RESUMO INFRACOM

Capítulo 2 - Camada de Aplicação

Monitor: Carlos Henrique Vieira Marques Veeck
chvmv@cin.ufpe.br

Professor: Kelvin Lopes Dias

2.1: Princípios de aplicações de redes

- As aplicações de rede são uma das principais razões da existência de uma rede de computadores.
- As aplicações vão desde os correios eletrônicos convencionais, a World Wide Web, serviços de streaming, VoIP, jogos on-line e etc.
- Um dos núcleos do desenvolvimento de aplicações de rede é escrever programas que sejam executados em diferentes sistemas finais.
    - Esse software pode ser escrito em diferentes linguagens de programação.
    - Você não precisará se preocupar em escrever programas para os comutadores no network core, pois os mesmos não executam a camada de aplicação.

2.1.1 Arquiteturas de aplicações de rede:

- Antes de começar a codificar, é necessário ter em mente uma arquitetura de aplicação.
    - Essa arquitetura de aplicação é projetada pelo programador e define como a aplicação se organiza nos sistemas finais.
- Duas arquiteturas de aplicação comuns: **cliente-servidor** e **P2P (peer-to-peer).**
    - **Cliente-servidor**: Utiliza um hospedeiro sempre em funcionamento nomeado de **servidor**, que atende as requisições de outros sistemas finais, nomeados **clientes**. Exemplo comum é um servidor Web atendendo requisições de diversos navegadores de clientes.

        - É importante observar que nessa arquitetura, os clientes **não se comunicam diretamente entre si.**
        - O servidor possui um endereço de IP fixo, permitindo que clientes sempre o contatem.
        - Como essa arquitetura utiliza apenas um host como servidor, é comum que ele fique rapidamente **sobrecarregado** de requisições. Por isso, um **datacenter** com diversos hospedeiros muitas vezes é utilizado como servidor para aplicações populares.
    - **P2P**: Nessa arquitetura, existe pouca ou nenhuma dependência de servidores. Em vez disso, a aplicação utiliza comunicação direta entre duplas de hospedeiros conectados alternadamente, chamados de ***peers***. Um exemplo atual é o BitTorrent.
        - Uma de suas principais características é sua **auto-escalabilidade**.
        - Bom custo benefício, pois requerem menos infraestrutura.
        - Devido à sua **descentralização**, podem enfrentar problemas de **segurança e performance**.

2.1.2 Comunicação entre processos:

- Um processo é um programa que está rodando dentro de um sistema final.
    - Quando rodam no mesmo sistema final, se comunicam com regras estabelecidas pelo sistema operacional.
    - Em processos em diferentes hosts, a comunicação é feita por meio do envio e recebimento de **mensagens**.
- Normalmente, cada processo em um par comunicante é rotulado como cliente ou outro como servidor.
    - Na Web, um navegador é um processo cliente e um servidor Web o processo servidor.
    - Num compartilhamento de arquivos P2P, um processo pode atuar como ambos.

No contexto de uma sessão de comunicação entre um par de processos, aquele que inicia a comunicação (i.e., o primeiro a contatar o outro no início da sessão) é rotulado de cliente. *O que espera ser contatado para iniciar a sessão é o* servidor.

- Um processo envia e recebe mensagens por uma interface de software chamada de ***socket***.
    - Funciona como uma porta de entrada e saída para o processo.
    - Também pode ser definido como a interface entre a camada de aplicação e transporte em um hospedeiro.
- Para identificar o processo receptor rodando em outro hospedeiro, é necessário o endereço IP destino e um identificador para o processo. Esse identificador do processo é o **número de porta de destino**.
    - Algumas aplicações possuem números de portas específicos:
        - Servidor Web: Porta 80.
        - Processo Servidor de correio: Porta 25.

2.1.3: Serviços de transporte disponíveis para aplicações :

- Ao desenvolver uma aplicação, você deve escolher algum dos protocolos de transporte disponíveis.
- Cada protocolo pode oferecer diferentes serviços:
    - **Transferência confiável de dados**: Alguns protocolos podem oferecer esse serviço para facilitar a implementação de aplicações que precisam de entrega **garantida** de pacotes, sem perdas significativas.
    - **Vazão**: Com esse serviço, o protocolo pode oferecer uma vazão garantida para garantir o funcionamento de **aplicações sensíveis à largura de banda**.
    - **Temporização**: Serviço atrativo para aplicações que não toleram atrasos, como jogos multiplayer, pois garante que os bits cheguem em um tempo máximo determinado.
    - **Segurança**: Alguns protocolos podem promover a codificação e integridade dos dados enviados.

2.1.4: Serviços de transporte providos pela internet :

- A internet provê dois protocolos de transporte: **UDP e TCP**.
- Serviços do TCP:

    - **Serviço orientado à conexão**: O TCP faz cliente e servidor realizarem uma troca de mensagens conhecida como **handshake** antes que as mensagens comecem a fluir.
    - **Conexão full-duplex**: A conexão originada no handshake permite que cliente e servidor troquem mensagens entre si simultaneamente.
    - **Serviço confiável de transporte**: Os dados enviados são entregues sem erros e na ordem correta.
    - **Controle de congestionamento**: Limita a capacidade de transmissão quando a rede está muito congestionada.
- Serviços do UDP:
    - **Protocolo simplificado e leve.**
    - **Serviço não orientado à conexão:** Não realiza handshake.
    - **Serviço não confiável de entrega de dados**: sem garantias de entrega ou integridade.
    - **Sem controle de congestionamento**: processo originador pode bombear dados para a rede na taxa que quiser.

2.1.5: Protocolos da camada de aplicação:

- Um protocolo de camada de aplicação define a maneira processos de uma aplicação, que funcionam em diferentes sistemas finais, trocam mensagens entre si.
    - Define, mais especificamente:

        - Os tipos de mensagens trocadas; por exemplo, de requisição e de resposta.
        - A sintaxe dos vários tipos de mensagens, tais como os campos da mensagem e como os campos são delimitados.
        - A semântica dos campos, isto é, o significado da informação nos campos.
        - Regras para determinar quando e como um processo envia mensagens e responde a mensagens.

- Alguns exemplos de protocolos e suas aplicações:
    - **SMTP**: Correio eletrônico.
    - **HTTP 1.1**: Web
    - **FTP**: Transferência de arquivos

2.2: A Web e o HTTP.

- Inicialmente, a internet era apenas utilizada em instituições de pesquisa.
- Com o surgimento da aplicação Web, ela foi alavancada para outro patamar de popularidade.

2.2.1 Descrição geral do HTTP:

- O HTTP é o protocolo da camada de aplicação da Web.
    - É executado em dois programas: um cliente (pode ser um navegador web) e um servidor (servidores web, como o Apache), em sistemas finais diferentes.
    - Esses dois programas trocam mensagens HTTP entre si.
- Terminologia Web:
    - Página Web: constituída por objetos, geralmente constituída por um arquivo base HTML.
    - Objeto: Um arquivo que pode ser acessado com um URL.
- O HTTP define como esses clientes podem fazer a requisição de páginas aos servidores, e como essas mesmas são enviadas de volta.
- **O HTTP usa o TCP** como seu protocolo de transporte.
- O HTTP é um **protocolo sem estado**, pois não mantém informações sobre os clientes armazenadas.

2.2.2 Conexões persistentes e não persistentes:

- **Conexões Persistentes**: Todas as requisições e suas respostas são enviadas por uma mesma conexão TCP.
- **Conexões Não Persistentes**: Cada par requisição/resposta é enviado por conexões TCP diferentes

- Numa conexão não persistente, cada objeto sofrerá **2 RTTs** (um para estabelecer a conexão e outro para solicitar e receber um objeto). Nas persistentes, não temos esse problema, já que as conexões só são fechadas caso passem um tempo ociosas.
- Por padrão, o HTTP usa **conexões persistentes com paralelismo**.

2.2.3 Formato da mensagem HTTP:

- **Mensagem de requisição HTTP:**

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

- **Componentes:**
    - **Linha de requisição:** a primeira linha da mensagem é composta de 3 campos:
        - Método: GET, POST, HEAD, PUT e DELETE
        - URL
        - Versão do HTTP
    - **Linhas de cabeçalho:**
        - Host: especifica o hospedeiro que o objeto reside.
        - Connection: define se a conexão deve ou não ser fechada.
        - User-agent: o navegador que faz a requisição.
        - Accept-language: preferência de idioma do objeto recebido.
- **Mensagem de resposta HTTP:**

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(dados dados dados dados dados ...)
```

- **Componentes:**
    - **Linha inicial:** A primeira linha da mensagem é composta de 3 campos:
        - Versão do protocolo.
        - Código de estado.
        - Mensagem de estado correspondente.
    - **Linhas de cabeçalho:**
        - Connection.
        - Date: data e hora que a resposta HTTP foi criada e enviada pelo servidor.
        - Server: indica o servidor Web que gerou a mensagem.
        - Last-modified: hora da criação ou última alteração do objeto.
        - Content-length: número de bytes do objeto.
        - Content-type: mostra o tipo de objeto presente no corpo da resposta. (ex: texto HTML).
    - **Corpo da entidade**
- **Códigos de estado e suas mensagens:**

- `200 OK`: requisição bem-sucedida e a informação é entregue com a resposta.
- `301 Moved Permanently`: objeto requisitado foi removido em definitivo; novo URL é especificado no cabeçalho `Location:` da mensagem de resposta. O *software* do cliente recuperará automaticamente o novo URL.
- `400 Bad Request`: código genérico de erro que indica que a requisição não pôde ser entendida pelo servidor.
- `404 Not Found`: o documento requisitado não existe no servidor.
- `505 HTTP Version Not Supported`: a versão do protocolo HTTP requisitada não é suportada pelo servidor.

2.2.4 Interação usuário-servidor: *cookies*:

- ***Cookies*** permitem que sites monitores os usuários.
- Quatro componentes:
    - Linha de cabeçalho de cookie na resposta HTTP.
    - Linha de cabeçalho de cookie na requisição HTTP.
    - Um arquivo de cookie mantido no sistema final do usuário.
    - Banco de dados de apoio no site.

- Por meio de um **número de identificação** designado ao usuário no primeiro acesso, um site consegue monitorar a atividade de um usuário sem necessariamente saber quem é esse usuário.
    - São mantidas informações como páginas visitadas, ordem e horários.
- Pode ser utilizado, por exemplo, para oferecer um serviço de carrinho de compras ou um serviço de recomendações de produtos personalizadas.

2.2.5 Caches Web:

- Um ***cache Web,*** também conhecido como **servidor *proxy*** é uma entidade de rede que atende requisições HTTP no lugar de um servidor Web de origem.
    - Um cache possui um disco próprio e armazena cópias de objetos recentemente requisitados.
    - Dessa forma, o cache serve para agilizar algumas requisições mais comuns.
- O navegador, antes de contatar o servidor de origem, faz uma conexão com o cache para verificar se o objeto já está armazenado.
    - Caso negativo, o cache inicia uma conexão TCP com o servidor de origem e envia uma requisição HTTP, procurando pelo objeto. Ao receber, pode armazenar uma cópia desse objeto para atender a futuras requisições.
- O cache pode ser, ao mesmo tempo, servidor e cliente;
    - Servidor quando recebe requisições de um navegador e envia respostas.
    - Cliente quando envia requisições a um servidor de origem.
- Principais razões para utilização de caches:
    - Potencial redução de tempo de resposta para requisições de clientes.
    - Pode reduzir o tráfego de enlace na rede de uma instituição, evitando atrasos e garantindo economia de recursos financeiros, uma vez que menos dinheiro precisaria ser direcionado para aumento de largura de banda.
- **GET Condicional:**
    - Mecanismo do protocolo HTTP para verificar se as cópias de objetos no cache estão atualizadas.
        - Uma mensagem de requisição HTTP é considerada um GET condicional se:
            - Usa o método GET.

        - Possui a linha de cabeçalho If-Modified-Since.
    - A linha de cabeçalho anteriormente citada serve para dizer ao servidor para apenas enviar o objeto caso ele tenha sido modificado desde a data especificada.

2.2.6 HTTP/2:

- Primeira nova versão do HTTP.
- Principais objetivos:
    - Redução de latência.
    - Permitir a priorização de solicitações.
    - Oferecer compressão mais eficientes dos campos de cabeçalho.
- Motivação: **HOL blocking**.
    - Quando um pacote maior atrasa a transmissão de diversos outros pacotes no enlace.
- Enquadramento do HTTP/2:
    - Para solucionar o HOL blocking, cada mensagem é dividida em quadros menores.
    - Também são intercaladas as mensagens de solicitação e resposta numa mesma conexão.
    - Além disso, é realizada a codificação binária desses quadros, permitindo uma leitura mais eficiente.
- HTTP/3:
    - Implementação do protocolo QUIC, utilizando os avanços do HTTP/2.

2.3: Correio eletrônico na internet.

- Uma das primeiras e mais populares aplicações da internet.
- Componentes principais:
    - Agentes de usuário: Permitem que usuários leiam, escrevam e respondam e-mails. (Outlook, Gmail, Yahoo...)
    - Servidores de correio: O núcleo da infraestrutura do e-mail. Cada usuário possui uma “caixa postal” em um desses servidores, que administra e guarda as mensagens enviadas a ele.

- **SMTP** - Simple Mail Transfer Protocol

2.3.1 SMTP:

- Principal protocolo da camada de aplicação do correio eletrônico.
  - Utiliza o **serviço confiável** de transporte do TCP.
- Tecnologia antiga na internet, possuindo algumas características arcaicas.
  - Limita o corpo de todas as mensagens de correio ao formato ASCII, por exemplo.
- Funcionamento:

1. Alice chama seu agente de usuário para e-mail, fornece o endereço de Bob (p. ex., `bob@someschool.edu`), compõe uma mensagem e instrui o agente de usuário a enviá-la.
2. O agente de usuário de Alice envia a mensagem para seu servidor de correio, onde ela é colocada em uma fila de mensagens.
3. O lado cliente do SMTP, que funciona no servidor de correio de Alice, vê a mensagem na fila e abre uma conexão TCP para um servidor SMTP, que funciona no servidor de correio de Bob.
4. Após alguns procedimentos iniciais de apresentação (*handshaking*), o cliente SMTP envia a mensagem de Alice pela conexão TCP.
5. No servidor de correio de Bob, o lado servidor do SMTP recebe a mensagem e a coloca na caixa postal dele.
6. Bob chama seu agente de usuário para ler a mensagem quando for mais conveniente para ele.

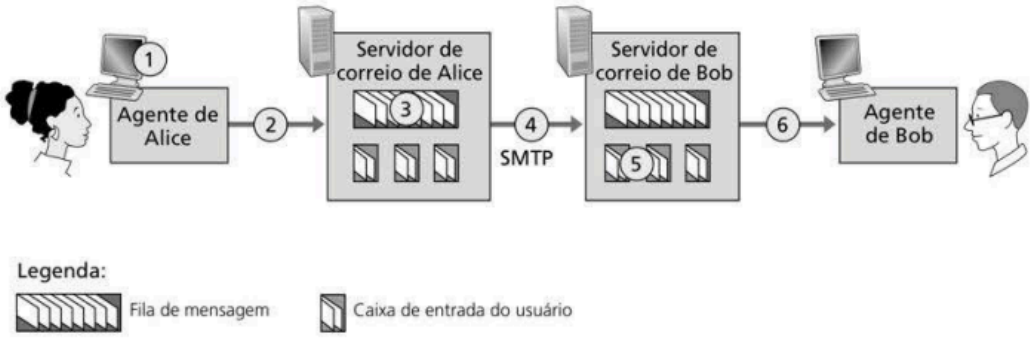


Figura 2.15 Alice envia uma mensagem a Bob.

- Em geral, o SMTP **não utiliza servidores intermediários** para enviar correspondências, mesmo quando os servidores comunicantes estão muito distantes.

2.3.2 Formatos de mensagem de correio:

Um cabeçalho de mensagem típico é semelhante a:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

2.3.3 Protocolos de acesso ao correio:

- Diferentes protocolos podem ser utilizados no processo do envio de um email.
    - Para um usuário enviar um email de sua máquina para o seu servidor de correio, pode ser utilizado o HTTP ou SMTP.
    - Na transferência do seu servidor para o do destinatário, é usado o SMTP.
    - Para o destinatário acessar a correspondência, pode ser utilizado o HTTP ou o **IMAP - Internet Mail Access Protocol**.

2.4: DNS: O Serviço de diretório da Internet.

- Hospedeiros na internet podem ser identificados pelo seu ***hostname*** e **endereços IP**.
    - Um endereço IP é constituído de 4 bytes, no qual cada ponto separa um dos bytes expressos, em notação decimal de 0 a 255. Dizemos que é um endereço hierárquico pois ao examiná-lo da direita para a esquerda, podemos verificar informações mais específicas sobre onde o host está localizado.

2.4.1 Serviços fornecidos pelo DNS:

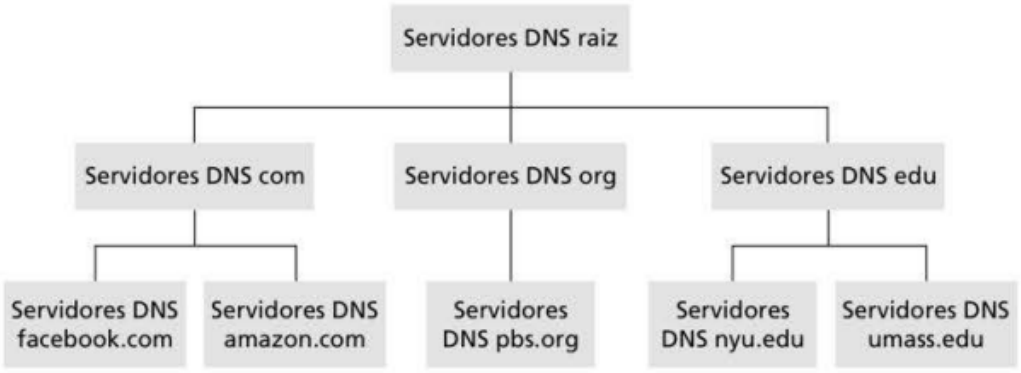
- A principal função do DNS é **traduzir hostnames para endereços IP**.

- É um banco de dados distribuído executado em uma hierarquia de servidores DNS e é um protocolo que permite usuários acessarem tal banco.

- Outros serviços envolvem:
  - **Apelidos (aliasing) de hospedeiro:** Simplificação do nome de hospedeiro para facilitar o acesso.
  - **Apelidos de servidor de correio.**
  - **Distribuição de carga:** Distribui a carga entre servidores replicados em sites movimentados.

2.4.2 Visão geral do funcionamento do DNS:

- Para aplicações traduzirem hostnames em endereços IP, eles chamam o lado cliente do DNS, especificando o nome do cliente que precisa ser traduzido, que envia uma mensagem de consulta para a rede. Após um certo atraso, o DNS recebe uma resposta do servidor com o mapeamento desejado.
- No ponto de vista do cliente, o DNS é uma caixa preta que oferece um serviço de classificação simples, mas na verdade, possui muitas complexidades.
  - Múltiplos servidores distribuídos ao redor do mundo.
- **Hierarquia** do DNS:
  - DNS Raiz
  - TLD (Top Level Domain)
  - Autoritativos



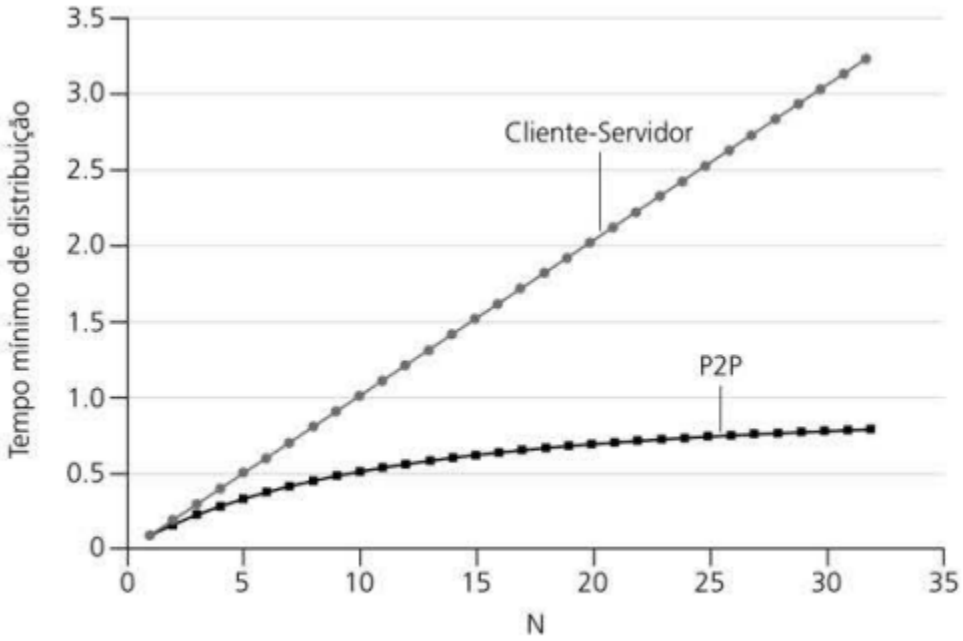
    - Além desses, existem os servidores DNS Locais, que não pertencem diretamente à hierarquia. Esses servidores trabalham juntamente com as ISPs.
- No DNS, temos consultas iterativas e recursivas.
- Cache DNS:
    - Melhora extensivamente o desempenho do DNS.
    - Quando um servidor DNS recebe uma resposta DNS, o mesmo pode colocar as informações da memória em uma cache local.
    - Se um par nome/hospedeiro estiver em cache e outra consulta chegar exigindo o mesmo nome, o servidor pode disponibilizar esse mapeamento, **mesmo que não tenha nível de autoridade para isso**.
- Registros e mensagens DNS:
    - Um Registro de recurso DNS é uma tupla de quatro elementos que contém os seguintes campos:

```
(Name, Value, Type, TTL)
```

    - Os valores de Name e Value dependem do Type:
        - Type A: Name é um nome de hospedeiro e Value o endereço IP para esse nome.
        - Type NS: Name é um domínio e Value é um nome de servidor autoritativo que sabe como obter os endereços IP dos hosts desse domínio.
        - Type CNAME: Value é o nome canônico para o apelido de hostname em Name.
        - Type MX: Nome canônico de um servidor de correio cujo apelido está em Name.
- Ambas as mensagens de consulta e resposta DNS possuem o mesmo formato.
- Entidades Registradoras: Entidades responsáveis por verificar se um nome de domínio que está sendo criado é exclusivo e por registrar o mesmo no banco de dados do DNS.

2.5: Distribuição de arquivos P2P.

- Escalabilidade de arquiteturas P2P:
    - Como visto anteriormente, cada par de hospedeiros pode ajudar o servidor na distribuição de um arquivo, podendo usar sua própria capacidade de upload para isso, tornando a distribuição mais eficaz.



2.23 Tempo de distribuição para arquiteturas P2P e cliente-servidor.

2.7: Programação com Sockets: criando aplicações de rede.

- Muitas aplicações de rede consistem em um par de programas cliente e servidor, que residem em sistemas finais diferentes.
    - Quando estes programas são executados, cria-se um processo cliente e outro servidor, que se comunicam através de seus sockets.
    - Podem ser usados sockets TCP ou UDP na hora de desenvolver uma aplicação, cada uma com suas vantagens e desvantagens.
- Para ser enviado para a rede, o pacote precisa das seguintes informações:
    - Endereços IP de origem e destino.

    - Números de porta de origem e destino.
- Principais diferenças entre Sockets UDP e TCP:
    - Como já foi visto, o TCP é um protocolo orientado à conexão, então utiliza algumas funções como o **.connect** pelo lado do cliente e **.listen** pelo lado do servidor, que vão possibilitar a comunicação por meio dessa conexão TCP estabelecida.