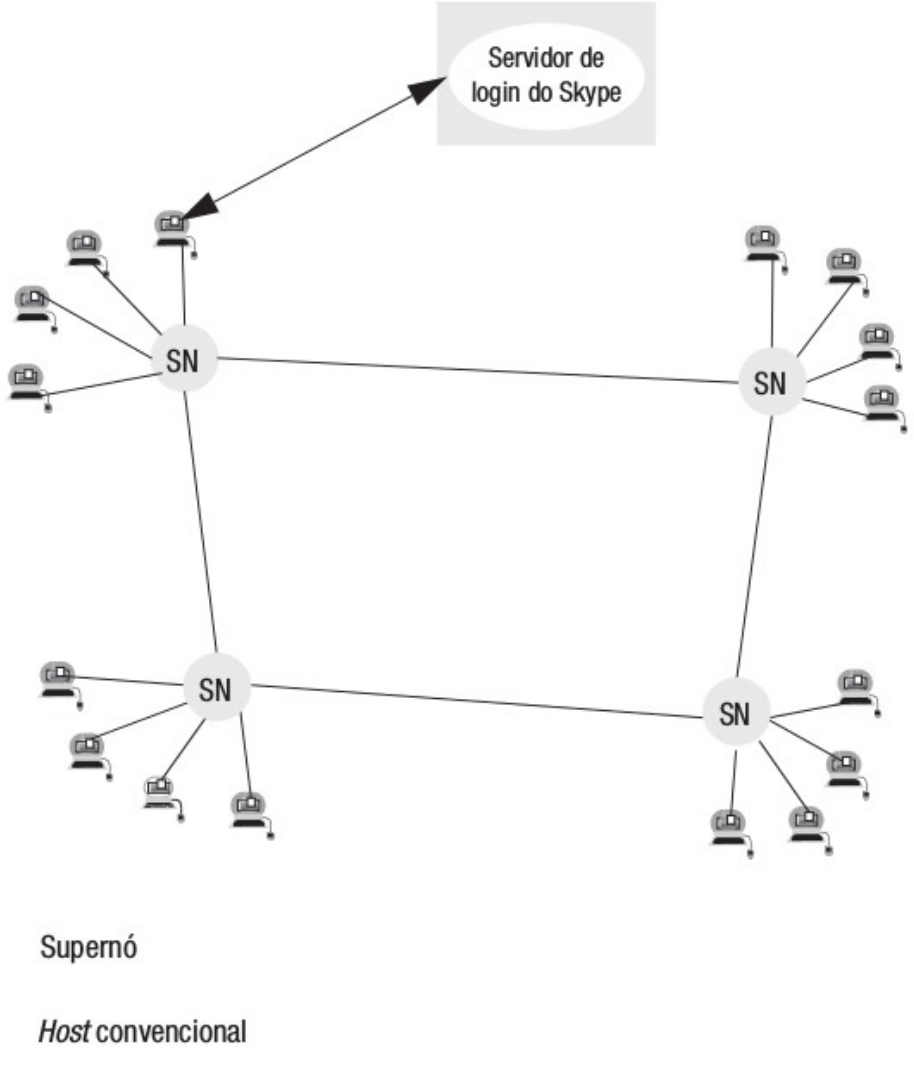
- **Desvantagens das redes de sobreposição**:
  - as redes de sobreposição **introduzem um nível extra de indireção** (e portanto podem acarretar queda no desempenho)
  - aumentam a **complexidade dos serviços de rede**, quando comparadas, por exemplo, com a arquitetura relativamente simples das redes TCP/IP.

Qual seria um exemplo de rede de sobreposição?

- **Skype: um exemplo de rede de sobreposição**
  - O Skype é uma rede virtual no sentido de que **estabelece conexões entre pessoas** (assinantes do Skype correntemente ativos).
  - Nenhum endereço IP ou porta é necessária para estabelecer uma chamada.
  - A arquitetura da rede virtual que dá suporte ao Skype não é amplamente publicada
    - É baseado em uma **infraestrutura peer-to-peer** que consiste em máquinas normais de usuário (referidas como **hosts**) e **supernós** – os quais são hosts normais do Skype que têm recursos suficientes para cumprir sua função avançada.
    - Os usuários do Skype são autenticados por meio de um **servidor de login conhecido**.

- **Skype: um exemplo de rede de sobreposição**
  - Operações:
    - **Busca de usuários**: O principal objetivo dos supernós é fazer a pesquisa eficiente de índices globais de usuários, os quais são distribuídos pelos supernós.
    - **Conexão de voz:** Uma vez encontrado o usuário desejado, o Skype estabelece uma conexão de voz entre as duas partes, **usando TCP para sinalizar pedidos e términos de chamada e UDP ou TCP para o streaming de áudio**. UDP é preferido, mas TCP, junto ao uso de um nó intermediário, é usado em determinadas circunstâncias para contornar firewalls

- **Skype: um exemplo de rede de sobreposição**



Obrigado!!!

Dúvidas?

Exercício Prático

1. Implementar o exemplo de Multicast IP
2. Baseado no exercício anterior, criar uma aplicação de bate-papo entre os alunos da sala baseado em multicast IP.

Exercícios

1. É concebivelmente útil que uma porta tenha vários receptores?
2. Um servidor cria uma porta que ele utiliza para receber pedidos dos clientes. Discuta os problemas de projeto relativos ao relacionamento entre o nome dessa porta e os nomes usados pelos clientes.
3. Por que dados binários não podem ser representados diretamente em XML, por exemplo, como valores em Unicode? Os elementos XML podem transportar strings representados como base64. Discuta as vantagens ou desvantagens de usar esse método para representar dados binários.

WebScraping

WebScraping

WebScraping

Data scraping - is a technique in which a computer program extracts data from human-readable output coming from another program.

WebScraping

Data scraping - is a technique in which a computer program extracts data from human-readable output coming from another program.

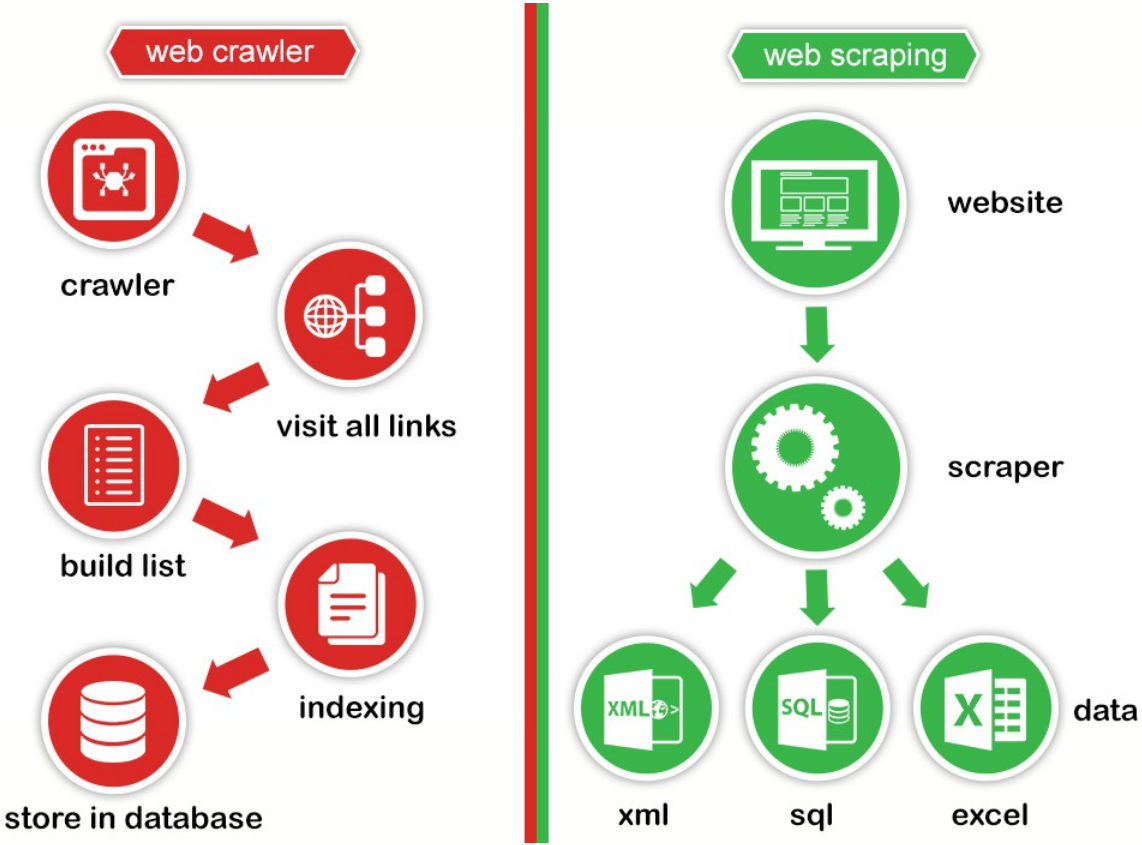
WebScraping

Screen scraping is the method of collecting screen display data from one application and translating it so that another application is able to display it.

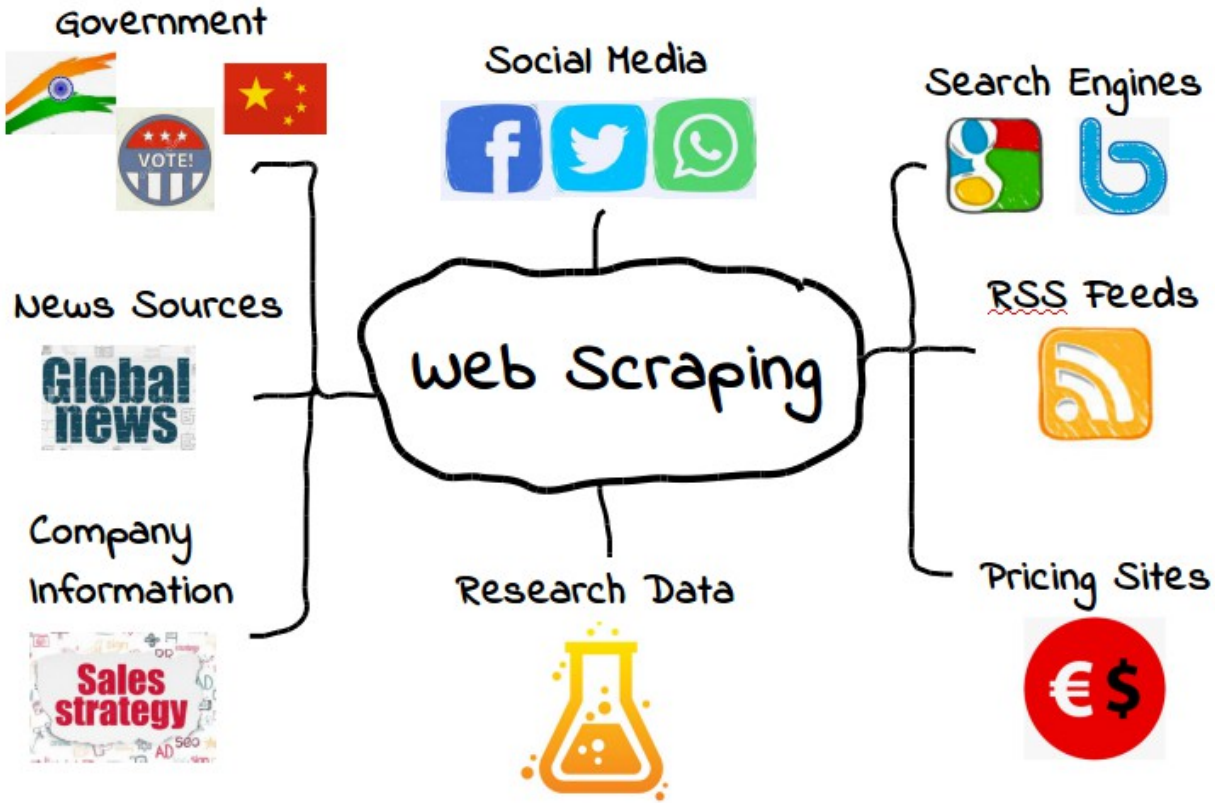
Report mining is the extraction of data from human readable computer reports.

Web scraping is a web technique of extracting data from the web, and turning unstructured data on the web (including HTML formats) into structured data that you can store to your local computer or a database.

WebScraping



WebScraping



WebScraping

https://www.youtube.com/watch?v=n7fob_XVsbY&feature=youtu.be