PRC 2009 – Resumo de Aula
Camada de Aplicação – Protocolos HTTP e FTP

- ## A camada de Aplicação

  - **Aplicação é o objetivo final das redes. Foram as aplicações criativas que motivaram o aparecimentos das redes desde os anos 80**
    - **texto e dados (telnet, e-mail, ftp, etc ...)**
    - **mensagens instantâneas, chat**
    - **compartilhamento de arquivos**
    - **multimedia (www, telefonia IP, telefonia via internet, audio/vídeo armazenado e em audio/vídeo em tempo real, videoconferência.**
    - **algumas vantagens**

  - **Arquiteturas das aplicações**
  - **Arquitetura Cliente – Servidor.**
    - **A maioria das aplicações tem essa arquitetura. O cliente inicia o processo.**
    - **Exemplos: WEB (há o processo cliente=browser e o processo servidor=servidor de páginas), Telnet (login remoto), transferência de arquivos, e-mail**
  - **Arquitetura P2P – Peer-to-Peer (Par a Par)**
    - **Não há uma hierarquia. Ambos os lados podem iniciar o processo.**
    - **Intrinsecamente escalável**
    - **Exemplos: GNUTELLA – compartilhamento de arquivos e códigos fonte abertos**
  - **Arquitetura Mixta – CS e P2P**
    - **Em geral, o processo se inicia com CS e depois passa a P2P. Há um servidor central para o estabelecimento da conexão e depois os dados são trocados diretamente entre cada par.**
    - **Exemplos: Napster (não tem mais), Mensagem instantânea (MSN, Yahoo Messenger, etc.), Skype**

- **Requisitos da camada de transporte para a camada de aplicação**
    - **a camada de transporte é acessada através de APIs (Application Program Interface) que na internet tem o nome de Socket**
    - **a aplicação precisa ser analisada segundo 3 características: perda de dados (algumas podem perder um pouco e outras não), tempo de resposta (algumas são mais tolerantes) e velocidade (algumas precisam uma velocidade mínima garantida)**
    - **O que é oferecido?**
        - **TCP – mecanismo de controle de fluxo e congestionamento**
            - **Garantia de entrega dos pacotes**
            - **Não garante atrasos (tempo de resposta)**
            - **Não garante velocidade mínima**
        - **UDP – não tem mecanismos de controle de fluxo e congestionamento**
            - **Não garante nada – nem entrega, nem atraso e nem velocidade mínima**
            - **É mais eficiente**
    - **Nem TCP nem UDP oferecem velocidade mínima**
        - **Aplicações que precisam de velocidade mínima (voz e vídeo) não podem rodar na Internet?**
        - **Podem – basta haver tolerância do usuário**

- **A WEB e o HTTP (versões 1.0 e 1.1)**
  - **Lado cliente (browser) e lado servidor (web browser)**
  - **Página WEB – formada por objetos (textos, imagens JPEG, GIF)**
    - **Em geral: um arquivo base HTML e diversos outros objetos referenciados por este HTML**
  - **Uma pequena história dos browsers (1992-1998)**
    - **Netscape vs Internet Explorer**
  - **O protocolo HTTP (versão 1.0 - RFC 1945 e versão 1.1 – RFC 2616)**
    - **Usa o TCP como transporte**
    - **sem estado**
    - **persistente (vários objetos são solicitados pelo browser dentro da mesma conexão TCP) e não persistente**
    - **persistente com e sem paralelismo. Com paralelismo, quando um objeto é solicitado pelo browser sem que o anterior ainda tenha chegado**
    - **default no http 1.1 – persistente e com paralelismo**

- **As mensagens HTTP**
  - **Mensagens de requisição de páginas e de resposta à requisições**
  - **Formato (Método; URL e parâmetros)**
  - **Métodos**
    - **GET - requisita página (o mais comum)**
    - **POST – envia também os dados digitados pelo usuário (formulário)**
    - **HEAD - igual ao GET mas não devolve o objeto – usado para depuração**
    - **Outros métodos no http/1.1**
  - **Mensagens de resposta (versão http, Código de retorno, parâmetros e a página)**
  - **Protocolo baseado em mensagens de texto claras e possíveis de serem lidas no vídeo**
    - **Quem tentou um telnet na porta 80?**
      - `>telnet www.ime.usp.br 80`
      - `GET /~mms/mac4481s2009/mac4481s2009.html HTTP/1.0 <enter>`
      - `Host: www.ime.usp.br <enter>`
      - `<enter>`
    - **Substitua GET por HEAD**
    - **Requisite uma página inválida**

- **Outros usos do http**
  - **Usado também para transferir dados de um servidor para um cliente.**
  - **Cliente não necessariamente um Browser.**
  - **Dados XML, VoiceXML, etc.**

- **Identificação ou autenticação de usuários no protocolo HTTP**
  - **Cookies – RFC 2109**
    - **Maneira de associar um usuário a um site. O site fica sabendo toda vez que este usuário o acessa.**
    - **Quando um usuário acessa um site, recebe a resposta com o parâmentro:**
      - `Set-cookie: 156478` **(número de identificação)**
    - **O cliente recebe e armazena essa identificação junto com seu arquivo de controle dessa página. Toda vez que esse usuário acessa novamente esse site, envia o parâmetro:**
      - `Cookie: 156478`
    - **Uma maneira fraca de identificação, mas considerando que o http é stateless isso funciona.**
    - **Problema – violação de privacidade**

- **CACHE**

  - **Web caches = proxy servers**

    - **mantém cópia dos objetos recentemente requisitados**
    - **o próprio browser tem um primeiro nível de cache**
    - **o próximo nível é o servidor de cache da rede**
    - **O servidor de cache é servidor e cliente, pois caso não tenha o elemento requisitado ele próprio requisita o mesmo**
    - **vantagens (tempo de resposta e redução do tráfego)**

  - **Vários tipos**

    - **sofisticados e caros**
    - **simples e baratos (alguns rodam num PC e são software free)**

  - **Hierarquia de caches (mais de 1 nível)**

  - **CDN – Content Delivery Network**

    - **Rede de Distribuição de Conteúdo baseada em servidores de conteúdo distribuidos, de forma que o ususário possa acessar o mais próximo**
    - **Solução para distribuição de vídeo sob demanda**
      - **Exemplo: block buster sem loja**
    - **É uma solução em busca de um problema**
    - **Não há demanda ainda para este tipo de serviço e a solução ficou apenas na tentativa e com um monte de investimento perdido.**

- **O comando GET condicional**
  - **atributo if-modified-since no GET**
  - **resposta com a página ou com mensagem:** `304 Not Modified`
  - **Mecanismo:**
    - **Browser consulta pela primeira vez uma página.**
    - **Envia ao servidor cache que não tem a página e portanto solicita a mesma a frente.**
    - **Servidor cachê recebe página com parâmetro:**
      - `Last-Modified: Wed, 16 Ago 2008 04:05:33`
    - **Próxima solicitação desta página – o servidor cachê envia o parâmetro**
      - `If-Modified-Since: Wed, 16 Ago 2008 04:05:33`
    - **Se não houve modificação após essa data, o servidor não envia a página e sim apenas os cabeçalhos**

- **O Protocolo FTP – File Transfer Protocol – RFC 959**
  - **protocolo para transferência de arquivos**
  - **guarda o estado do usuário (diretórios local e remoto, user/psw)**
  - **usa 2 portas lógicas**
    - **porta 21 – comandos – persistente**
    - **porta 20 – dados - não persistente (sessão a cada arquivo)**
  - **o formato é totalmente texto, como no HTTP**
  - **Existem variações descritas em outras RFCs**
    - **para garantir maior segurança (criptografia e outros) alguns protocolos devirados apareceram: SFTP (secure FTP) por exemplo**
  - **Aplicações de FTP**
    - **ativadas pelo usuário**
      - **versão de texto**
        - **ftp** ftp.ime.usp.br **(acesso ao site ftp do ime)**
      - **versão gráfica – veja ssh SFTP**
    - **O protocolo FTP pode ser usado para construir outras aplicações que envolvem transferência de dados em geral e que rodam automaticamente a medida que os dados vão sendo criados, por temporização, etc ...**
      - **Exemplo: coletor de dados. A cada 1 hora por exemplo, abre uma sessão FTP com o servidor e transfere os dados coletados até o momento.**