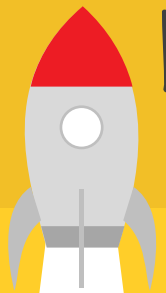
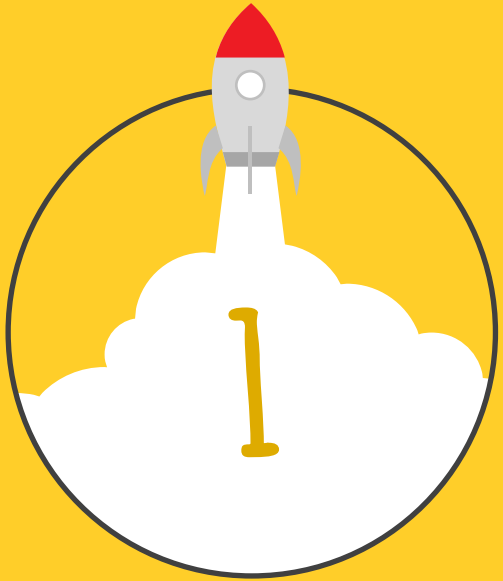
LRODRIGO



104192 - Redes de Computadores

Aula 5– Camada de Aplicação

Luis Rodrigo – luis.goncalves@ucp.br

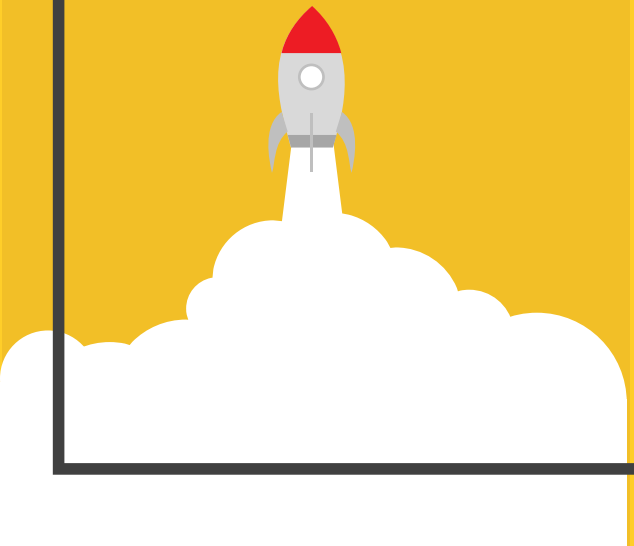


Camada de Aplicação:

Aplicações de Protocolos

Camada de Aplicação

Aplicações e Protocolos



Aplicação: processos distribuídos em comunicação

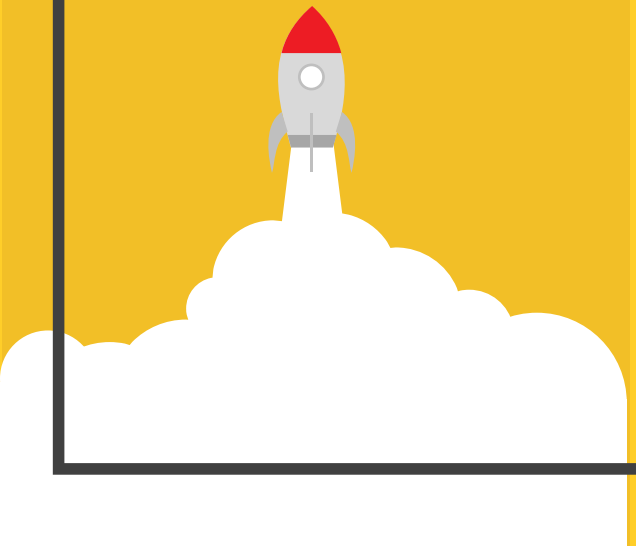
- Rodam nos computadores da rede como programas de usuário
- Trocam mensagens para realização das atividades
- Exemplo: email, ftp, Web

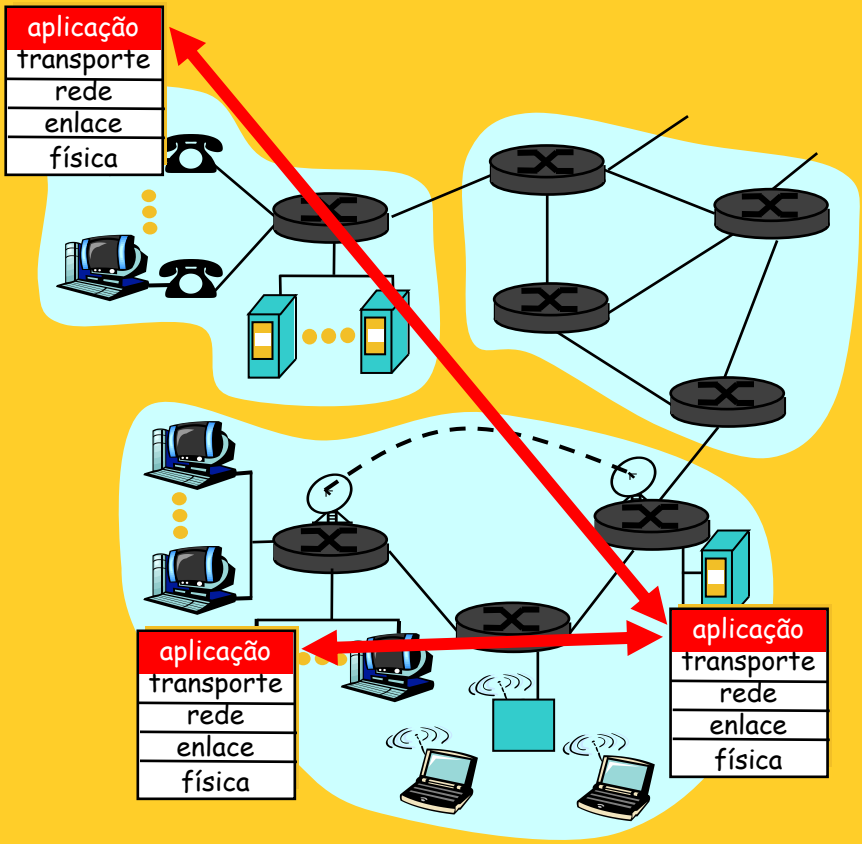
Protocolos de aplicação

- Fazem parte das aplicações
- Definem mensagens trocadas e as ações tomadas
- Usam serviços de comunicação das camadas inferiores

Camada de Aplicação

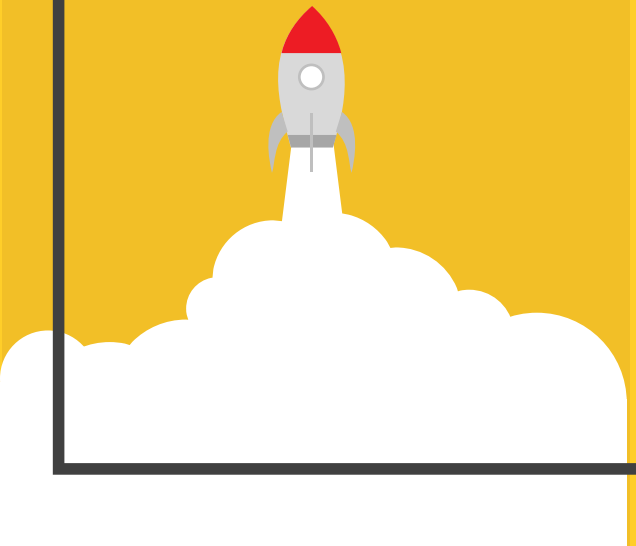
Aplicações e Protocolos





Camada de Aplicação

Aplicações e Protocolos



Processo: programa executando num host.

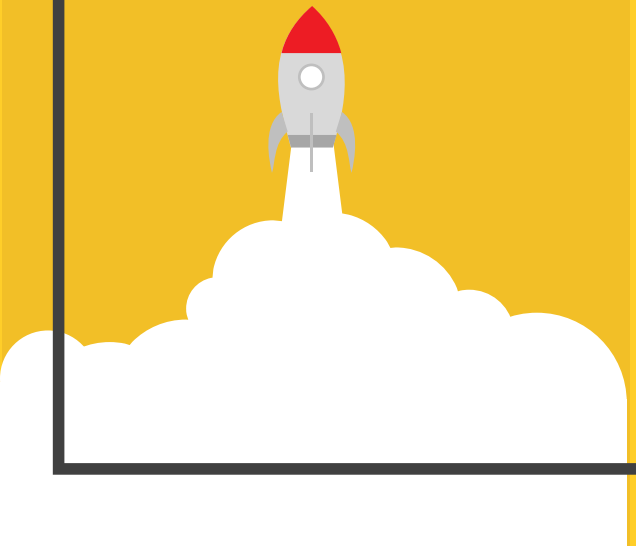
- Dentro do **mesmo host**: interprocess communication (definido pelo OS).
- Executando em **diferentes hosts** se comunicam com um protocolo da camada de aplicação

Agente usuário: fornece a interface entre o usuário e a rede.

- Implementa protocolo da camada de aplicação
- **Web**: browser
- **E-mail**: leitor de correio
- **streaming audio/video**: media player

Camada de Aplicação

Aplicações e Protocolos



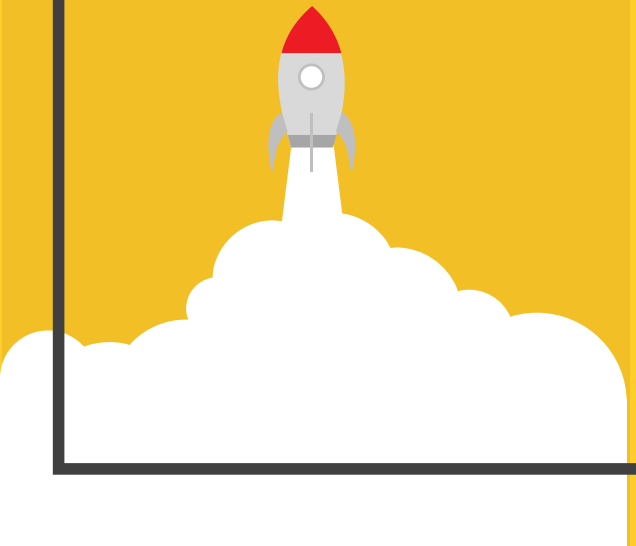
Aplicações de rede típicas têm duas partes:
cliente e *servidor*

Cliente:

- Inicia comunicação com o servidor ("fala primeiro")
- Tipicamente solicita serviços do servidor,
  - **Web**: cliente implementado no browser;
  - **e-mail**: leitor de correio

Camada de Aplicação

Aplicações e Protocolos



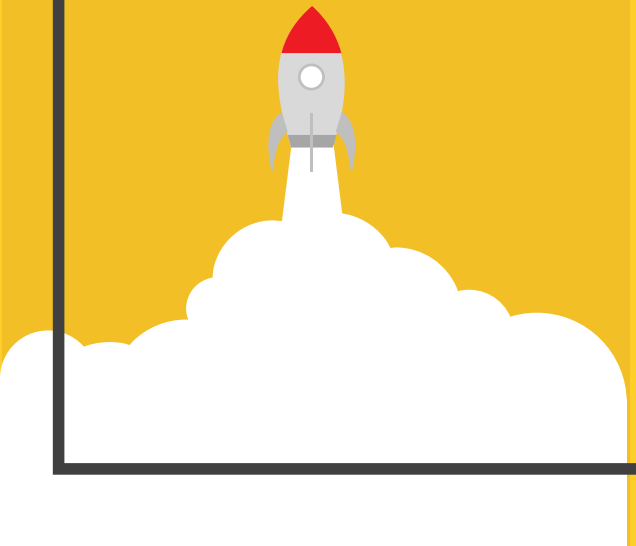
Aplicações de rede típicas têm duas partes:
cliente e *servidor*

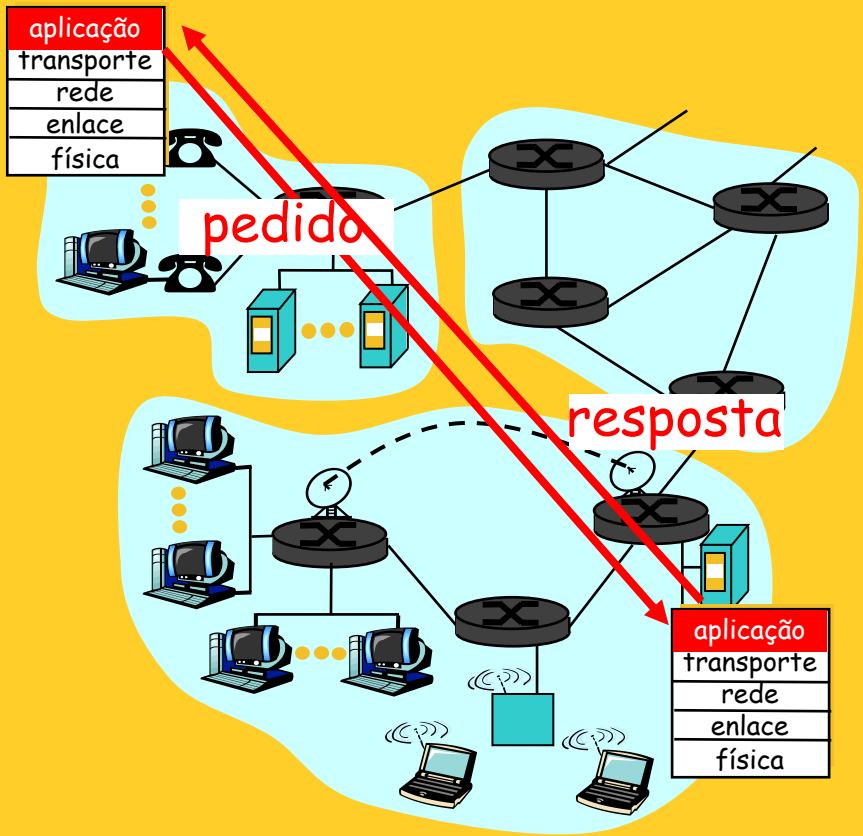
Servidor:

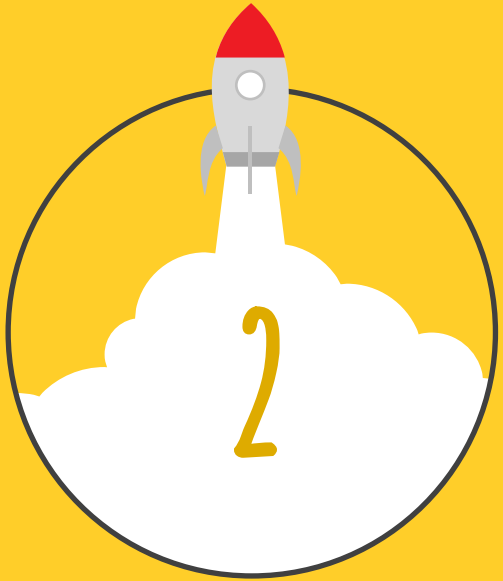
- Fornece os serviços solicitados ao cliente
- **Web server:** envia a página Web solicitada,
- **Servidor de e-mail**: envia as mensagens, etc.

Camada de Aplicação

Aplicações e Protocolos





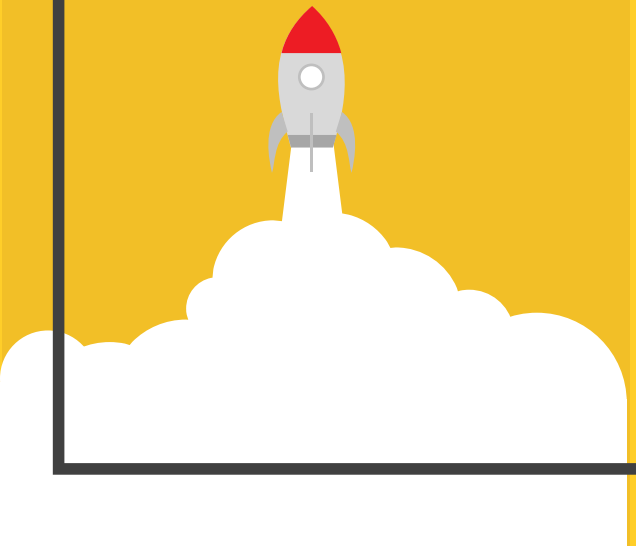


Camada de Aplicação:

Interfaces de Programação

Camada de Aplicação

Interfaces de Programação



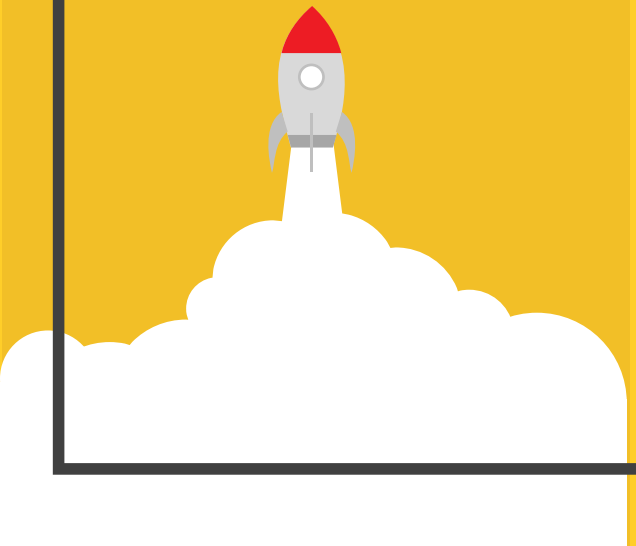
API: application programming interface

- Define a interface entre a camada de aplicação e de transporte
- Socket: Internet API
  - dois processos se comunicam enviando (**send**) dados para o **socket** e lendo (**recv**) dados de dentro do **socket**

Camada de Aplicação

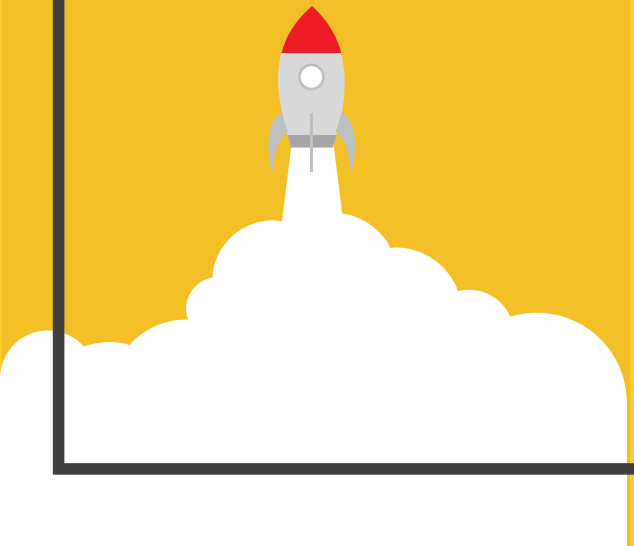
Interfaces de Programação

Como um processo "**identifica**" o outro processo com o qual ele quer se comunicar?



Camada de Aplicação

Interfaces de Programação



Como um processo "**identifica**" o outro processo com o qual ele quer se comunicar?

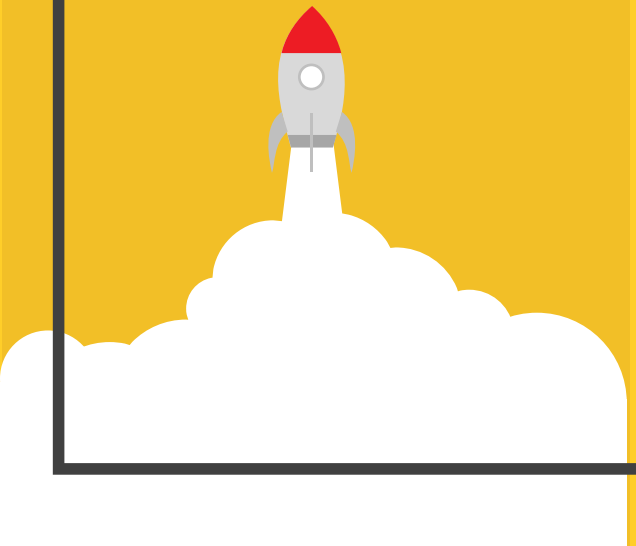
- IP address do computador no qual o processo remoto executa
- Port Number permite determinar o processo local para o qual a mensagem deve ser entregue.

Camada de Aplicação:

Serviços de Transporte

Camada de Aplicação

Serviços de Transporte



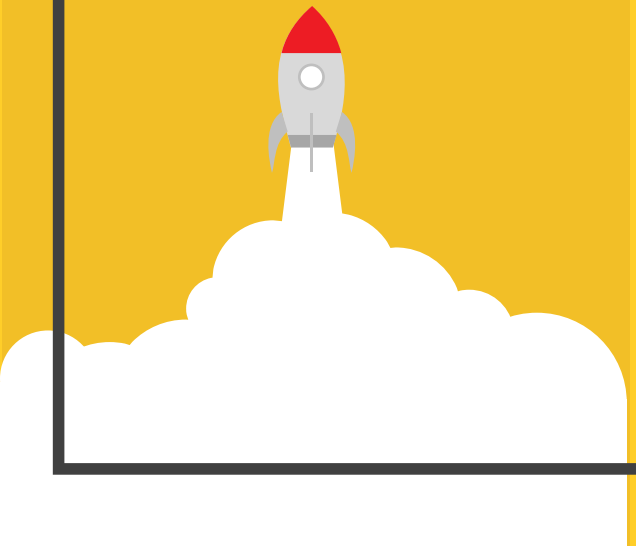
Perda de dados

- Algumas aplicações (**áudio**) podem tolerar alguma perda

- Outras aplicações (**transferência de arquivos, telnet**) exigem transferência de dados 100% confiável

Camada de Aplicação

Serviços de Transporte

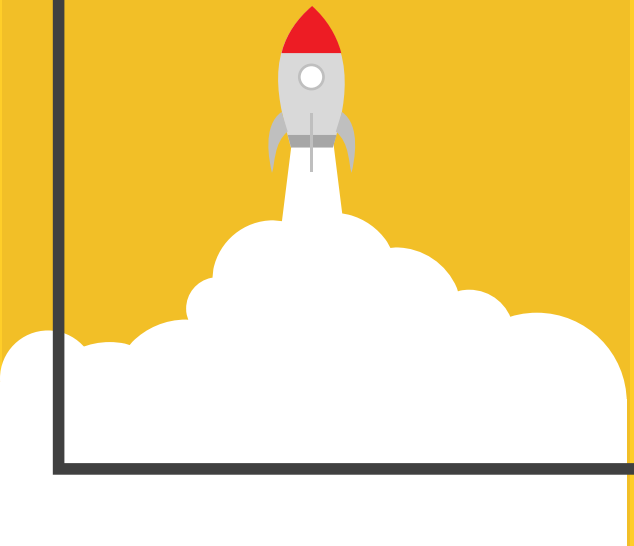


Temporização

- Algumas aplicações (**telefonia Internet, jogos interativos**) exigem baixos atrasos para operarem

Camada de Aplicação

Serviços de Transporte



Banda Passante

- algumas aplicações (**multimídia**) exigem uma banda mínima para serem utilizáveis
- outras aplicações (**"aplicações elásticas"**) melhoram quando a banda disponível aumenta

Camada de Aplicação

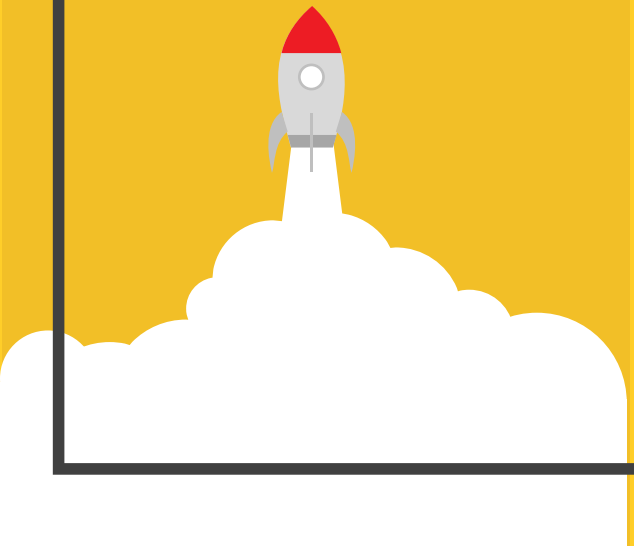
Serviços de Transporte

Requisitos de Transporte de Aplicações Comuns

Aplicação	Perdas	Banda	Sensível ao Atraso
Transf. de Arq.	sem perdas	elástica	não
e-mail	sem perdas	elástica	não
Documentos Web	tolerante	elástica	não
Real-time áudio/vídeo	tolerante	áudio: 5Kb-1Mb vídeo:10Kb-5Mb	sim, 100's msec
Stored áudio/vídeo	tolerante	igual à anterior	sim, segundos
Jogos interativos	tolerante	??? Kbps	sim, 100's msec
e-business	sem perda	elástica	sim

Camada de Aplicação

Serviços de Transporte da Internet

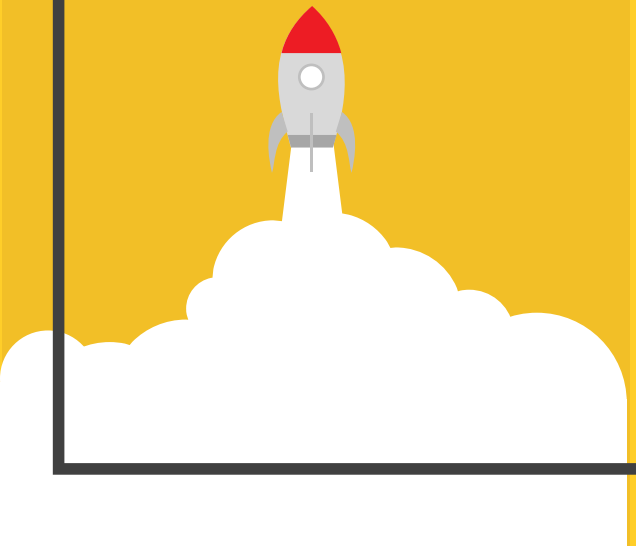


Serviço TCP:

- *orientado á conexão:* conexão requerida entre cliente e servidor
- *transporte confiável:* dados perdidos na transmissão são recuperados
- *controle de fluxo:* compatibilização de velocidade entre o transmissor e o receptor
- *controle de congestionamento:* protege a rede do excesso de tráfego
- *não oferece:* garantias de temporização e de banda mínima

Camada de Aplicação

Serviços de Transporte da Internet



Serviço UDP

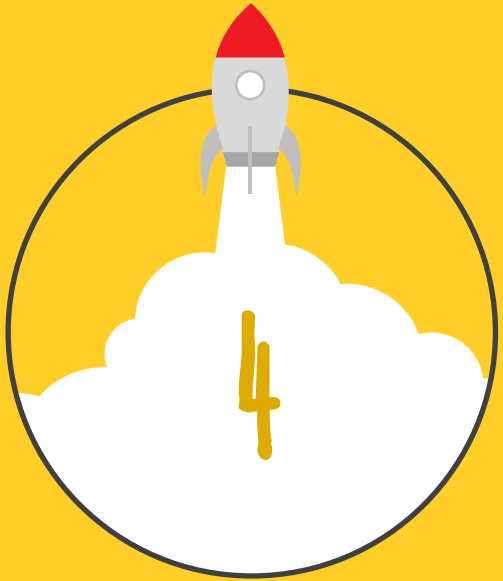
- Transferência de dados **não confiável** entre os processos transmissor e receptor
- **Não oferece**:
  - estabelecimento de conexão,
  - confiabilidade,
  - controle de fluxo e de congestionamento,
  - garantia de temporização e
  - de banda mínima.

Camada de Aplicação

Serviços de Transporte

Aplicações e Protocolos de Transporte da Internet

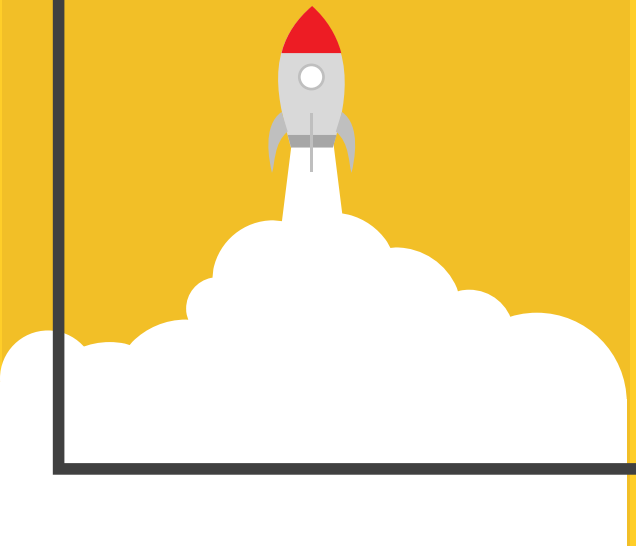
Aplicação	Protocolo de Aplicação	Protocolo de Transporte
e-mail	smtp [RFC 821]	TCP
Acesso de terminais remotos	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
Transferência de arquivos	ftp [RFC 959]	TCP
streaming multimídia	RTP ou proprietário (e.g. RealNetworks)	TCP ou UDP
Servidor de arquivos remoto	NSF	TCP ou UDP
Telefonia via Internet	RTP ou proprietário (e.g., Vocaltec)	tipicamente UDP



Camada de Aplicação:

Protocolo HTTP

Camada de Aplicação
Protocolo HTTP

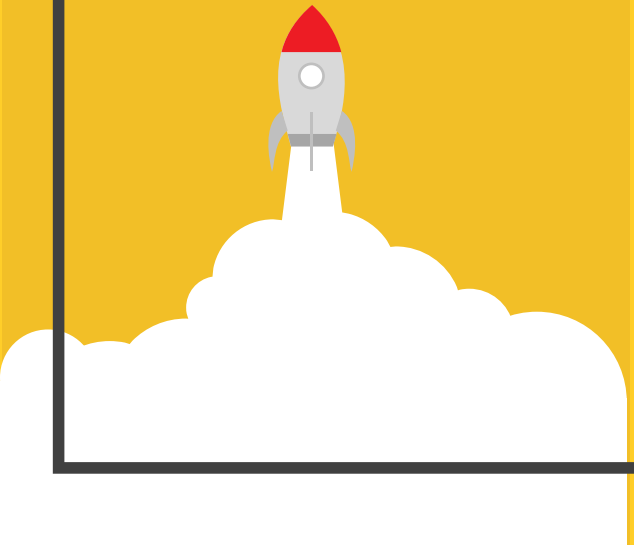


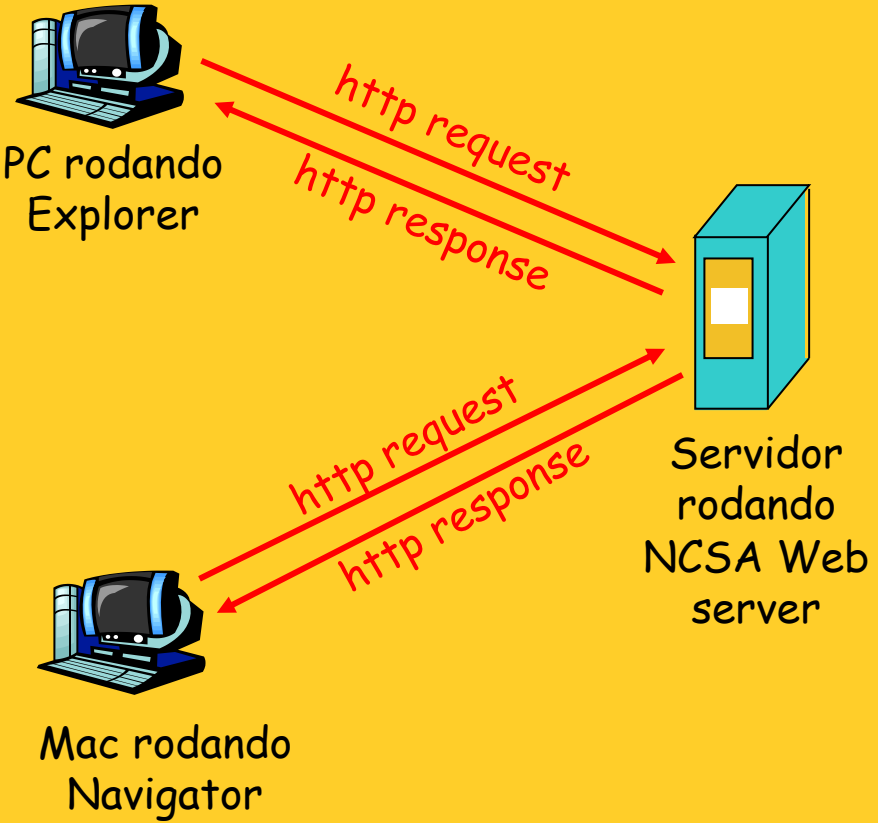
HTTP: hypertext transfer protocol

- Protocolo **WEB** da camada de aplicação
- **Modelo cliente/servidor**
  - *cliente:* **browser** que solicita, recebe e apresenta objetos da Web
  - *server:* envia objetos em resposta a pedidos
- **Padronização**
  - http1.0: RFC 1945
  - http1.1: RFC 2068

Camada de Aplicação

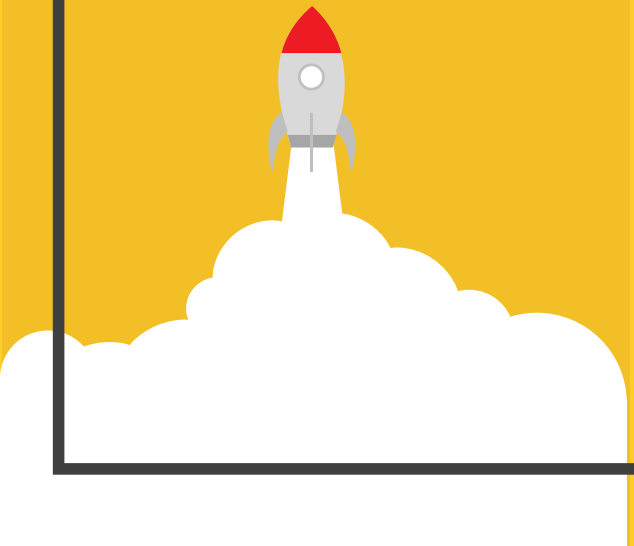
Protocolo HTTP





Camada de Aplicação

Protocolo HTTP

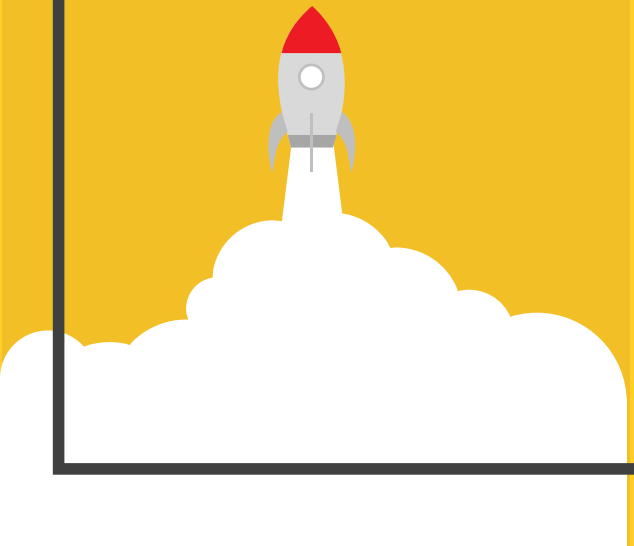


Uso do protocolo de transporte TCP

- **Cliente inicia** conexão TCP (**cria socket**) para o servidor na porta 80
- **Servidor** aceita uma conexão TCP do cliente
- Mensagens **http** (mensagens do protocolo de camada de aplicação) são trocadas entre o **browser** (cliente http) e o **servidor Web** (servidor http)
- A conexão TCP é fechada

Camada de Aplicação

Protocolo HTTP

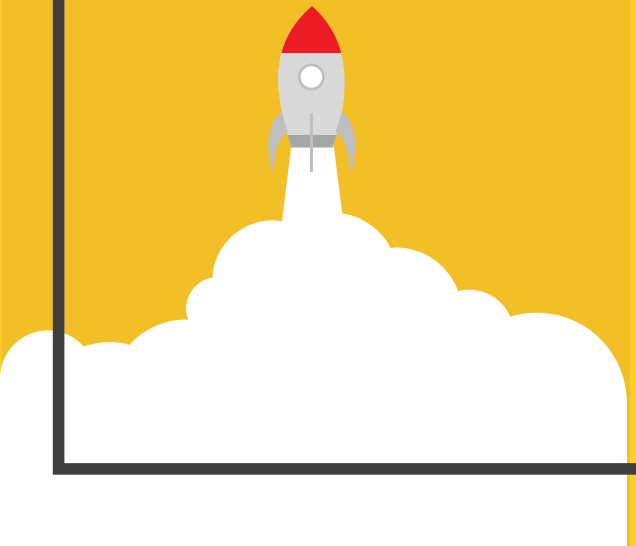


http é “stateless”

O servidor não mantém informação sobre os pedidos passados pelos clientes

Camada de Aplicação

Protocolo HTTP



http é “stateless”

O servidor não mantém informação sobre os pedidos passados pelos clientes

Protocolos que mantém informações de estado são complexos!

- necessidade de organizar informações passadas
- se ocorrer um crash as informações podem ser perdidas ou gerar inconsistências entre o cliente e o servidor

Usuário entra com a URL: http://luisrodrigoog.github.io/aulas/redes.html

1a. **cliente http** inicia conexão TCP ao servidor **http** (processo) em **luisrodrigoog.github.io** porta **80** é a default para o servidor **http** .

1b. **servidor http** no host **luisrodrigoog.github.io** esperando pela conexão TCP na porta 80. "**aceita**" conexão, notificando o cliente

2. **cliente http** envia http *request message* (contendo a URL) para o socket da conexão TCP

3. **servidor http** recebe mensagem de pedido, forma *response message* contendo o objeto solicitado (**aulas/redes.html**), envia mensagem para o socket

tempo

Usuário entra com a URL: http://luisrodrigoog.github.io/aulas/redes.html

4. **servidor http** fecha conexão TCP.

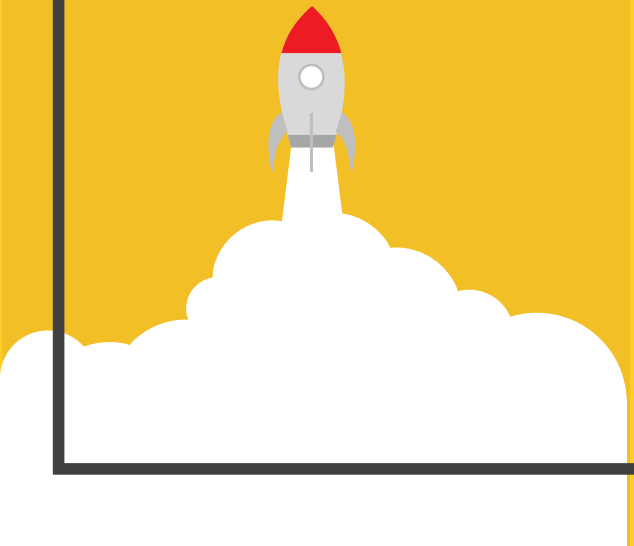
5. **cliente http** recebe mensagem de resposta contendo o **arquivo html**, apresenta o **conteúdo html**. **Analisando** o arquivo html encontra vários objetos jpeg referenciados

6. **Passos 1-5** são repetidos para cada um dos vários objetos jpeg.

tempo

Camada de Aplicação

Protocolo HTTP - Conexões

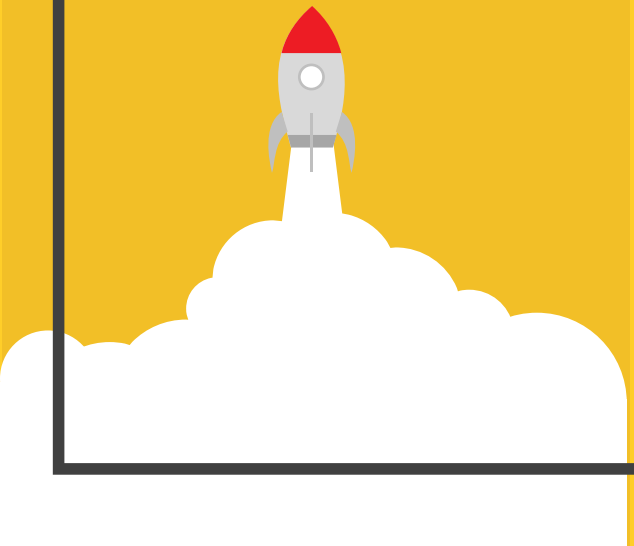


Não-persistente

- **http/1.0:** servidor analisa pedido, envia resposta e fecha a conexão TCP
- **2 RTTs** para obter um objeto
  - Conexão TCP
  - Solicitação e transferência do objeto
- Cada transferência sofre por causa do mecanismo de **slow-start** do TCP
- Muitos **browser** abrem várias **conexões paralelas**

Camada de Aplicação

Protocolo HTTP - Conexões



Persistente

- Modo default para **http/1.1**
- Na **mesma conexão** TCP são trazidos **vários objetos**
- O **cliente** envia **pedido para todos os objetos** referenciados tão logo ele recebe a página HTML básica .
- **Poucos** RTTs, menos **slow start.**

Camada de Aplicação

HTTP - Formato das Mensagens

Tipos de mensagens HTTP

- *request, response*
- http request message:
- ASCII (formato legível para humanos)

linha de pedido (comandos GET, POST,HEAD)

linhas de cabeçalho

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

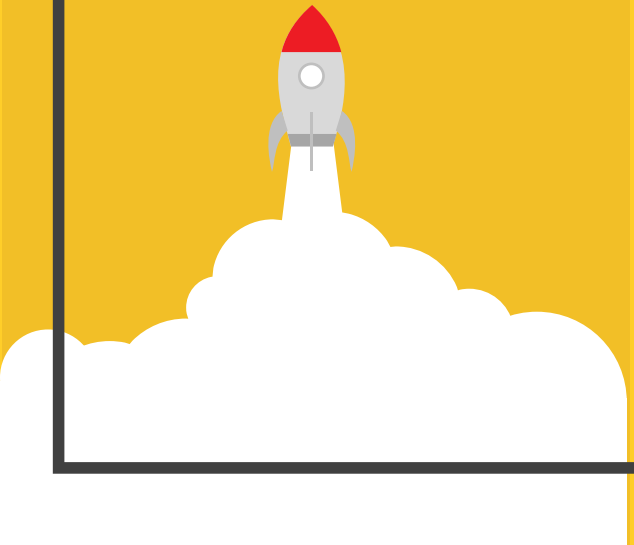
(extra carriage return, line feed)

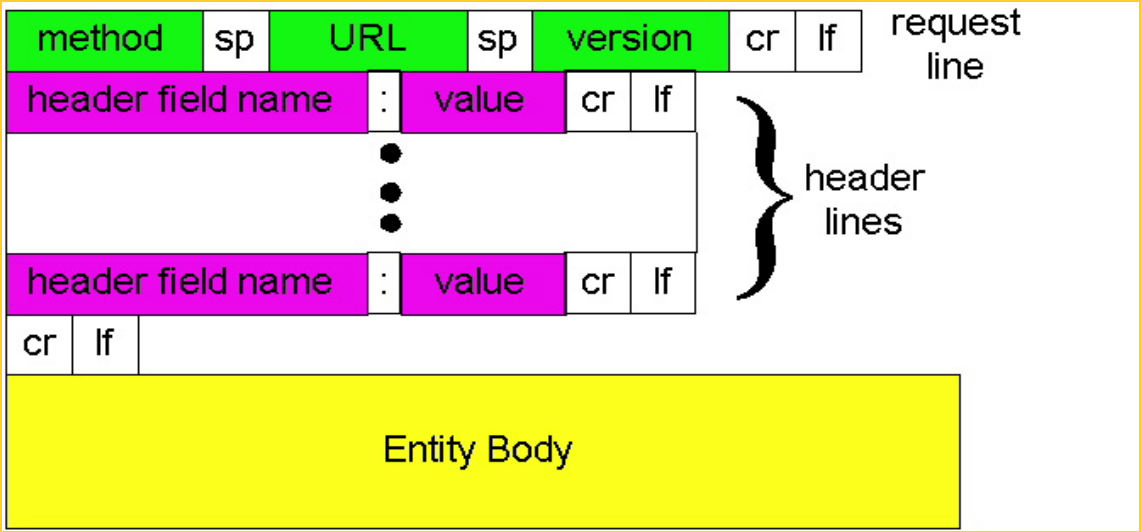
Carriage return, line feed indica fim da mensagem

Camada de Aplicação

HTTP - Formato das Mensagens

A mensagem "HTTP request"





Camada de Aplicação

HTTP - Formato das Mensagens

A mensagem "HTTP response"

linha de status (protocolo código de status frase de status)

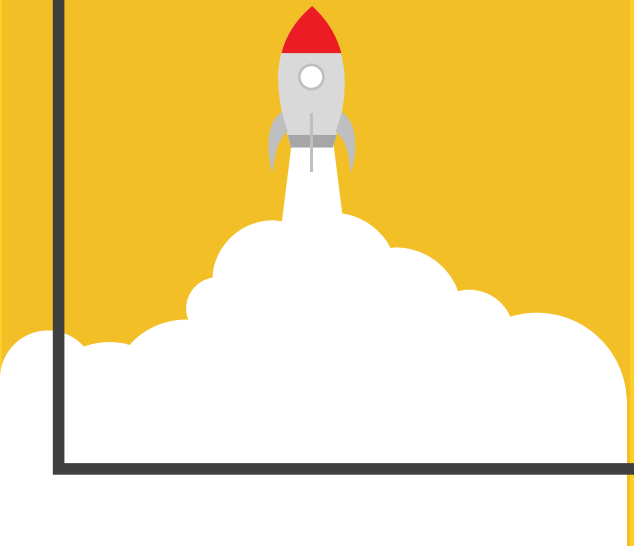
linhas de cabeçalho

dados, e.x., arquivo html

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

Camada de Aplicação

HTTP - Formato das Mensagens



Códigos de status das respostas :

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (Location:)

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Camada de Aplicação

HTTP Cliente

1. Telnet para um servidor Web:

```
telnet lrodrigo.sgs.lncc.br 80
```

Abre conexão TCP para a porta 80
(porta default do servidor http) em `lrodrigo.sgs.lncc.br` .
Qualquer coisa digitada é enviada
para a porta 80 em `lrodrigo.sgs.lncc.br`

2. Digite um pedido GET http:

```
GET /wp/cursos/ HTTP/1.0
```

Digitando isto (tecle carriage return **duas** vezes), você envia este pedido HTTP GET mínimo (mas completo) ao servidor http

3. Examine a mensagem de resposta enviada pelo servidor HTTP

Faça você mesmo !!!!

Camada de Aplicação

HTTP - Cookies

Gerados e lembrados pelo **servidor**, usados mais tarde para:

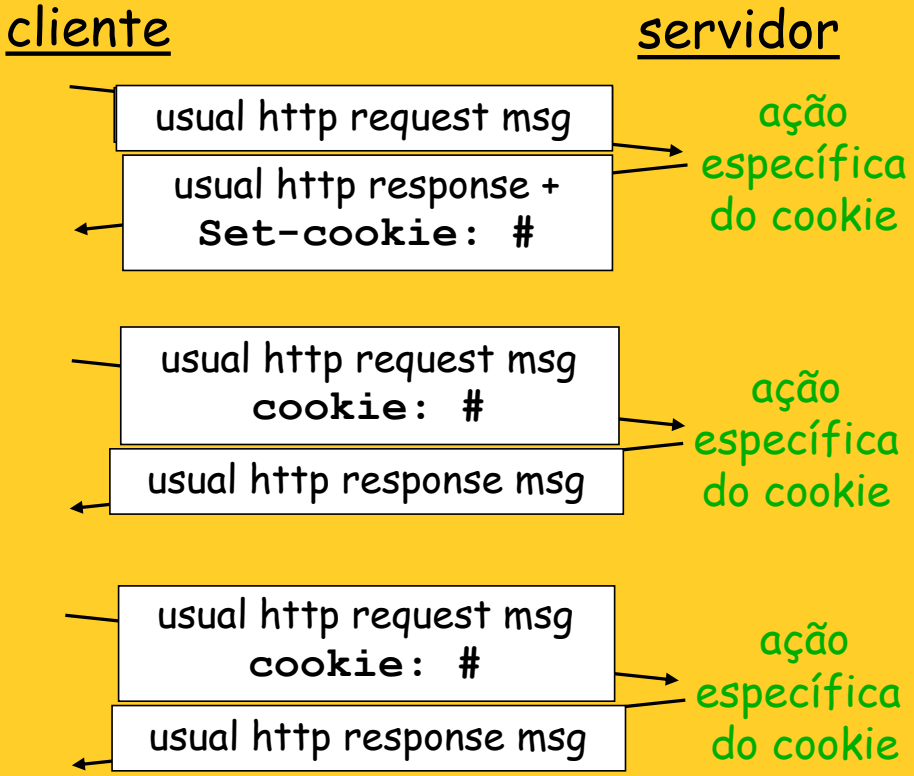
- **autenticação**
- **lembrar preferencias** dos usuários ou prévias escolhas

Servidor envia "cookie" ao cliente na resposta HTTP

Set-cookie: 1678453

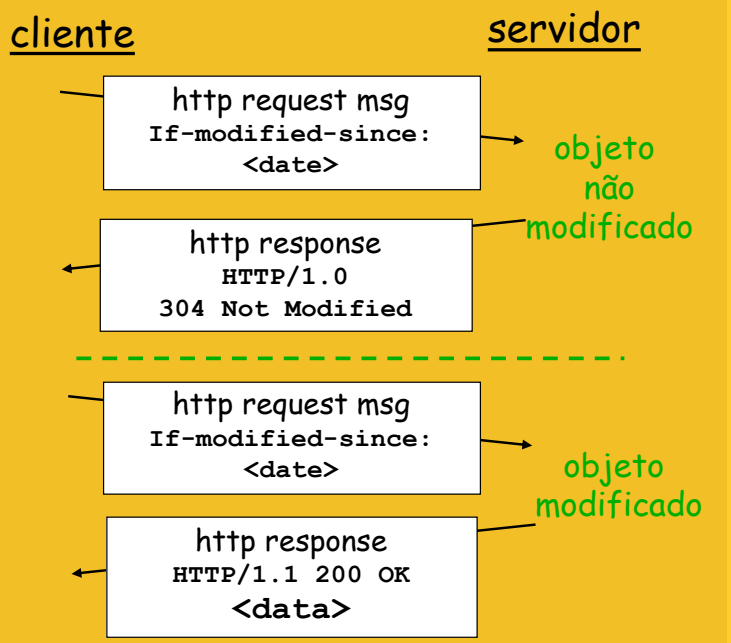
Cliente apresenta o cookie em pedidos posteriores

cookie: 1678453



Camada de Aplicação

Condicional GET (via Cliente)



Razão: não enviar objeto se a versão que o cliente já possui está atualizada.

- **cliente**: especifica data da versão armazenada no pedido HTTP

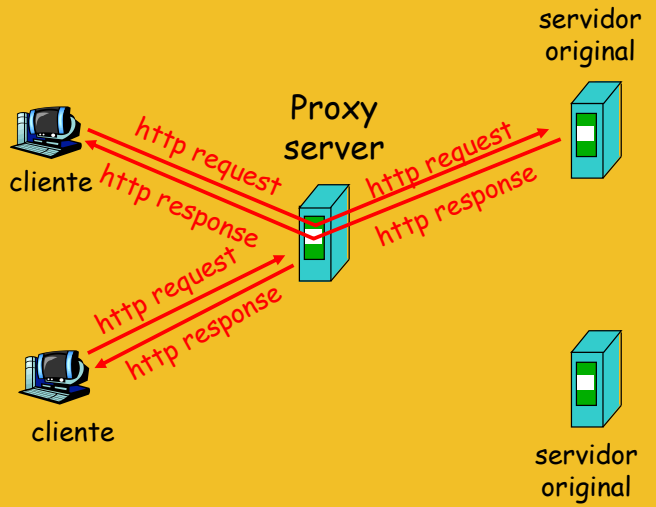
  If-modified-since: <date>

- **servidor:** resposta não contém objeto se a cópia é atualizada:

  HTTP/1.0 304 Not Modified

Camada de Aplicação

Web Caches – Proxy Server

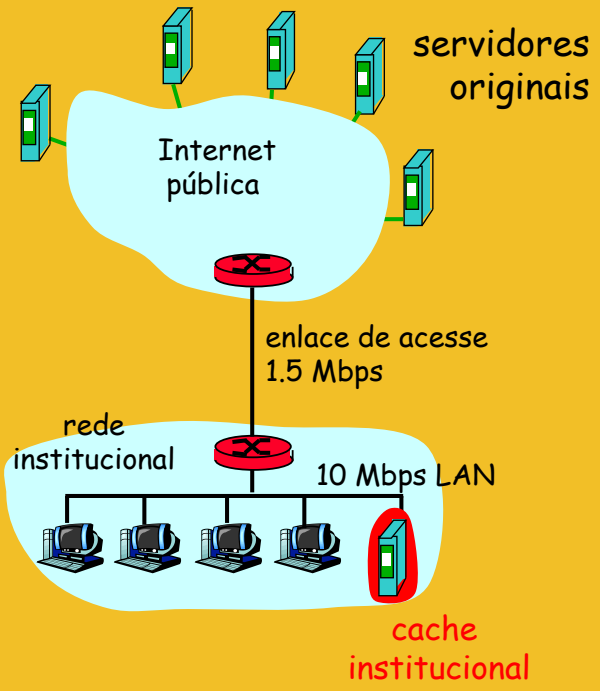


Objetivo: atender o cliente sem envolver o Servidor Web originador da informação

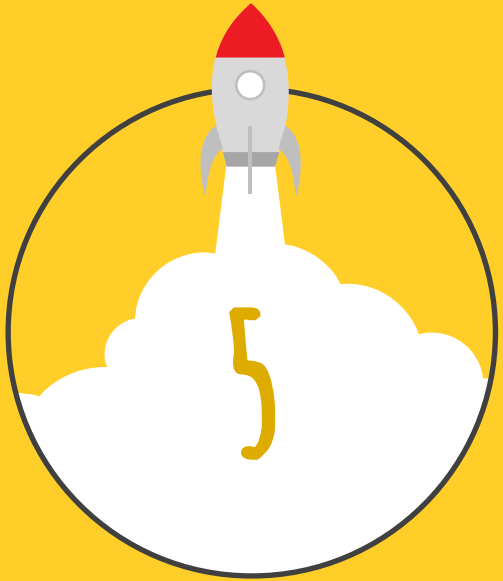
- **Usuário configura o browser**: acesso Web é feito através de um **proxy**
- Cliente envia todos os pedidos **http** para o **web cache**
  - se o objeto existe no **web cache** ele o retorna
  - ou o web cache solicita objeto do servidor original, então o envia ao cliente.

Camada de Aplicação

Web Caches – Proxy Server



- Armazenamento está "**perto**" do cliente (na mesma rede)
- Menor **tempo de resposta**
- **Reduz o tráfego** para servidor distante
  - links externos podem ser caros e facilmente congestionáveis



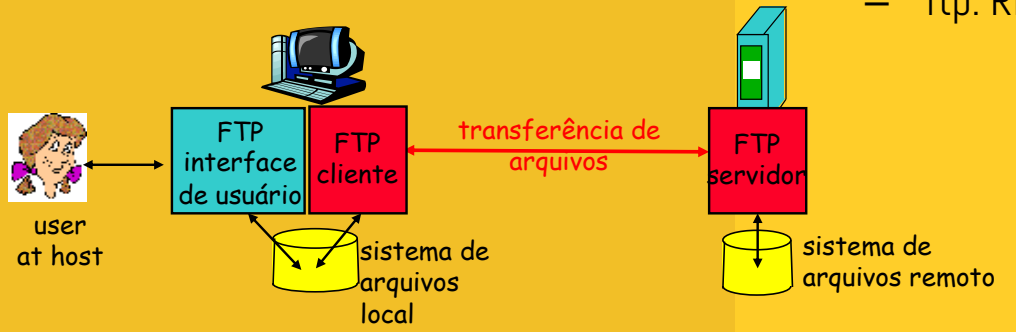
Camada de Aplicação:

Protocolo FTP

Camada de Aplicação

FTP – Transferência de Arquivos

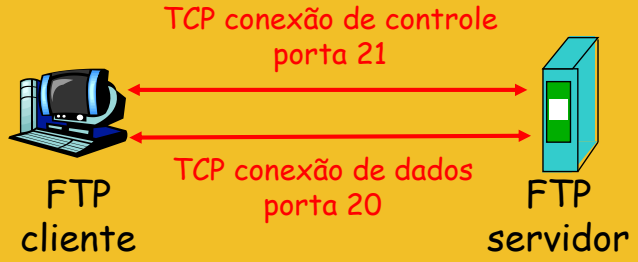
- Transferência de arquivos de e para o computador remoto
- Modelo cliente servidor
  - *cliente:* lado que inicia a transferência
  - *servidor:* host remoto
  - ftp servidor: porta 21
- Padronização
  - ftp: RFC 959



Camada de Aplicação

FTP – Transferência de Arquivos

Controle separado da conexão de Dados



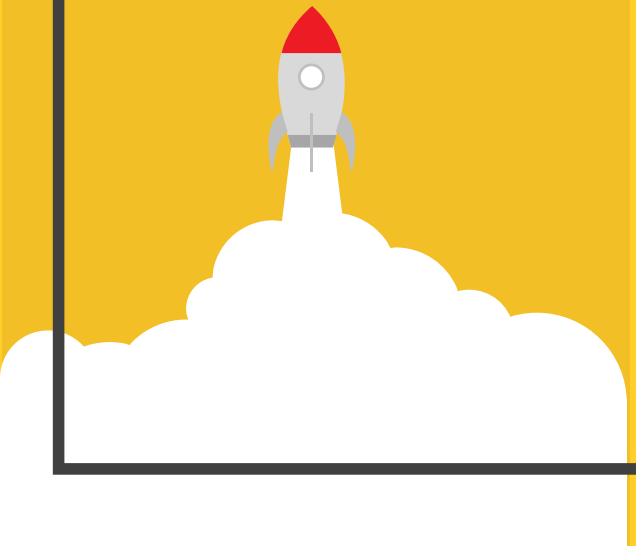
- **Cliente** FTP contata o servidor na porta **21**, especificando **TCP** como protocolo de transporte
- **Duas conexões** TCP paralelas são abertas:
  - **controle**: troca de comandos e respostas entre cliente e servidor.

    "controle out of band"
  - **dados**: dados do arquivo trocados com o servidor
- **Servidor** mantém o "**estado**", ou seja, diretório corrente, autenticação anterior

Camada de Aplicação

FTP – Transferência de Arquivos

Comandos e Repostas



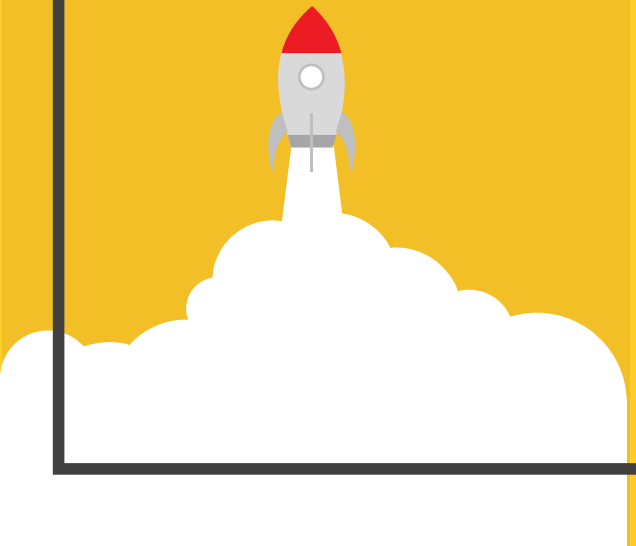
Exemplos de comandos

- Conectando-se
  - USER *username*
  - PASS *password*
- LIST retorna listagem do arquivo no diretório atual
- RETR filename recupera (obtém) o arquivo
- STOR filename armazena o arquivo no host remoto

Camada de Aplicação

FTP – Transferência de Arquivos

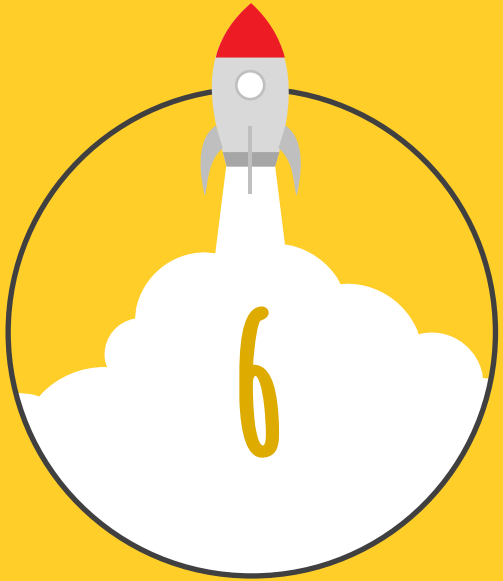
Comandos e Repostas



Exemplos de códigos de retorno

Código de status e frase (como no http)

- `331 Username OK, password required`
- `125 data connection already open; transfer starting`
- `425 Can’t open data connection`
- `452 Error writing file`

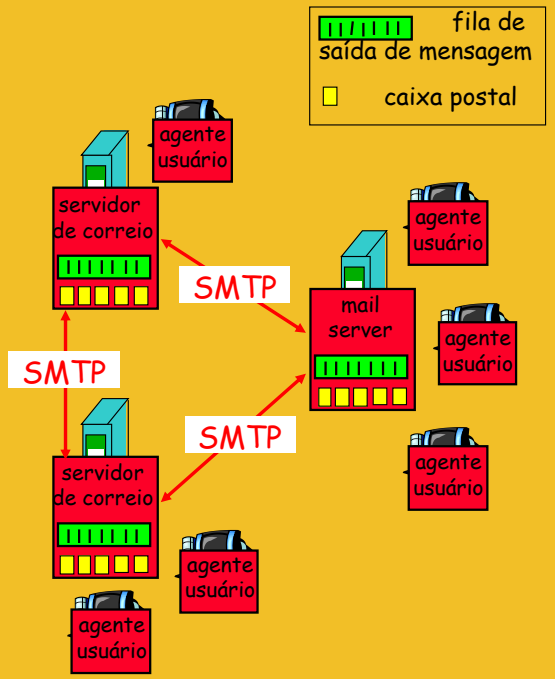


Camada de Aplicação:

Correio Eletrônico

Camada de Aplicação

Correio Eletrônico

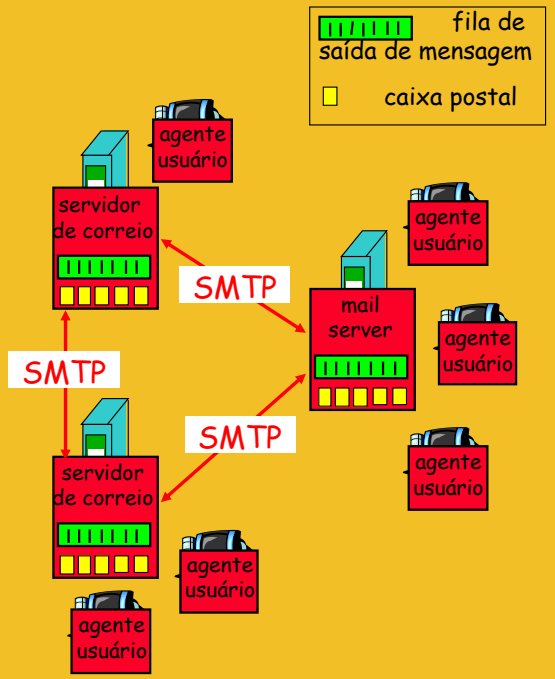


Três componentes principais

- Agentes de usuário
- Servidores de correio
- Simple Mail Transfer Protocol: **SMTP**

Camada de Aplicação

Correio Eletrônico



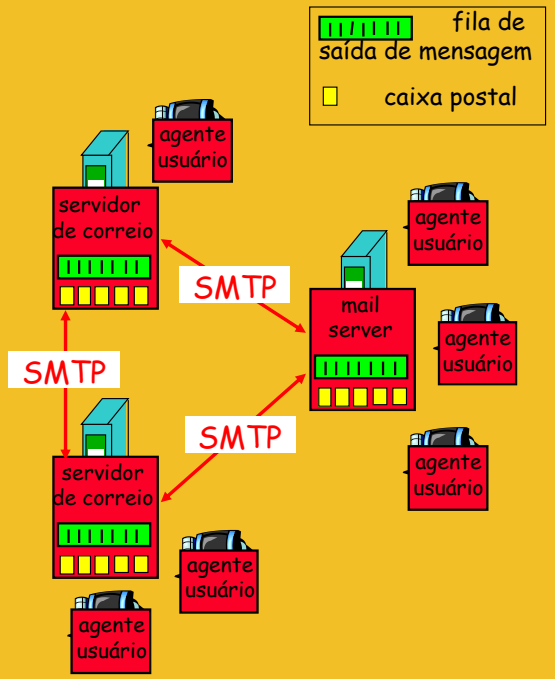
Três componentes principais

Agente de usuário

- "leitor de correio"
- Composição, edição, leitura de mensagens de correio
  - **Eudora, Outlook, elm, Thunderbird**
- Mensagens de entrada e de saída são armazenadas no servidor

Camada de Aplicação

Correio Eletrônico



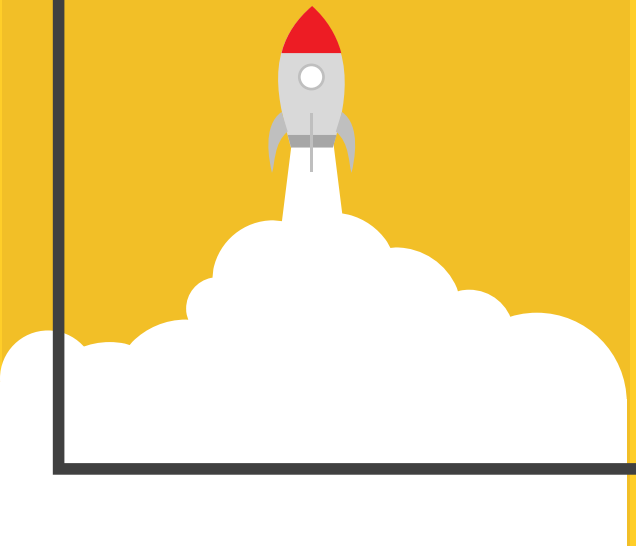
Servidores de Correio

- **caixa postal** contém mensagens que chegaram (ainda não lidas) para o usuário
- **fila de mensagens** contém as mensagens de correio a serem enviadas
- **protocolo smtp** permite aos servidores de correio trocarem mensagens entre eles
  - **cliente**: servidor de correio que envia
  - "**servidor**": servidor de correio que recebe

Camada de Aplicação

Correio Eletrônico

SMTP – RFC:821



- Usa TCP para transferência de mensagens do cliente ao servidor, porta 25
- Transferência direta: servidor que envia para o servidor que recebe
- **Três fases** de transferência
  - handshaking (apresentação)
  - transferência de mensagens
  - fechamento
- Interação comando/resposta
  - comandos: texto ASCII
  - resposta: código de status e frase
- Mensagens devem ser formatadas em código **ASCII** de **7 bits**

Exemplo de interação SMTP

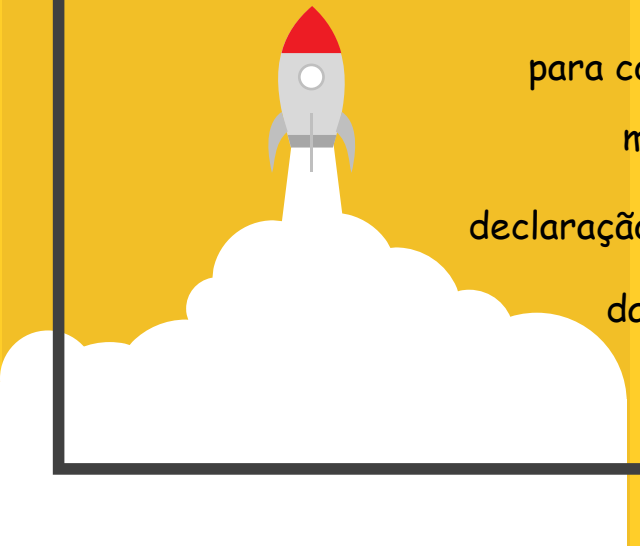
```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Camada de Aplicação

Correio Eletrônico

Formato das Mensagens

Extensão Multimídia



MIME: multimídia mail extension

- RFC 2045, 2056
- Linhas adicionais no cabeçalho declaram o tipo de conteúdo MIME

MIME versão

método usado para codificar dados

multimedia data tipo, subtipo, declaração de parâmetro

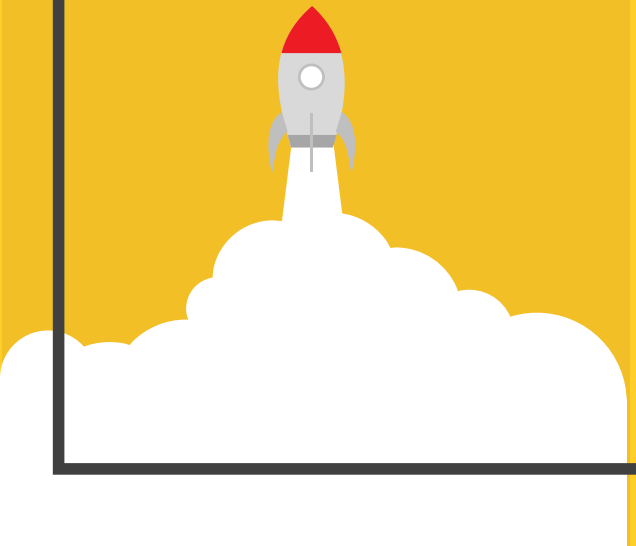
dados codificados

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Camada de Aplicação

Correio Eletrônico

Tipos MIME



`Content-Type: type/subtype; parâmetros`

Text

- Exemplo: **plain, html**

Image

- Exemplo: **jpeg, gif**

Áudio

- Exemplo: **basic** (codificado 8-bit m-law), **32kadpcm** (codificação 32 kbps)

Vídeo

- Exemplo: **mpeg, quicktime**

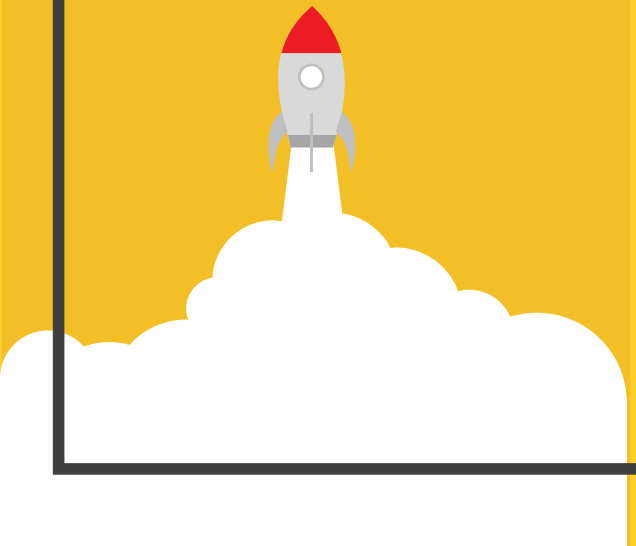
Application

- Outros dados que devem ser processados pelo leitor antes de serem apresentados “visualmente”
- Exemplo: **msword, octet-stream**

Camada de Aplicação

Correio Eletrônico

Tipo Multiparte

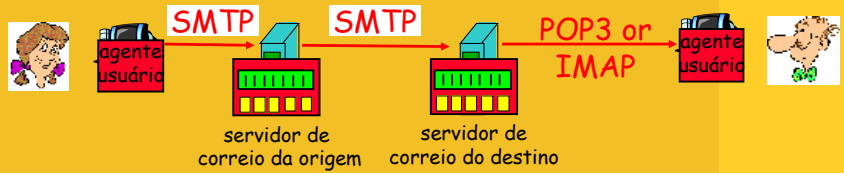


```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Camada de Aplicação

Correio Eletrônico

Protocolos de Acesso



Protocolo de acesso: recupera mensagens do servidor

- **POP**: **Post Office Protocol** [RFC 1939]
  - autorização (agente <-->servidor) e download
- **IMAP**: **Internet Mail Access Protocol** [RFC 1730]
  - maiores recursos (mais complexo)
  - manipulação de mensagens armazenadas no servidor
- **HTTP**: Hotmail, Yahoo! Mail, etc.

Protocolo POP3

Fase de autorização

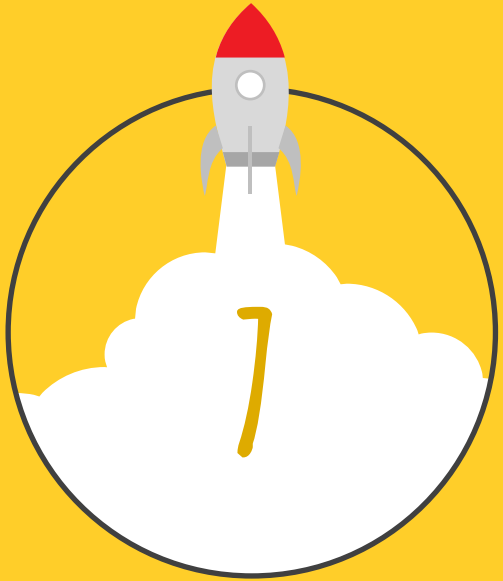
- Comandos do cliente:
  - **user:** declara nome do usuário
  - **pass:** senha/password
- Respostas do servidor
  - **+OK**
  - **-ERR**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Fase de transação, cliente:

- **list:** lista mensagens e tamanhos
- **retr:** recupera mensagem pelo número
- **dele:** apaga
- **quit**

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

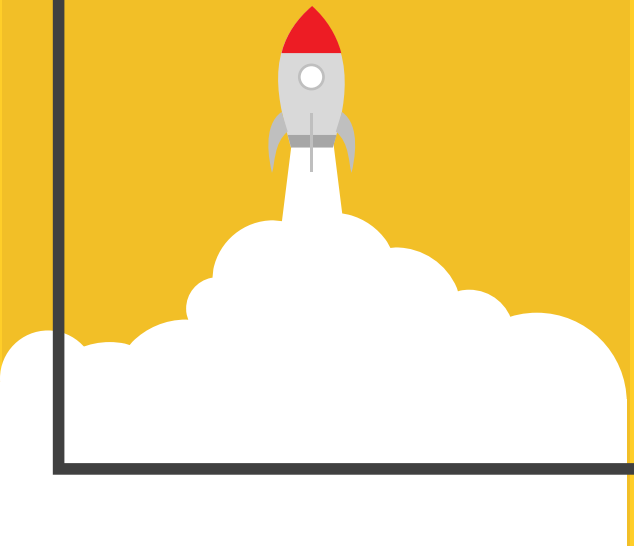


Camada de Aplicação:

DNS – Domain Name System

Camada de Aplicação

DNS: Domain Name System



Pessoas: muitos identificadores:

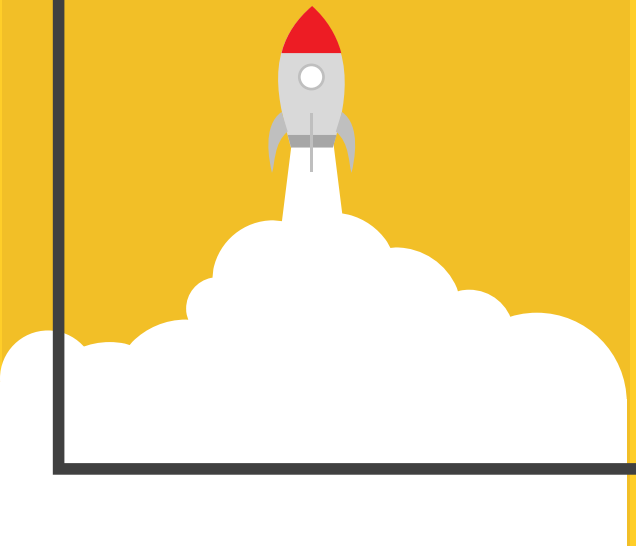
- RG, Nome, Passaporte

Internet hosts, roteadores:

- End. IP (32 bit) - usados para endereçar datagramas
- FQDN (lrodrigo.sgs.lncc.br) usados por humanos

Camada de Aplicação

DNS: Domain Name System



Pessoas: muitos identificadores:

- RG, Nome, Passaporte

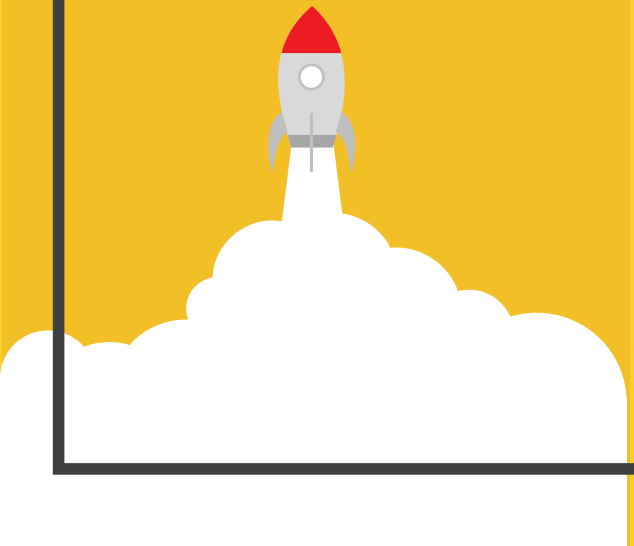
Internet hosts, roteadores:

- End. IP (32 bit) - usados para endereçar datagramas
- FQDN (lrodrigo.sgs.lncc.br) usados por humanos

Como relacionar nomes com endereços IP?

Camada de Aplicação

DNS: Domain Name System



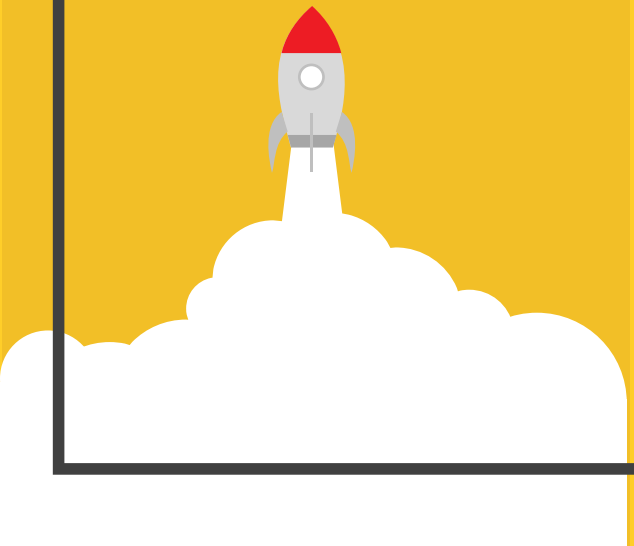
Domain Name System

- Base de dados distribuída implementada numa hierarquia de muitos *servidores de nomes*
- Protocolo de camada de aplicação host, roteadores se comunicam com servidores de nomes para *resolver* nomes (translação nome/endereço)
  - **nota**: função interna da Internet, implementada como protocolo da camada de aplicação
  - complexidade na "**borda**" da rede

Camada de Aplicação

DNS: Domain Name System

Servidores de Nome



Porque não centralizar o DNS?

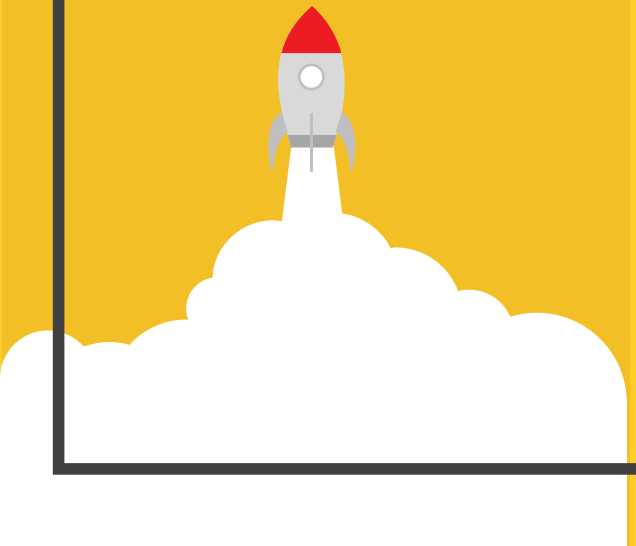
- Ponto único de falha
- Volume de tráfego
- Base de dados distante
- Manutenção

Não cresce junto com a rede!

Camada de Aplicação

DNS: Domain Name System

Servidores de Nome



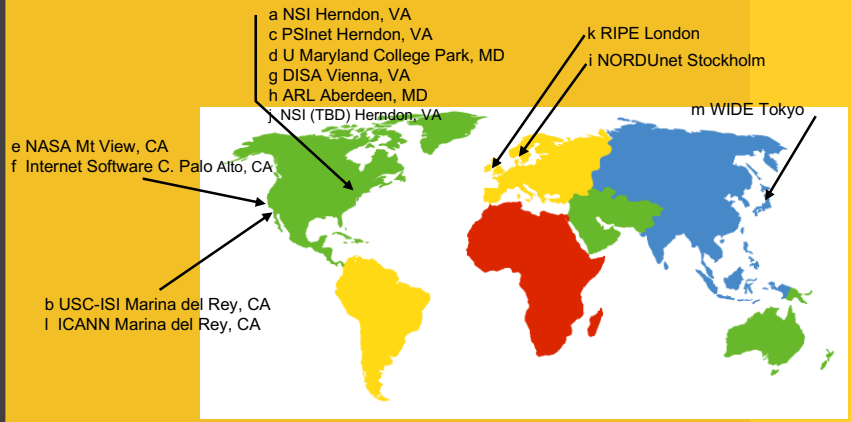
Nenhum servidor tem todos os mapeamentos de nomes para endereços IP

- **Servidores de nomes locais**
  - Cada ISP ou empresa tem um *servidor de nomes local (default)*
  - Consultas dos computadores locais ao DNS vão primeiro para o servidor de nomes local
- **Servidor de nomes autoritativo**
  - para um computador: armazena o nome e o endereço IP daquele computador
  - pode realizar mapeamentos de nomes para endereços para aquele nome de computador

Camada de Aplicação

DNS: Domain Name System

Servidores de Nome Raiz



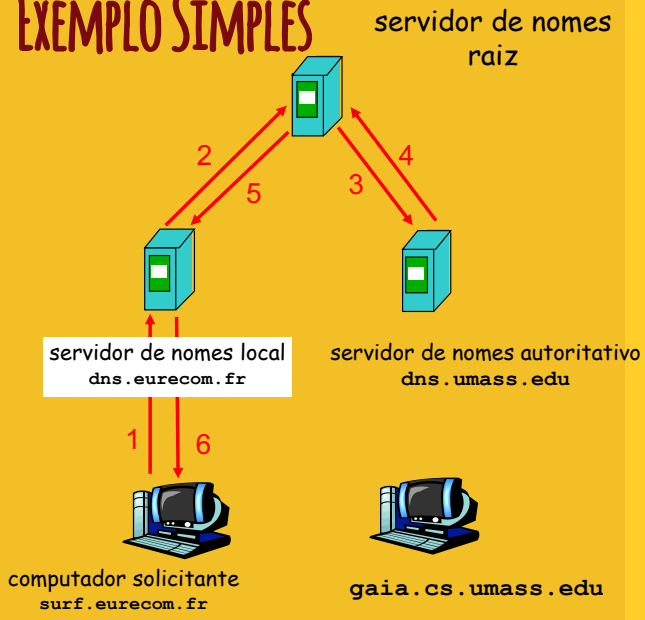
- São contatados pelos servidores de nomes locais que não podem resolver um nome
- Servidores de nomes raiz::
  - Buscam servidores de nomes **autoritativos** se o mapeamento do nome não for conhecido
  - Conseguem o **mapeamento**
  - **Retornam** o mapeamento para o servidor de nomes local

Existem 13 servidores de nomes raiz no mundo

Camada de Aplicação

DNS: Domain Name System

Exemplo Simples

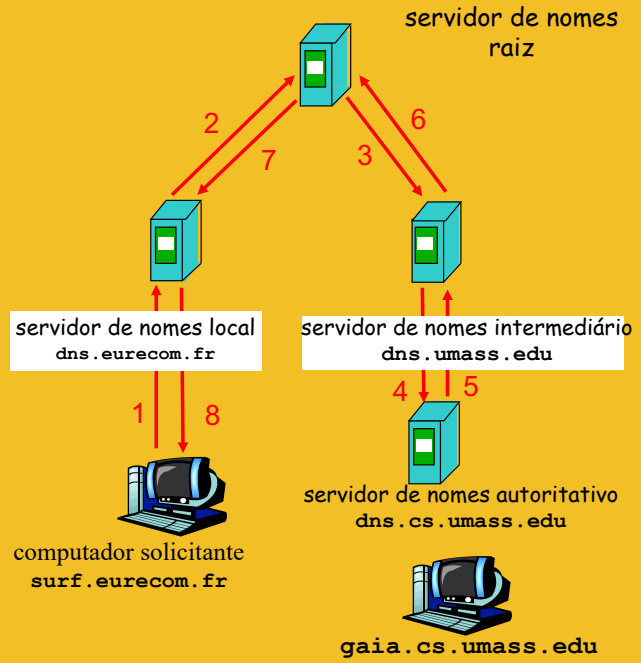


O Host **surf.eurecom.fr** quer o endereço IP de **gaia.cs.umass.edu**

1. contata seu servidor DNS local, **dns.eurecom.fr**
2. **dns.eurecom.fr** contata o servidor de nomes raiz se necessário
3. o servidor de nomes raiz contata o servidor de nomes **autoritativo**, **dns.umass.edu,** se necessário

Camada de Aplicação

DNS: Domain Name System - Exemplos

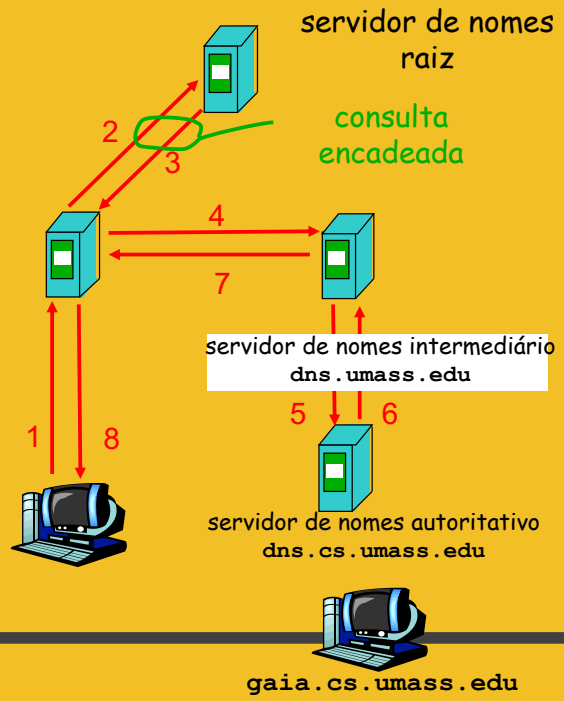


Servidor de nomes raiz

- Pode não conhecer o servidor de nomes **autoritativo** para um certo nome
- Pode conhecer: *servidor de nomes intermediário* aquele que deve ser contatado para encontrar o servidor de nomes **autoritativo**

Camada de Aplicação

DNS: Domain Name System - Consultas



Consulta recursiva:

- Transfere a tarefa de resolução do nome para o servidor de nomes consultado
- Carga pesada?

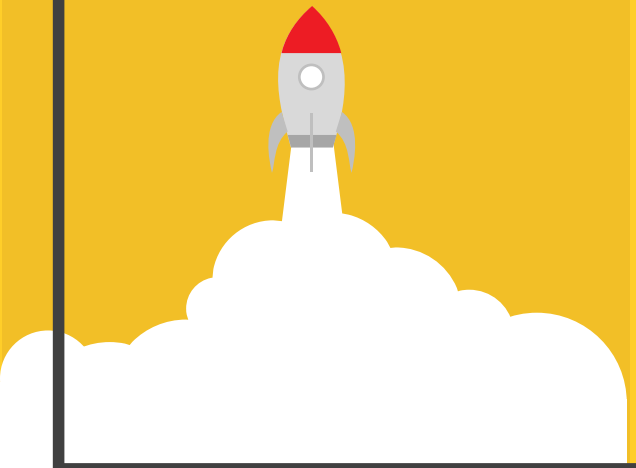
Consulta encadeada:

- Servidor **contatado** responde com o nome de outro servidor de nomes para contato
- "Eu não sei isto ,mas pergunte a este servidor"

Camada de Aplicação

DNS: Domain Name System

Armazenamento e Atualizando Registro



Ama vez que um servidor de nomes apreende um mapeamento, ele armazena o mapeamento num registro ro tipo *cache*

- registro do cache tornam-se obsoletos (desaparecem) depois de um certo tempo

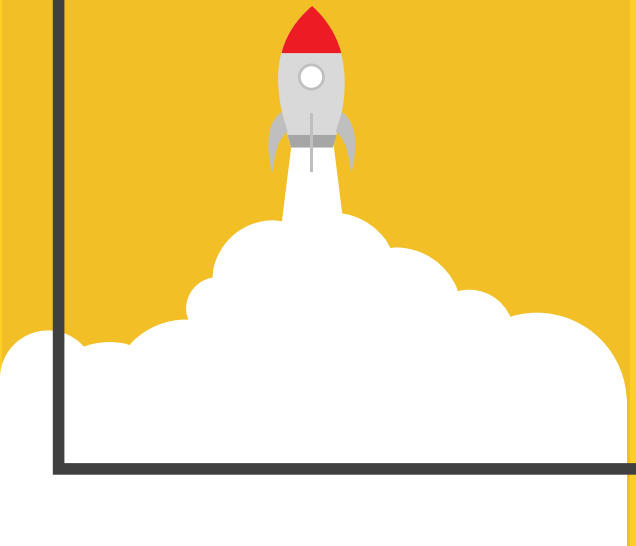
Mecanismos de atualização e notificação estão sendo projetados pelo IETF

- **RFC 2136**
- http://www.ietf.org/html.charters/dnsind-charter.html

Camada de Aplicação

DNS: Domain Name System

Registros do DNS



DNS: Registros de recursos (RR)

formato dos RR: `(name, value, type,ttl)`

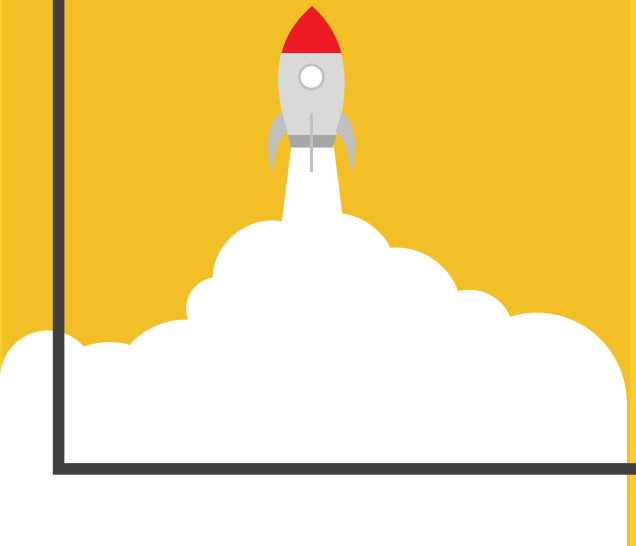
Type=A

- **name** é o nome do computador
- **value** é o endereço IP

Camada de Aplicação

DNS: Domain Name System

Registros do DNS



Type=NS

- **name** é um domínio (ex. foo.com)
- **value** é o endereço IP do servidor de nomes autoritativo para este domínio

Type=CNAME

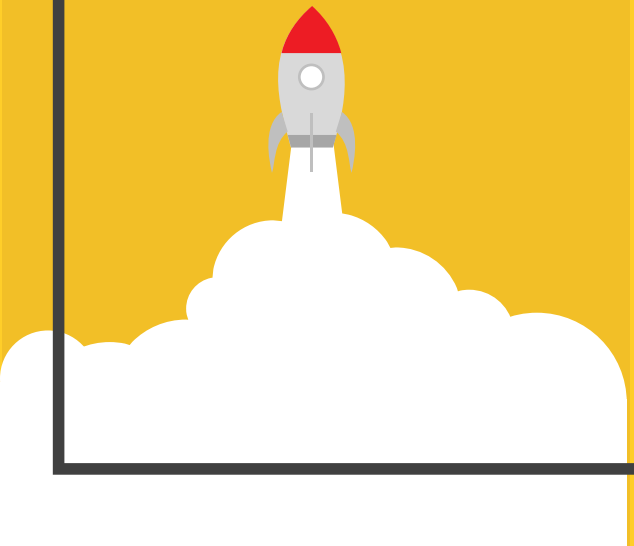
- **name** é um "apelido" para algum nome "canônico" (o nome real)
- **value** é o nome canônico

www.ibm.com é realmente servereast.backup2.ibm.com

Camada de Aplicação

DNS: Domain Name System

Registros do DNS



Type=MX

- **value** é o nome do servidor de correio associado com **name**

Camada de Aplicação

DNS: Domain Name System

Protocolo e Mensagens

Protocolo DNS

- mensagem de *consulta* e *resposta*
- ambas com o mesmo *formato de mensagem*

cabeçalho da msg

- identificação
  - número de 16 bit para consulta,
  - resposta usa o mesmo número
- flags:
  - consulta ou resposta
  - recursão desejada
  - recursão disponível
  - resposta é autoritativa

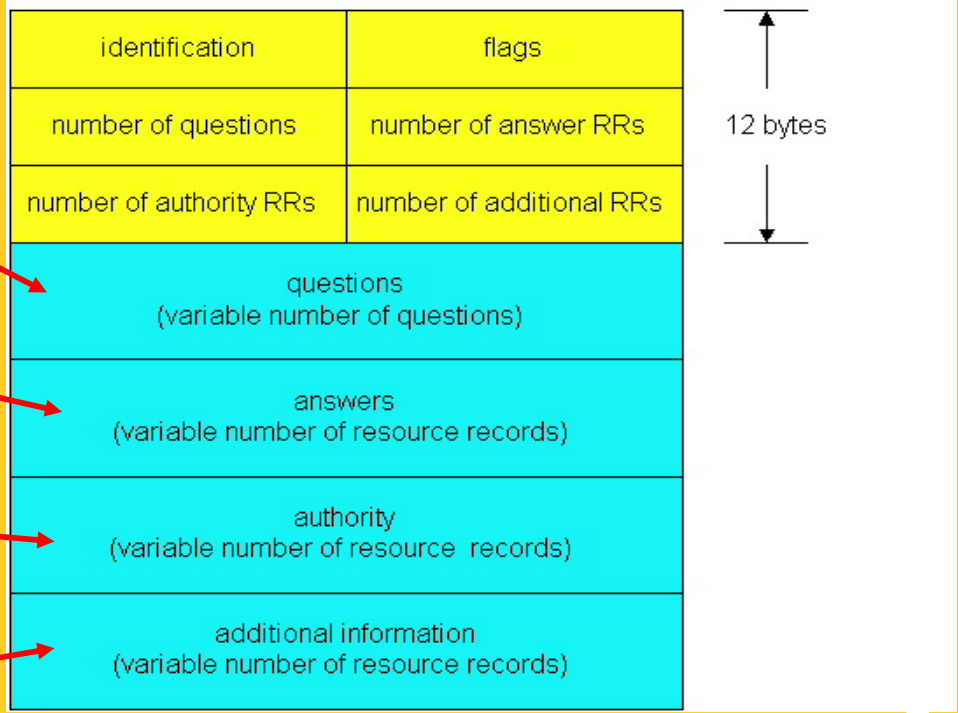
DNS: Protocolo e Mensagens

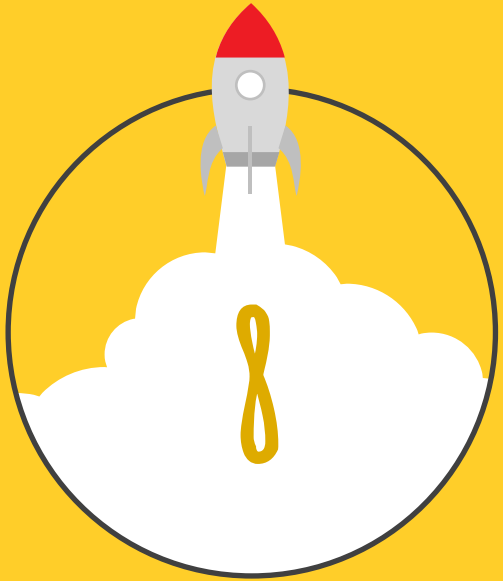
Campos de nome e tipo para uma consulta

RRs de resposta a uma consulta

registros para servidores autoritativos

informação adicional que pode ser útil





Camada de Aplicação:

Sockets

Camada de Aplicação

Programação de Sockets

Socket API

- Introduzida no BSD4.1 UNIX, 1981
- Explicitamente criados, usados e liberados pelas aplicações
- Paradigma cliente/servidor
- Dois tipos de serviço de transporte via socket API:
  - datagrama não confiável
  - confiável, orientado a cadeias de bytes

Camada de Aplicação

Programação de Sockets com TCP

Socket:

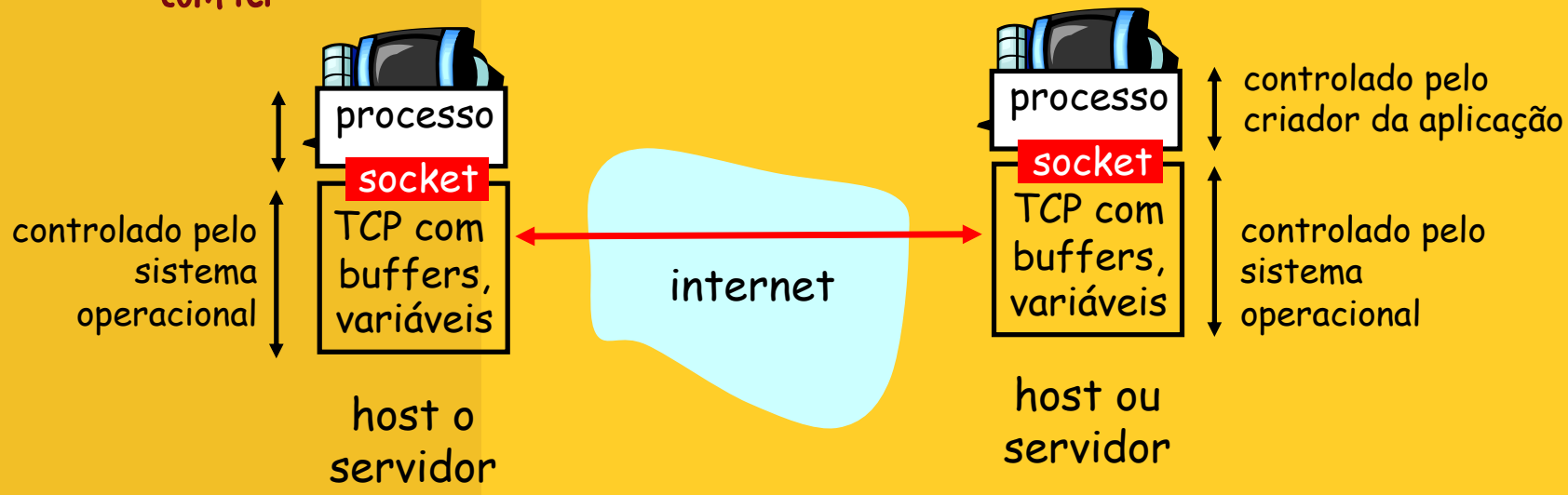
- uma porta entre o processo de aplicação e o protocolo de transporte fim-a-fim (UCP or TCP)

Serviço TCP:

- transferência confiável de bytes de um processo para outro

Camada de Aplicação

Programação de Sockets com TCP



Camada de Aplicação

Programação de Sockets com TCP

Cliente deve contatar o servidor

- Processo servidor já deve estar executando antes de ser contatado
- Servidor deve ter criado socket (porta) que aceita o contato do cliente

Cliente contata o servidor através de:

- Criando um socket TCP local
- especificando endereço IP e número da porta do processo servidor

Camada de Aplicação

Programação de Sockets com TCP

- Quando o cliente cria o socket ele estabelece conexão TCP com o servidor
- Quando contatado pelo cliente, o servidor cria um novo socket para comunicar-se com o cliente
  - permite que o servidor converse com múltiplos clientes

Camada de Aplicação

Programação de Sockets com TCP

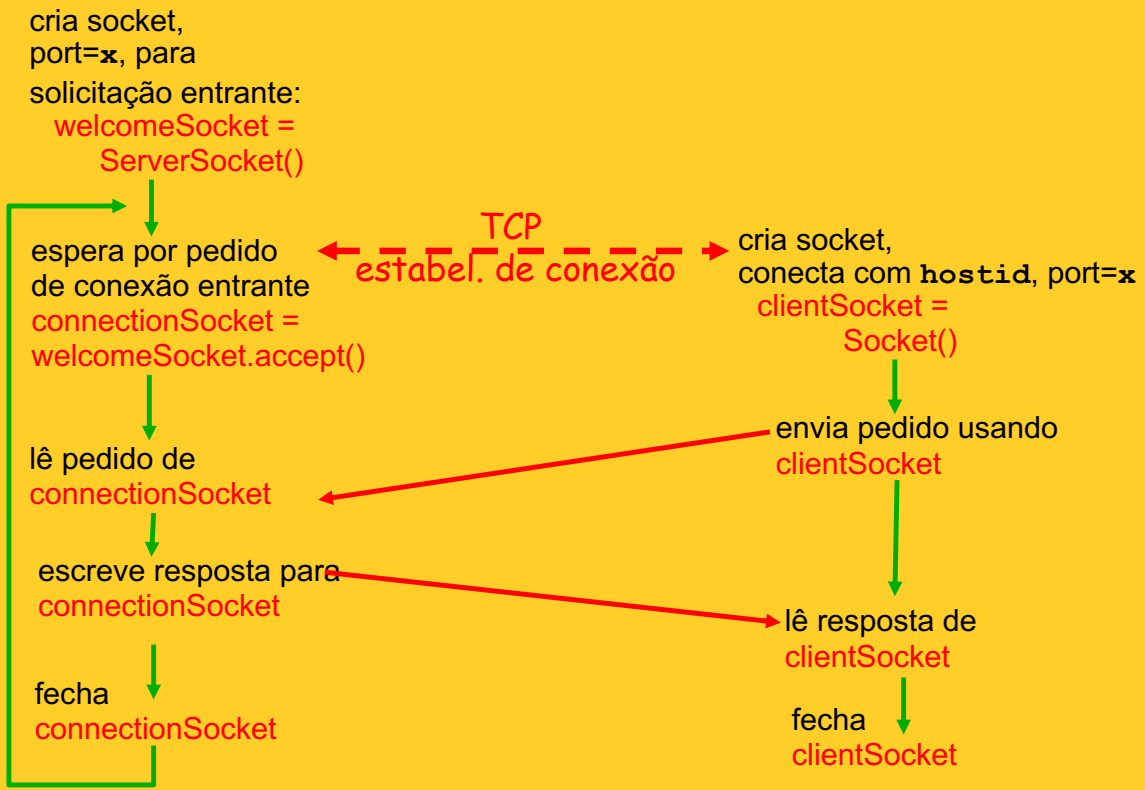
ponto de vista da aplicação

TCP fornece a transferência confiável, em ordem de bytes ("pipe") entre o cliente e o servidor

Camada de Aplicação

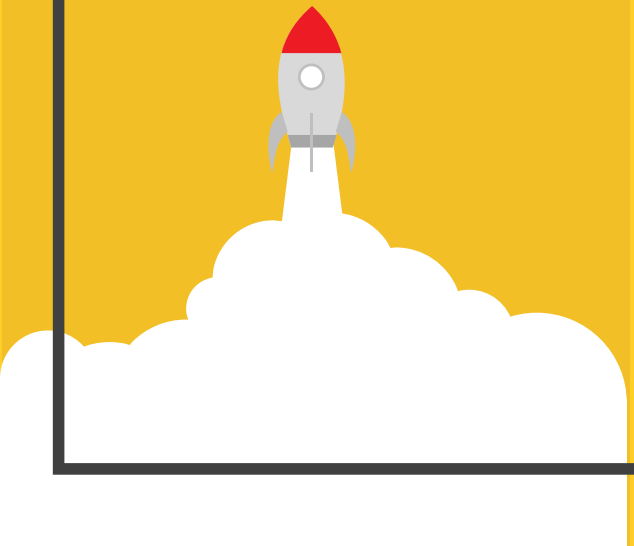
Programação de Sockets - TCP

Interação Cliente/Servidor



Camada de Aplicação

Programação de Sockets - UDP

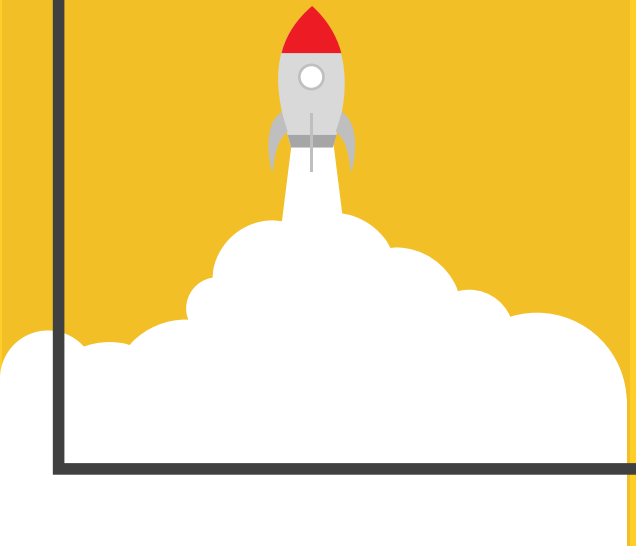


UDP: não há conexão entre o cliente e o servidor

- Não existe apresentação
- Transmissor envia explicitamente endereço IP e porta de destino em cada mensagem
- Servidor deve extrair o endereço IP e porta do transmissor de cada datagrama recebido
- Os dados transmitidos podem ser recebidos foram de ordem ou perdidos

Camada de Aplicação

Programação de Sockets - UDP



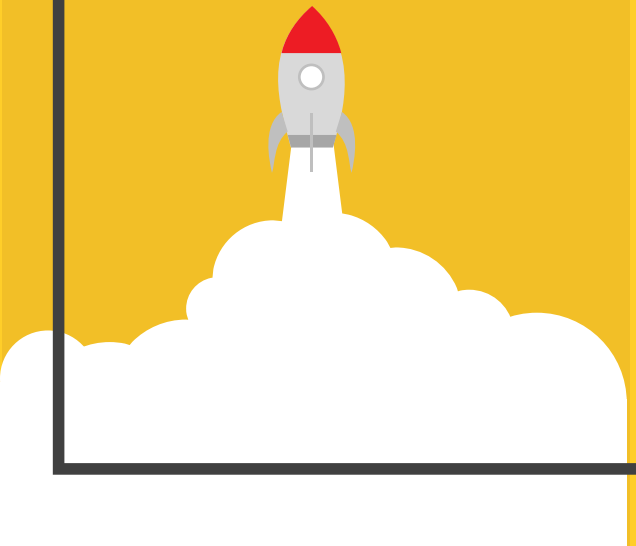
ponto de vista da aplicação

UDP fornece a transferência não confiável de grupos de bytes ("datagramas") entre o cliente e o servidor

Camada de Aplicação

Programação de Sockets – UDP

Interação Cliente/Servidor



Luis Rodrigo – luis.goncalves@ucp.br

104192 - Redes de Computadores

Aula 5 – Camada de Aplicação

Prof. Luis Rodrigo – luis.goncalves@ucp.br