UNIVERSIDADE DE
VASSOURAS

UNIVERSIDADE DE VASSOURAS

Curso de Graduação em Engenharia de Software

Aula 3

Redes de Computadores

Prof. André Saraiva
Mestre em Sistemas de Computação
Especialista em Arquitetura e Projeto de Cloud Computing
Analista Sênior Blue Team em Cibersegurança pela Kimoshiro
Tutor EaD pela Universidade Federal Fluminense - UFF

Bibliografia

KUROSE, James F; ROSS, Keith W. **Redes de computadores e a Internet: uma abordagem top-down** . 5. ed. São Paulo (SP): Pearson e Addison Wesley, 2010

Filosofias e Princípios da Internet

- **Administração descentralizada**:
  - A Internet é, em princípio, um esforço coletivo.
  - Múltiplas entidades **colaboram** para um objetivo maior.
    - Interconexão entre elementos computacionais.

- Mas estas entidades possuem metodologias, objetivos individuais diferentes.
  - Algumas vezes conflitantes.

- Necessidade de **uniformização**.
  - Estabelecimento de princípios básicos seguidos por todos.

Filosofias e Princípios da Internet

- Por projeto, a Internet é **heterogênea**.
  - Ex.: enlaces de tecnologias, capacidades diferentes.
  - Ex.: ISPs com políticas diferentes.
  - Ex.: dispositivos computacionais com arquiteturas diferentes.

- Consequências:
  - São necessários **protocolos comuns a todos os dispositivos.**
    - Uma suíte de protocolos da Internet.
  - Estes protocolos devem ser **flexíveis**.
    - Ex.: não devem assumir muitas características da rede.

O Argumento Fim-a-Fim

- Argumento apresentado por David Clark em 1981.
  - Chefe do desenvolvimento da arquitetura de protocolos na Internet entre 1981 e 1989.
- Guiou o desenvolvimento arquitetural da Internet.
  - Embora haja exceções.

ARGUMENTO FIM-A-FIM

Uma funcionalidade só pode ser implementada de forma correta e completa se isto for feito com o auxílio das aplicações executadas nas pontas do sistema de comunicação.

O Argumento Fim-a-Fim

- Primeira abordagem: rede “garantirá” a integridade dos dados.
  - Todo roteador/host, ao receber um pacote por um enlace, verificará sua integridade.
    - **Como?**
  - Caso pacote não seja íntegro, roteador/host requisitará uma **retransmissão**.
  - A rede também garantirá que pacotes **não cheguem fora de ordem.**
- Pergunta: isto é suficiente?

O Argumento Fim-a-Fim

- **Não**, por uma série de motivos.
- Quem garante que arquivo que saiu do host estava originalmente íntegro?
- Quem garante que verificação de integridade dos pacotes não falhou?
- Quem garante que as implementações dos roteadores/hosts estão corretas?

- Em resumo: em última instância, **receptor ainda precisa verificar integridade.**

O Argumento Fim-a-Fim

- **Alternativa:** deixar que sistemas finais lidem com o problema.
  - Façam as verificações.
  - Requisitem retransmissões, se necessário.
    - A partir do host de origem!

- **Vantagem:**
  - Já que as pontas precisam da funcionalidade, esta é mantida apenas lá.
  - Não há redundância de implementações.

O Argumento Fim-a-Fim: Desempenho

- Consequência direta: na Internet, procura-se manter a complexidade nas bordas.
  - “Inteligência nas bordas”.
  - “Núcleo simples”.
- Núcleo processa volume enorme de pacotes.
  - Mantê-lo simples, especializado, **rápido**.
  - Objetivo maior: **escalabilidade**.
- Volume de tráfego nas bordas é menor.
  - Processamentos mais complexos são mais viáveis.
  - Escalabilidade não é tão problemática.
- Oposição às redes telefônicas.
  - Núcleo complexo, bordas simples.

Princípio KISS

- KISS: Keep It Simple, Stupid! (Mantenha simples, idiota!)
  - Princípio originado na Marinha Americana, na década de 1960.
  - Incorporado nas filosofias que guiam a Internet.

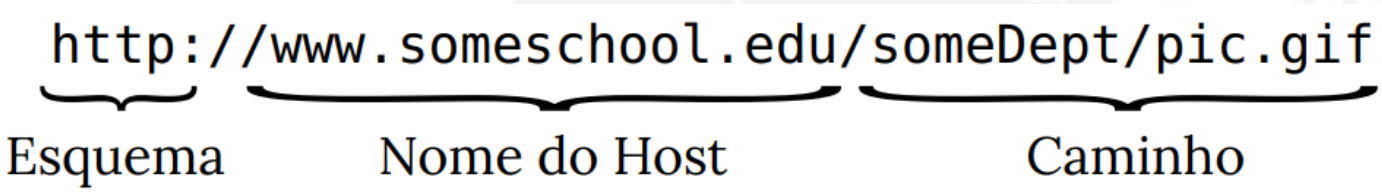
PRINCÍPIO KISS

Sistemas funcionam melhor quando são mantidos simples. Logo, simplicidade deve ser um objetivo de projeto, evitando complexidades desnecessárias.

- Aplicado à Internet:
  - Protocolos “simples”.
- Fáceis de implementar, depurar.

Web e HTTP

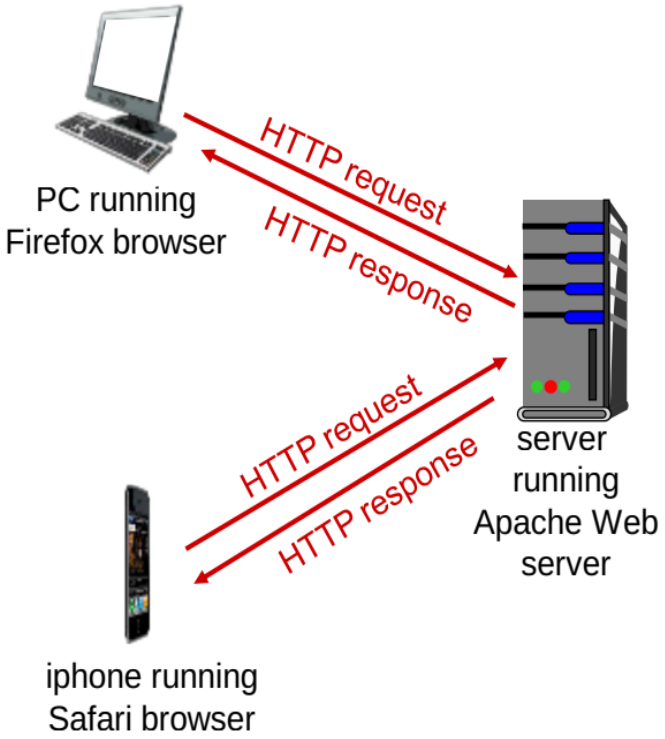
- Uma **página web** é formada por **objetos**.
  - Arquivos HTML, imagens jpg, png, vídeos, áudios, ...
- Arquivo HTML base da página faz **referência** a diversos objetos.
- Cada objeto é **endereçável através de uma URL**. Exemplo:



HTTP

HTTP: HyperText Transfer Protocol.

- Protocolo de camada de aplicação da Web.
- Arquitetura cliente — servidor.
- **Cliente:** browser que requisita, recebe (usando protocolo HTTP) e “exibe” objetos Web.
- **Servidor:** servidor Web envia (usando protocolo HTTP) objetos em resposta a requisições.



HTTP

Usa TCP.

- Cliente inicia conexão TCP (cria socket) com o servidor, porta 80.
- Servidor aceita conexão TCP do cliente.
- Mensagens HTTP (mensagens do protocolo de aplicação) trocadas entre browser (cliente HTTP) e servidor Web (servidor HTTP).
- Conexão TCP é fechada

HTTP é “stateless”.

- Servidor não mantém informação sobre requisições passadas do cliente.

Protocolos que mantêm estado são complexos!

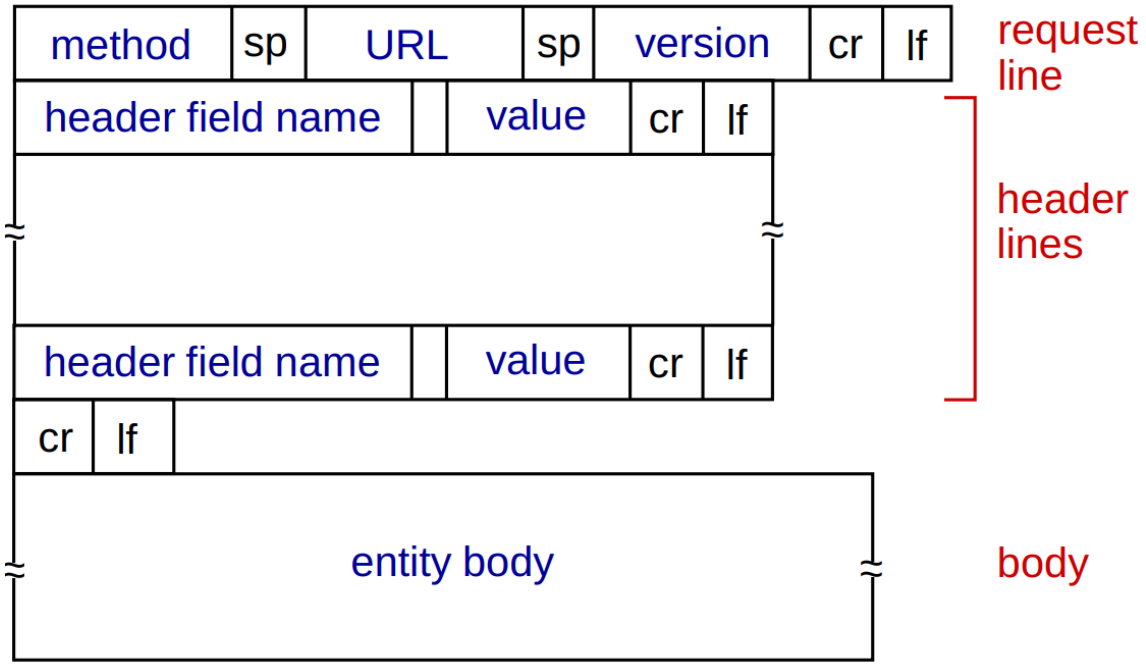
- Histórico do passado (estado) precisa ser armazenado.
- Se servidor/cliente cai, suas visões do estado podem ser inconsistentes, devem ser sincronizadas.

Requisição HTTP

- HTTP tem dois tipos de mensagem: **requisição e resposta**.
- Requisição HTTP: codificada em ASCII (formato legível por humanos)

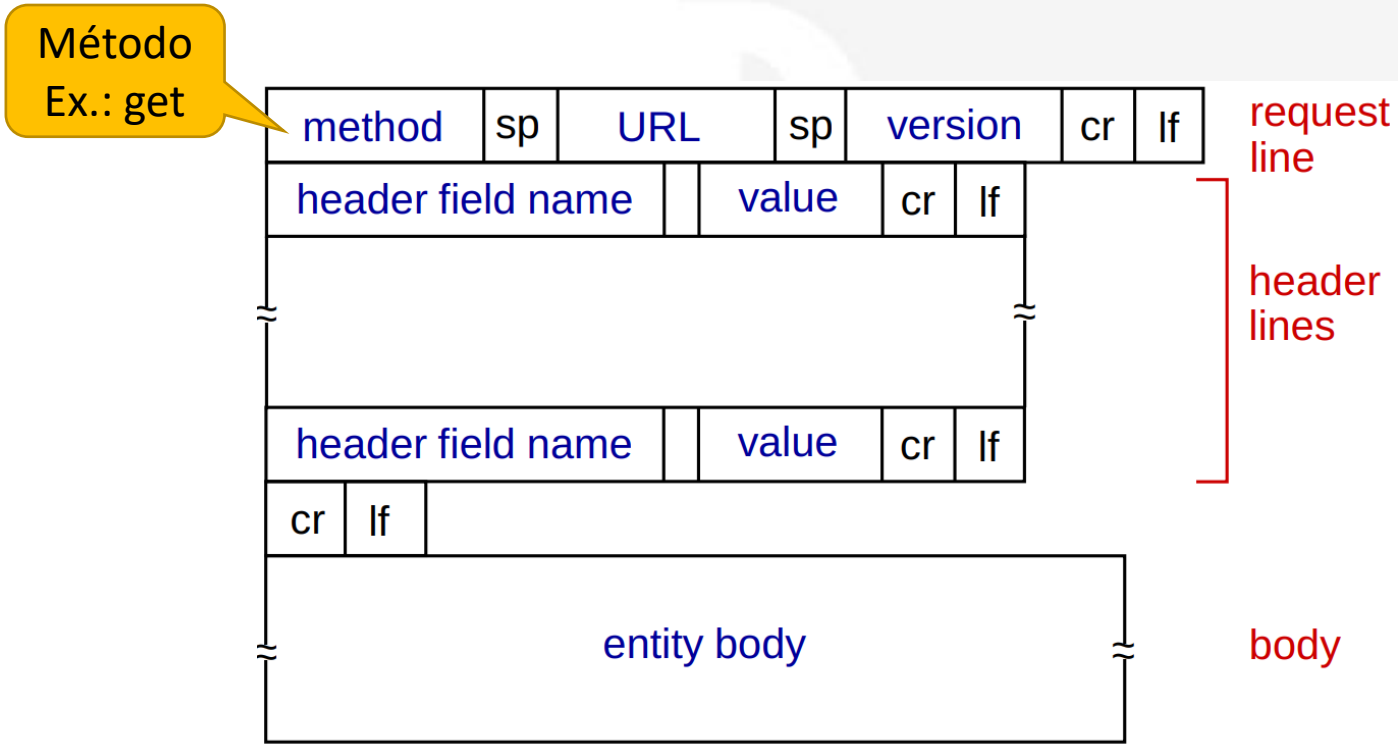
Requisição HTTP

- Formato geral da Requisição



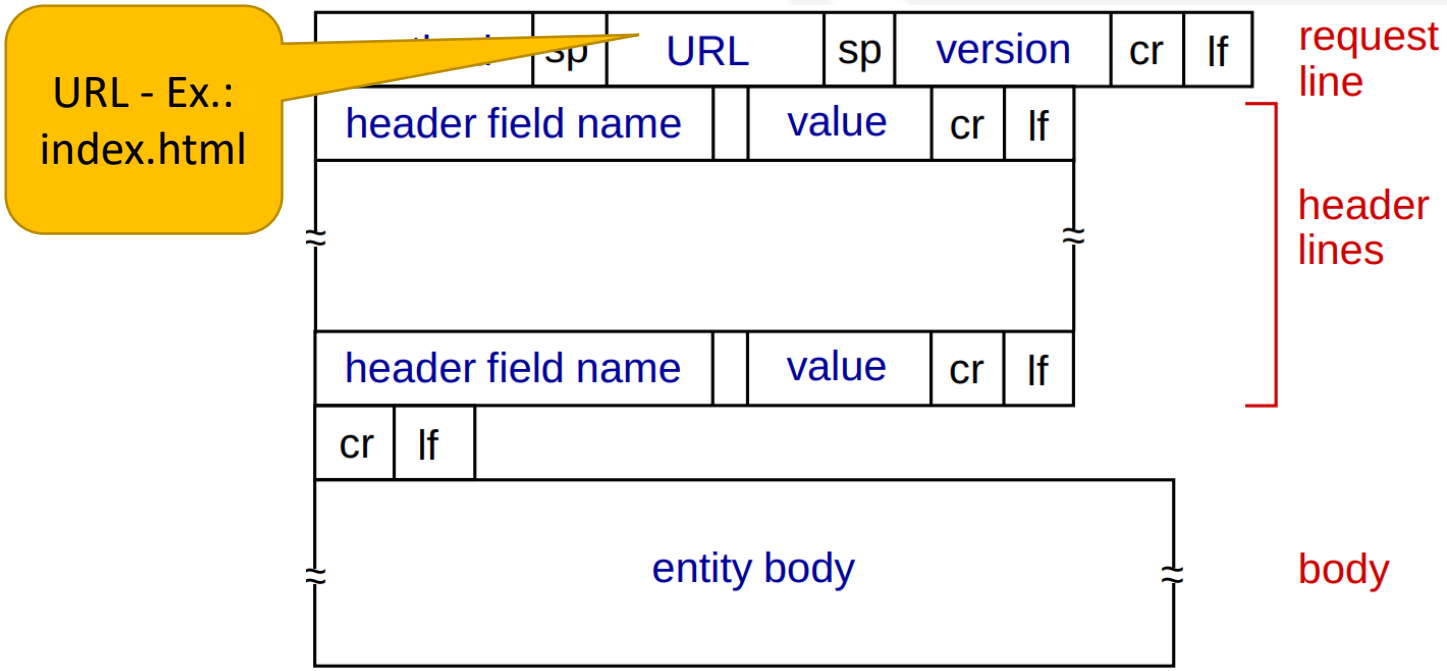
Requisição HTTP

- Formato geral da Requisição



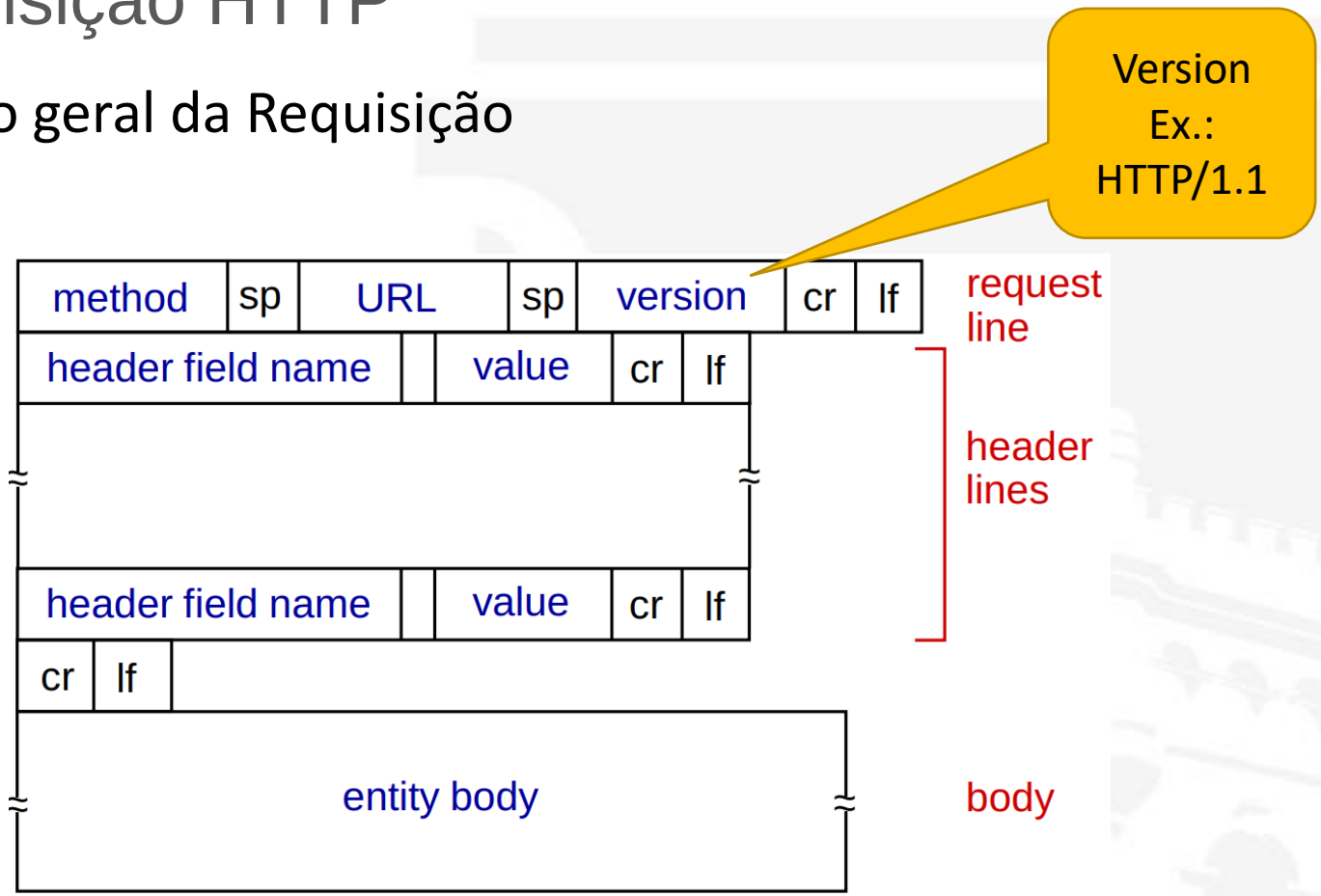
Requisição HTTP

- Formato geral da Requisição



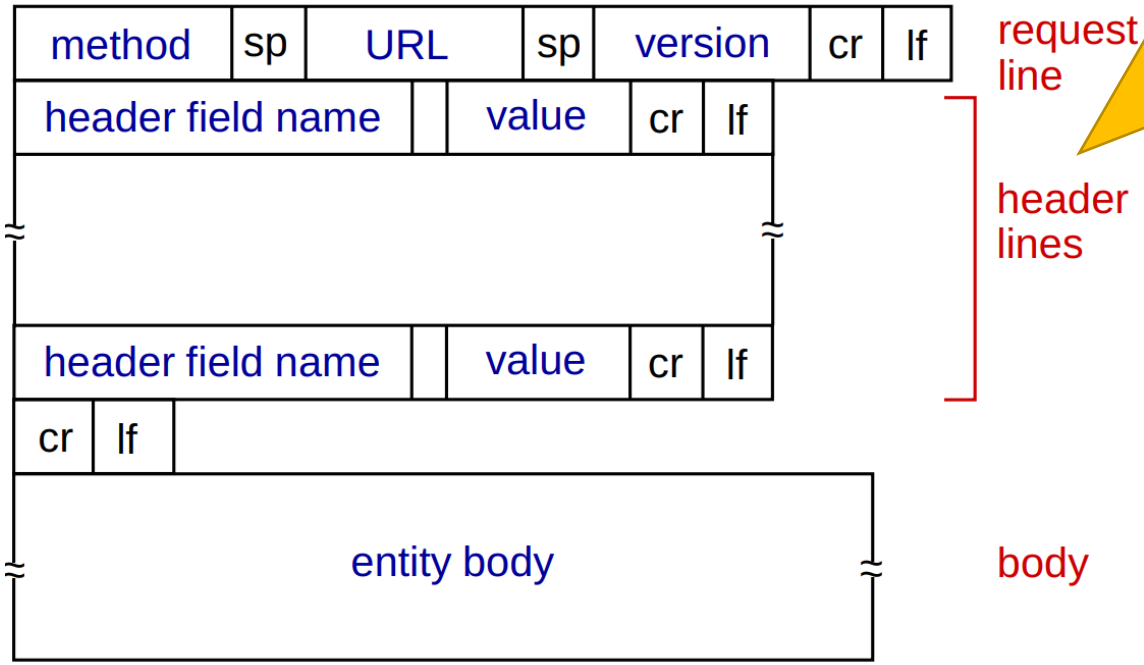
Requisição HTTP

- Formato geral da Requisição



Requisição HTTP

- Formato geral da Requisição



Tipos de Métodos HTTP

HTTP/1.0:

- GET.
- POST.
- HEAD: Pede que servidor não envia objeto na resposta

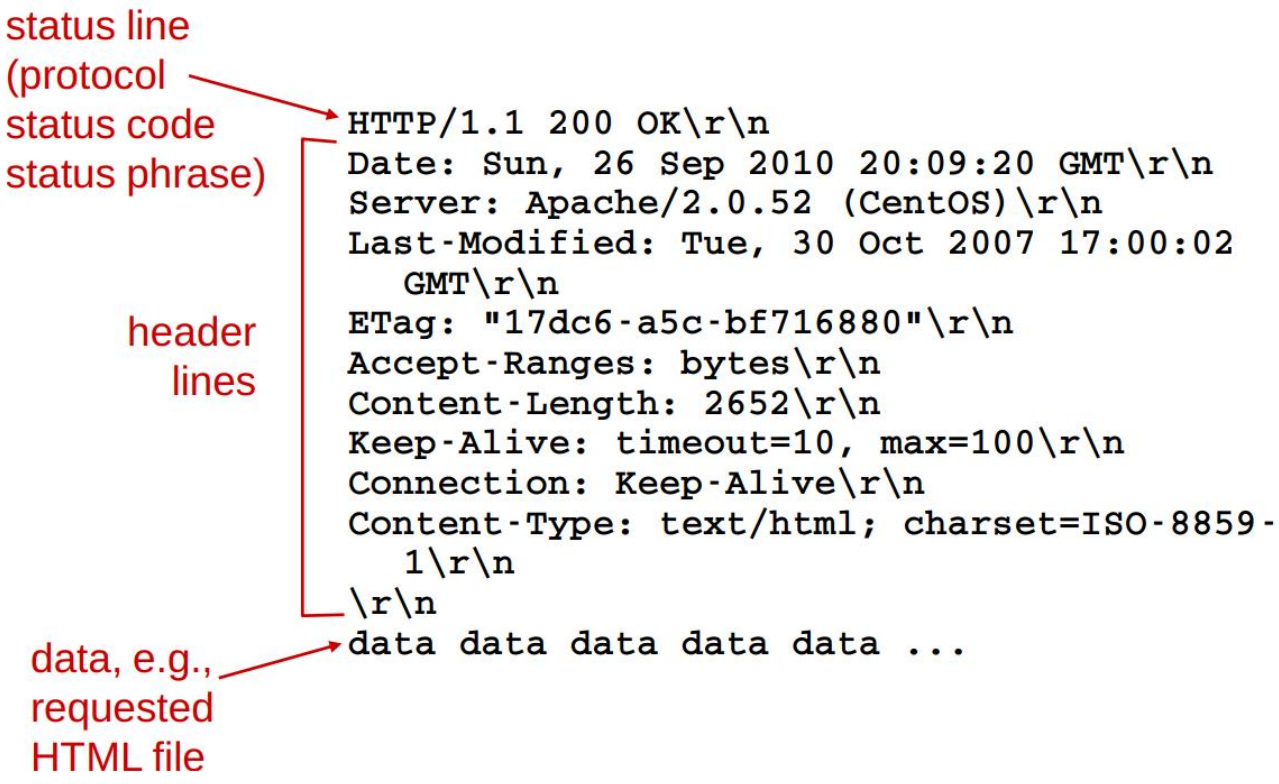
HTTP/1.1:

- GET, POST, HEAD.
- PUT: Faz upload de objeto para a URL especificada.
- DELETE: Apaga objeto da URL especificada.

Tipos de Métodos HTTP

- **Método POST**
  - Conteúdo é enviado para o servidor no corpo da mensagem
- **Método GET**
  - Conteúdo é enviado para o servidor no campo da URL

Resposta do HTTP



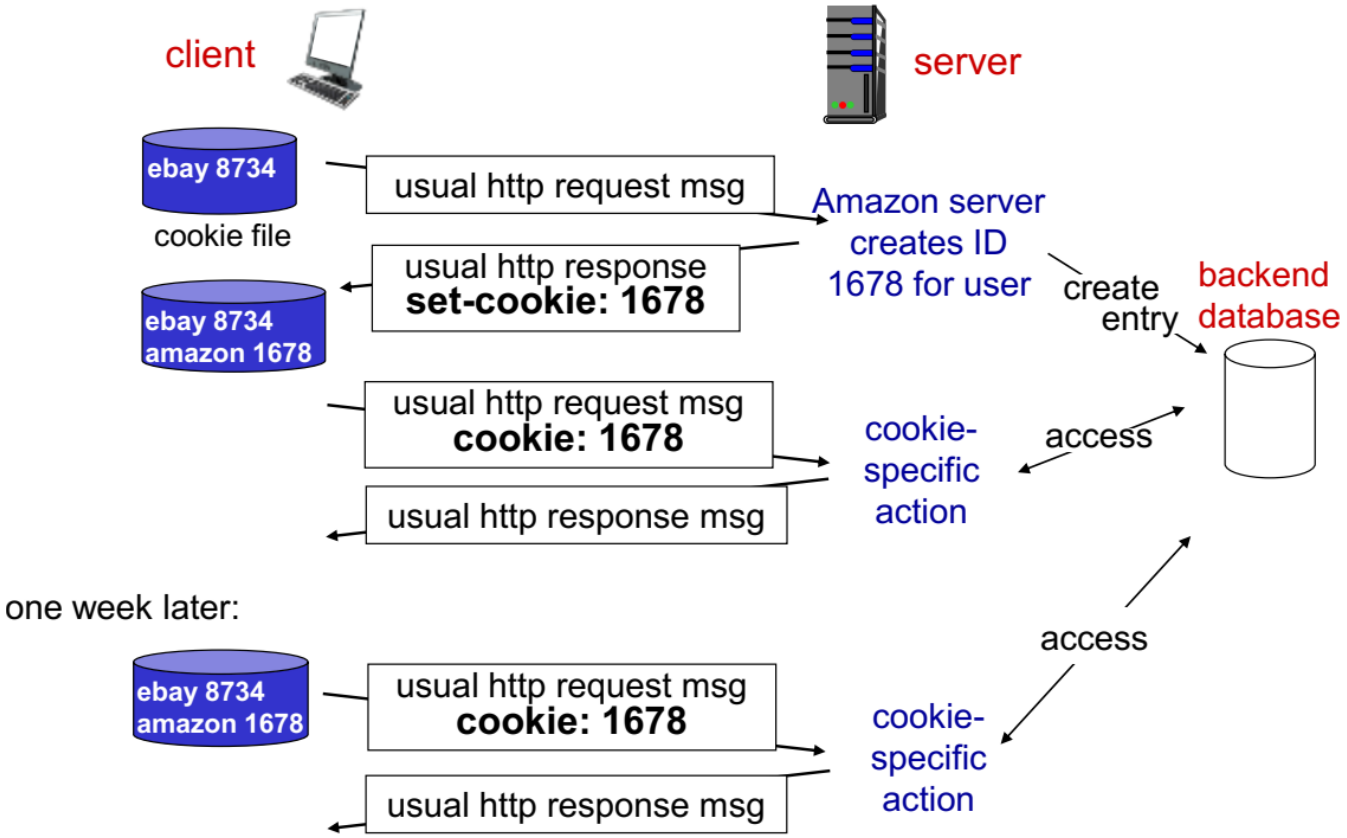
Resposta do HTTP: Código do Status

- Aparece na primeira linha da resposta enviada pelo servidor.
  - **200 OK** = Requisição bem sucedida, objeto se encontra no corpo da mensagem.
  - **301 Moved Permanently** = Objeto requisitado foi movido, nova localização se encontra no corpo da mensagem.
  - **400 Bad Request** = Servidor não entendeu a requisição.
  - **404 Not Found** = Objeto requisitado não foi encontrado no servidor.
  - **505 HTTP** = Version Not Supported

Cookies

- Muitos sites utilizam cookies.
- **Quatro componentes**:
  1) Entrada de cabeçalho relativa aos cookies na mensagem de resposta HTTP.
  2) Entrada de cabeçalho relativa aos cookies na mensagem de requisição HTTP.
  3) Arquivo de cookies mantido no host do usuário e gerenciado pelo browser.
  4) Base de dados no backend do site.

Cookies


client
server
ebay 8734
cookie file
usual http request msg
Amazon server creates ID 1678 for user
usual http response set-cookie: 1678
create entry
backend database
ebay 8734 amazon 1678
usual http request msg cookie: 1678
cookie-specific action
access
usual http response msg
one week later:
access
ebay 8734 amazon 1678
usual http request msg cookie: 1678
cookie-specific action
usual http response msg

Cookies

- **Para que cookies podem ser utilizados:**
  - Autorização.
  - “Carrinhos de compra”.
  - Recomendações.
  - Estado da sessão do usuário (Ex.: webmail)
- **Como manter “estado”:**
  - Sistemas finais mantém estado no transmissor, receptor ao longo de várias transações.
  - Cookies: mensagens HTTP carregam estado.

Cookies e privacidade

Cookies permitem que sites aprendam muito sobre você.

Você pode fornecer nome e e-mail para sites.

Web Caches (Servidores Proxy)

Cache age tanto como servidor, quanto como cliente.

- Servidor para a requisição original.
- Cliente para o servidor original do conteúdo.
- Tipicamente, cache é instalado na rede de acesso ou pelo ISP.

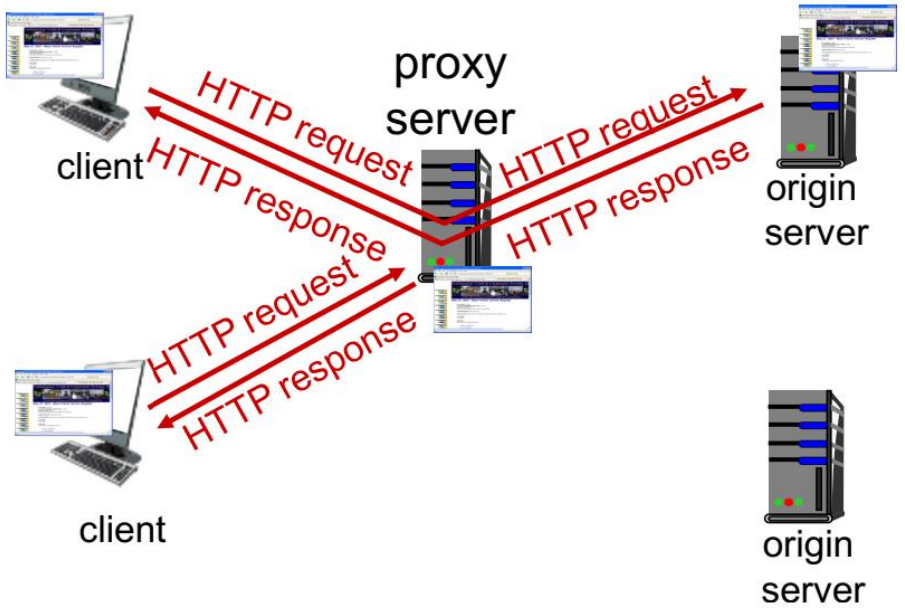
Por que usar um web cache?

- Reduzir o tempo de resposta para o cliente.
- Reduzir o tráfego no enlace de acesso da instituição.
- Internet densamente populada por cache: permite que provedores de conteúdo “pobres” entregue conteúdo de forma efetiva.
- P2P também

Web Caches (Servidores Proxy)

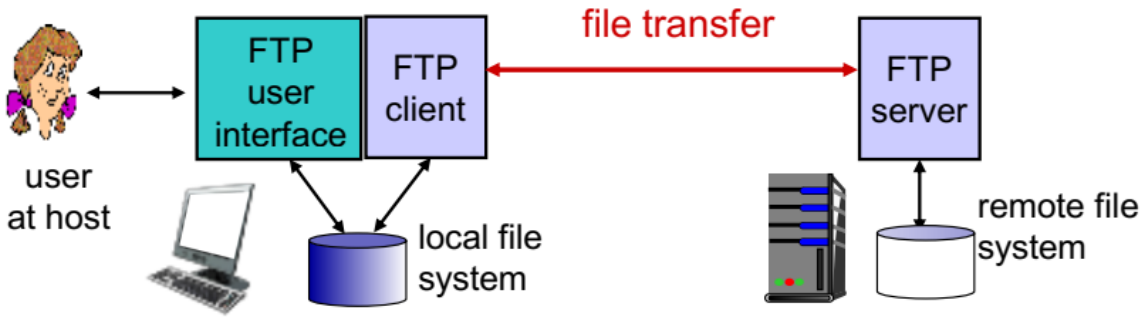
Objetivo: satisfazer requisição do cliente sem envolver servidor original do conteúdo

- Usuário configura browser: acessos web via cache.
- Browser envia todas as requisições HTTP ao cache.
- *Objetos em cache*: retornados imediatamente.
- Caso contrário, cache requisita objeto do servidor, retorna objeto ao cliente.



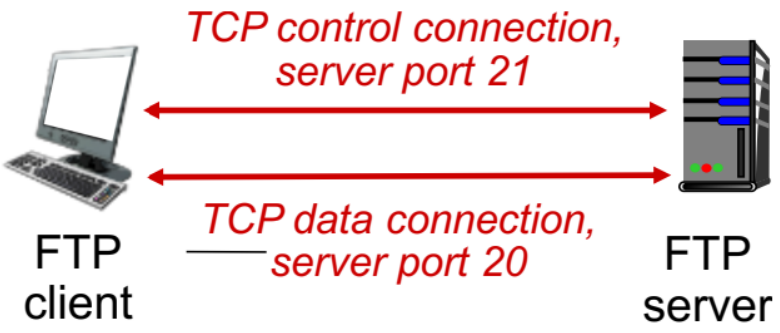
FTP

- **File Transfer Protocol**
- Transfere arquivos de/para host remoto.
- Arquitetura Cliente-Servidor.
  - Cliente: lado que inicia transferência (seja de ou para host remoto).
  - Servidor: host remoto.
- Servidor FTP: escuta, por padrão, na porta 21.



FTP

- Cliente FTP contacta servidor na porta 21, usando TCP.
- Cliente autorizado através da conexão de controle.
- Cliente navega diretórios remotos, envia comandos pela conexão de controle.
- Quando servidor recebe comando de transferência da arquivo, servidor abre 2ª conexão TCP (para arquivo) para o cliente.
- Depois de transferir arquivo, servidor fecha conexão de dados.
- Servidor abre nova conexão TCP para cada arquivo enviado.
- Conexão dedicada para controle: comunicação “fora-de-banda”.
- Servidor FTP precisa manter “estado”.
- Diretório corrente, autenticação.



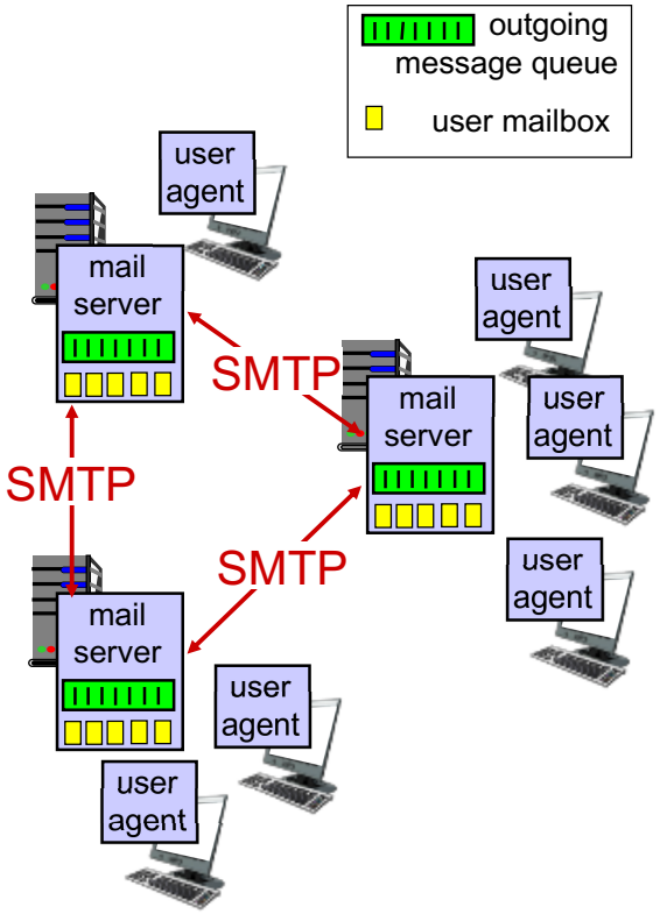
E-mail

Três grandes componentes:

- User agents.
- Servidores de e-mail.
- SMTP: Simple Mail Transfer Protocol.

User agent:

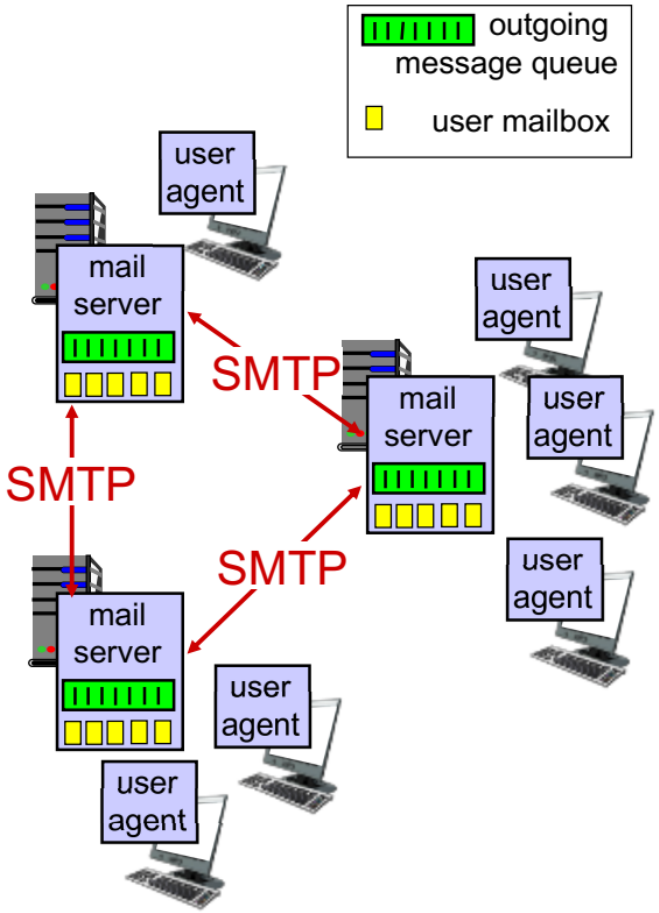
- Também conhecido como “leitor de e-mail”.
- Criação, edição, leitura de mensagens de e-mail.
  Ex.: Outlook, Thunderbird, cliente de e-mail do iPhone.
- Mensagens enviadas, recebidas armazenadas no servidor.



E-mail: Servidores de E-mail

Servidores de e-mail:

- Caixa de e-mail: contém mensagens que chegam para o usuário.
- Fila de mensagens: contém mensagens a serem enviadas.
- Protocolo SMTP: comunicação entre servidores de e-mail para envio de mensagens.
  - ➔ Cliente: servidor que envia mensagem.
  - ➔Servidor: servidor que recebe mensagem.

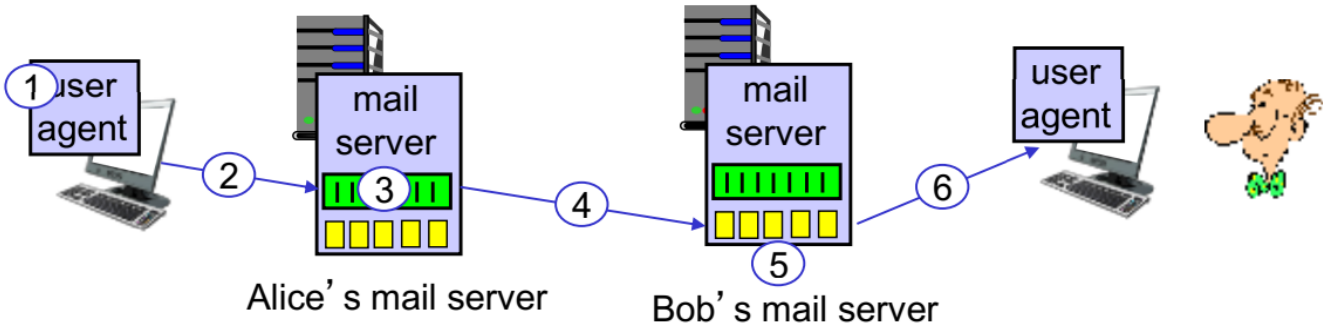


E-mail: SMTP

- Utiliza TCP para transferir mensagens de forma confiável entre cliente e servidor, porta 25.
- Transferência direta: do servidor do destinatário diretamente para o servidor do remetente.
- Protocolo em três fases:
  - Handshaking (apresentação).
  - Transferência da mensagens.
  - Encerramento.
- Interação do tipo comando/resposta (similar ao FTP, HTTP).
  - **Comandos:** texto ASCII.
  - **Resposta:** código de status e descrição.
- Mensagens necessariamente em ASCII de 7 bits

E-mail: Alice envia e-mail para Bob

1) Alice usa user agent para criar mensagem para bob@school.edu.
2) O user agent (SMTP) de Alice envia mensagem ao seu servidor de e-mail (de Alice); mensagem é enfileirada.
3) No servidor de Alice, lado cliente do SMTP abre conexão TCP para o servidor de Bob.
4) Mensagem de Alice é enviada pela conexão TCP.
5) Servidor de e-mail de Bob coloca mensagem na caixa de entrada de Bob.
6) Bob usa seu user agente (POP ou IMAP) para ler a mensagem.



E-mail: POP 3

Fase de autorização:

- Comandos do cliente:
  - **user:** declara o nome do usuário.
  - **pass:** senha.
- Servidor responde:
  - **+OK**
  - **-ERR**

Fase de transações:

- Cliente:
- **list:** lista números das mensagens.
- **retr:** obtém mensagem por número.
- **dele:** apaga mensagem.
- **quit:** encerra comunicação

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

```
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
```

E-mail: POP3 e IMAP

Mais sobre o POP3:

- Exemplo anterior usa POP3 no modo “download and delete”.
- Mensagens são baixadas para host do Bob e apagadas do servidor.
- Há também o modo “download and keep”: cópias são deixadas no servidor.
- POP3 é stateless entre sessões

IMAP:

- Mantém mensagens apenas no servidor.
- Permite ao usuário organizar mensagens em pastas.
- Mantém estado entre sessões do usuário.
- Nomes de pastas e mapeamentos de mensagens para pastas.

Contato

Professor:
André Saraiva, MSc

E-mail:
andre.saraiva@univassouras.edu.br