

UFCG
Universidade Federal
de Campina Grande



Redes de Computadores

Camada de Aplicação

Professor: Reinaldo Gomes
reinaldo@dsc.ufcg.edu.br

1

UFCG
Universidade Federal
de Campina Grande

Camada de aplicação

- Princípios de aplicações em rede de computadores
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS
- Programação de socket com TCP
- Programação de socket com UDP

2

UFCG
Universidade Federal
de Campina Grande

Algumas aplicações de rede

- Correio Eletrônico (e-mail)
- Hipertexto (<http://www...>)
- Mensagem instantânea
- Login remoto (e.g., ssh)
- Compartilhamento de Arquivos Entre-Pares (P2P *file sharing*)
- Jogos multi-usuário em Rede
- *Streaming stored* videoclipes
- Telefonia via Internet (VoIP)
- Videoconferência em tempo real
- Grades Computacionais (*grid computing*)

3

UFCG
Universidade Federal
de Campina Grande

Criando uma nova aplicação de rede

Escrever programas que

- Executem sobre diferentes sistemas finais e
- Se comuniquem através de uma rede.
- Ex.: *Software* de servidor Web se comunicando com *software* do navegador (*browser*).

Nenhum software é escrito para dispositivos no núcleo da rede

- Dispositivos do núcleo da rede não trabalham na camada de aplicação
- Esta estrutura permite um rápido desenvolvimento de aplicação



4

UFCG
Universidade Federal
de Campina Grande

Camada de aplicação

- 2.1 Princípios de aplicações em rede de computadores
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Compartilhamento de arquivos P2P

5

UFCG
Universidade Federal
de Campina Grande

Arquiteturas de aplicação

- Cliente-servidor
- Entre-Pares (Peer-to-peer, P2P)
- Híbrida de cliente-servidor e P2P

6

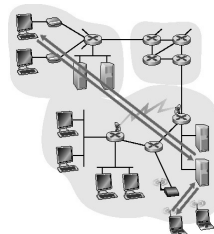
Arquitetura cliente-servidor

Servidor:

- Hospedeiro sempre ativo
- Endereço IP permanente
- Fornece serviços solicitados pelo cliente

Clientes:

- Comunicam-se com o servidor
- Pode ser conectado intermitentemente
- Pode ter endereço IP dinâmico
- Não se comunicam diretamente uns com os outros

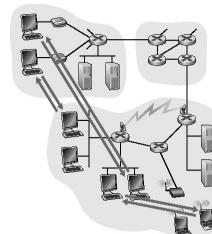


7

Arquitetura P2P pura

- Nem sempre no servidor
- Sistemas finais arbitrários comunicam-se diretamente
- Pares são intermitentemente conectados e trocam endereços IP
- Ex.: Gnutella

Altamente escaláveis mas difíceis de gerenciar



8

Híbrida de cliente-servidor e P2P

Napster

- Transferência de arquivo P2P
- Busca centralizada de arquivos:
- Conteúdo de registro dos pares no servidor central
- Consulta de pares no mesmo servidor central para localizar o conteúdo

Instant messaging

- Bate-papo entre dois usuários é P2P
- Detecção/localização de presença é centralizada:
- Usuário registra seu endereço IP com o servidor central quando fica on-line
- Usuário contata o servidor central para encontrar endereços IP dos "amigos"

9

O protocolo da camada de aplicação define

- Tipo das mensagens trocadas, mensagens de requisição e resposta
- Sintaxe dos tipos de mensagem: os campos nas mensagens e como são delineados
- Semântica dos campos, ou seja, significado da informação nos campos
- Regras para quando e como os processos enviam e respondem às mensagens

Protocolos de domínio público:

- Definidos nas RFCs
- Recomendados para interoperabilidade
- Ex.: HTTP, SMTP, DNS, SSH

Protocolos proprietários:

- Ex.: KaZaA, BitTorrent

10

De qual serviço de transporte uma aplicação necessita?

Perda de dados

- Algumas aplicações (ex.: áudio) podem tolerar alguma perda
- Outras aplicações (ex.: transferência de arquivos, sessão remota) exigem transferência de dados 100% confiável

Temporização

- Algumas aplicações (ex.: telefonia IP, jogos interativos) exigem baixos atrasos para serem "efetivos"

Banda passante

- Algumas aplicações (ex.: multimídia) exigem uma banda mínima para serem "efetivas"
- Outras aplicações ("aplicações elásticas") melhoram quando a banda disponível aumenta

11

Requisitos de transporte de aplicações comuns

Aplicação	Perdas	Taxa	Sensível ao atraso
file transfer	sem perdas	elástica	não
e-mail	sem perdas	elástica	não
Web documents	sem perdas	elástica	não
real-time áudio/video	tolerante	áudio: 5 Kb-1Mb vídeo: 10Kb-5Mb	sim, 100's mseg
stored áudio/video	tolerante	igual à anterior	sim, segundos
jogos interativos	tolerante	kbps	sim, 100's mseg
e-business	sem perda	elástica	sim

12

Serviços dos protocolos de transporte da Internet

Serviço TCP:

- Orientado à conexão: conexão requerida entre processos cliente e servidor
- Transporte confiável entre os processos de envio e recepção
- Controle de fluxo: o transmissor não sobrecarrega o receptor
- Controle de congestionamento: protege a rede do excesso de tráfego
- Não oferece: garantias de temporização e de banda mínima

Serviço UDP:

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

13

Aplicação 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 (ex.: RealNetworks)	TCP ou UDP
servidor de arquivos remoto	NFS	TCP ou UDP
telefonia Internet	RTP ou proprietário (ex.: Vocaltec)	tipicamente UDP

14

Camada de aplicação

- Princípios de aplicações em rede de computadores
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS
- Programação de socket com TCP
- Programação de socket com UDP

15

Visão geral do HTTP

HTTP: *hypertext transfer protocol*

- Protocolo da camada de aplicação da Web
- Modelo cliente/servidor
 - Cliente: navegador que solicita, recebe e apresenta objetos da Web
 - Servidor: envia objetos em resposta a pedidos
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068

16

Visão geral do HTTP

Utiliza 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

HTTP é "stateless"

- Por *default*, o servidor não mantém informação sobre os pedidos passados pelos clientes

Protocolos que mantêm informações de "estado" são complexos!

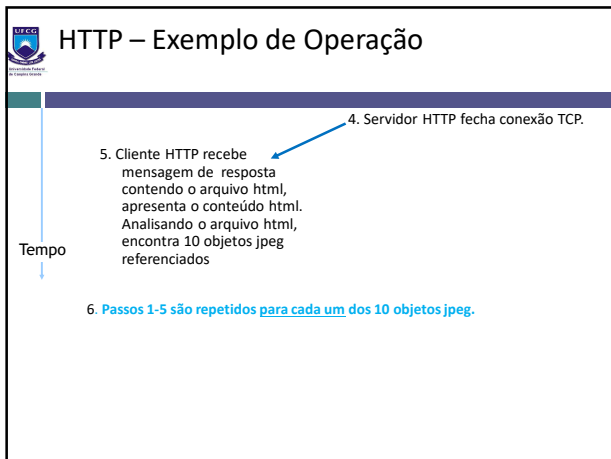
- Histórico do passado (estado) deve ser mantido
- Se o servidor/cliente quebra, suas visões de "estado" podem ser inconsistentes, devendo ser reconciliadas

17

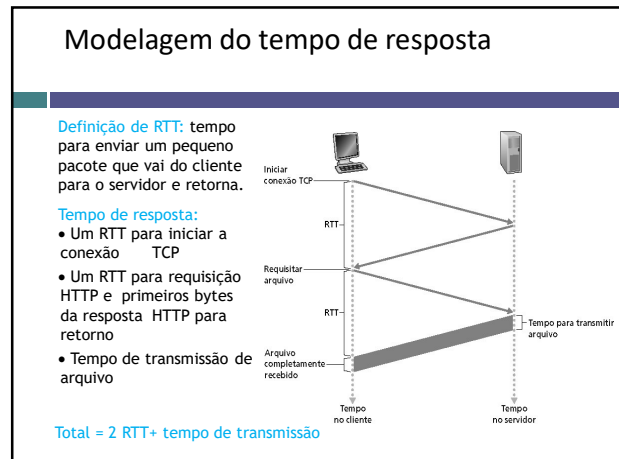
HTTP – Exemplo de Operação

Usuário entra com a URL: (contém texto, referências a 10 imagens jpeg)
 www.someSchool.edu/someDepartment/home.index

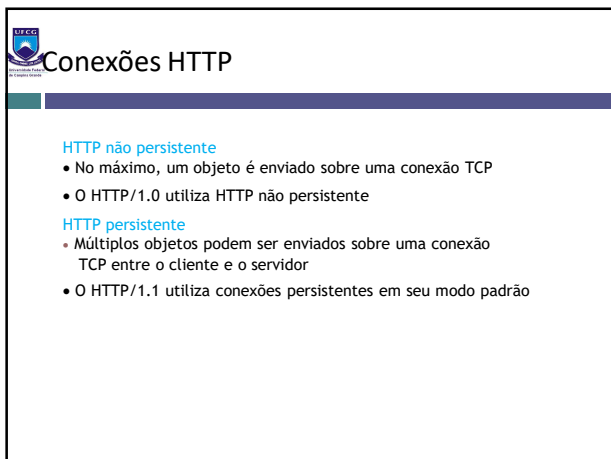
18



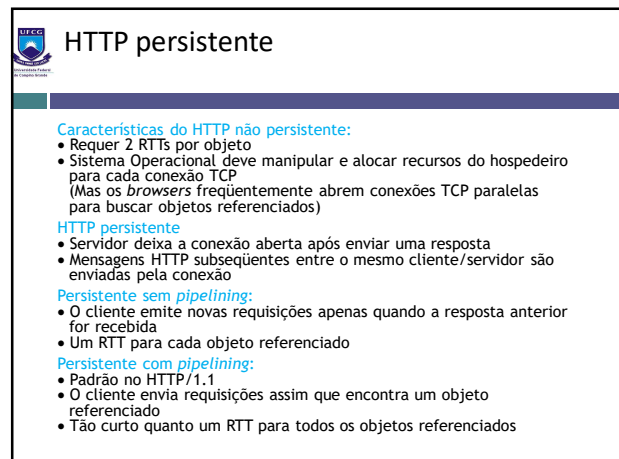
19



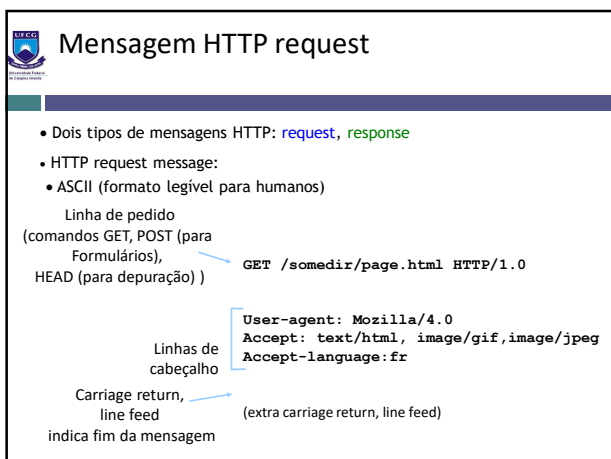
20



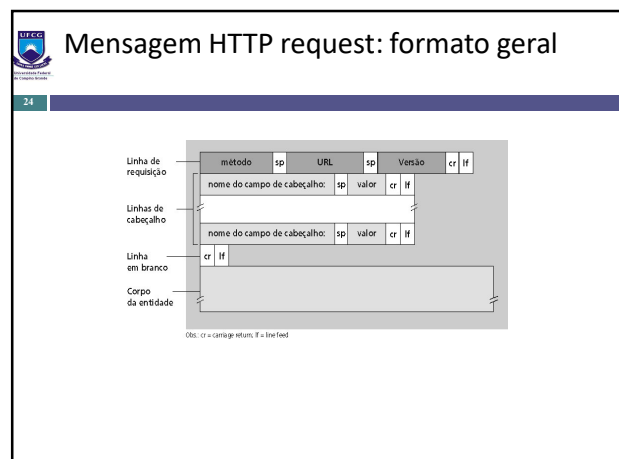
21



22



23



24

Entrada de formulário

Método Post:

- Página Web frequentemente inclui entrada de formulário
- A entrada é enviada para o servidor no corpo da entidade

Método URL:

- Utiliza o método GET
- A entrada é enviada no campo de URL da linha de requisição:

www.somesite.com/animalsearch?monkeys&banana

25

Tipos de métodos

HTTP/1.0

- GET
- POST
- HEAD
- Pede para o servidor deixar o objeto requisitado fora da resposta

HTTP/1.1

- GET, POST, HEAD
- PUT
- Envia o arquivo no corpo da entidade para o caminho especificado no campo de URL
- DELETE
- Apaga o arquivo especificado no campo de URL
- TRACE
- Loop-back em nível de aplicação (request é enviado no corpo da resposta)

26

Mensagem HTTP response

Linhas de status (protocolo, código de status, frase de status)

Linhas de cabeçalho

Dados, ex.: 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 ...
  
```

27

Códigos de *status* das respostas

Na primeira linha da mensagem de resposta servidor → cliente. Alguns exemplos de códigos:

200 OK

- Requisição bem-sucedida, objeto requisitado a seguir nesta mensagem

301 Moved permanently

- Objeto requisitado foi movido, nova localização especificada a seguir nesta mensagem (Location:)

400 Bad request

- Mensagem de requisição não compreendida pelo servidor

404 Not Found

- Documento requisitado não encontrado neste servidor

505 HTTP version not supported

28

Estado usuário-servidor: *cookies*

A maioria dos grandes sites Web utilizam *cookies*

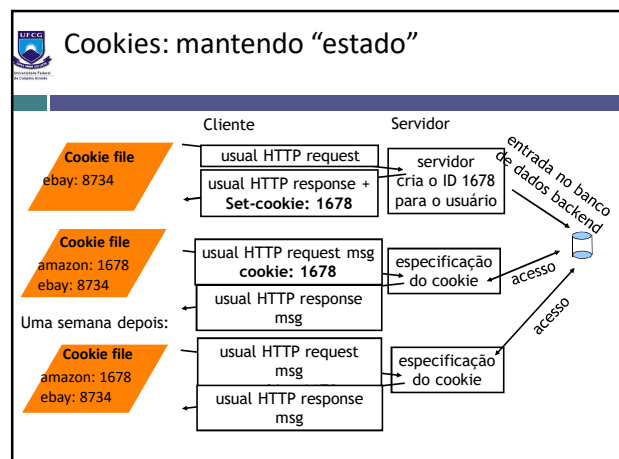
Quatro componentes:

- 1) Linha de cabeçalho do *cookie* na mensagem HTTP response
- 2) Linha de cabeçalho de *cookie* na mensagem HTTP request
- 3) Arquivo de *cookie* mantido no hospedeiro do usuário e manipulado pelo *browser* do usuário
- 4) Banco de dados *backend* no site Web

Exemplo:

- Susan acessa a Internet sempre do mesmo PC
- Ela visita um site específico de comércio eletrônico pela primeira vez
- Quando a requisição HTTP inicial chega ao site, este cria um identificador (ID) único e uma entrada no banco de dados *backend* para este ID

29



30

Cookies

O que os cookies podem trazer:

- Autorização
- Cartões de compra
- Recomendações
- Estado de sessão do usuário (Web e-mail)

Cookies e privacidade:

- Cookies permitem que *sites* saibam muito sobre você
- Você pode fornecer nome e e-mail para os *sites*
- Mecanismos de busca usam redirecionamento e cookies para saberem mais sobre você
- Companhias de propaganda obtêm informações por meio dos *sites*

31

Web caches (proxy server)

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

- Usuário configura o *browser*: acesso Web é feito por meio de um procurador (*proxy*).
- Cliente envia todos os pedidos HTTP para o *proxy*
- Se o objeto existe na *cache* do procurador: o procurador retorna o objeto
- Caso contrário, o procurador solicita o(s) objeto(s) do servidor original, e então envia o(s) objeto(s) ao cliente

32

Mais sobre Web caching

- A *cache* atua tanto no servidor como no cliente
- Tipicamente, a *cache* é instalada pelo ISP (universidade, companhia, ISP residencial)

Por que Web caching?

- Reduz o tempo de resposta para a requisição do cliente.
- Reduz o tráfego em um enlace de acesso de uma instituição.
- A densidade de *caches* na Internet habilita os “fracos” provedores de conteúdo a efetivamente entregarem o conteúdo (mas fazendo P2P file sharing)

33

Porque Web Caching?

- armazenamento está “perto” do cliente (ex., na mesma rede)
- menor tempo de resposta
- reduz o tráfego para servidor distante
 - links externos podem ser caros e facilmente congestionáveis

34

GET condicional

Cliente **Servidor**

- 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
- Servidor: resposta contém objeto se a cópia é atualizada: HTTP/1.1 200 OK <data>

35

Camada de aplicação

- Princípios de aplicações de rede
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS
- Programação de socket com TCP
- Programação de socket com UDP

36

Correio eletrônico

Três componentes principais:

- Agentes de usuário
- Servidores de correio
- Simple mail transfer protocol: SMTP

Agente de usuário

- “leitor de correio”
- Composição, edição, leitura de mensagens de correio
- Ex.: Eudora, Outlook, elm, Netscape Messenger, Thunderbird
- Mensagens de entrada e de saída são armazenadas no servidor

37

Correio eletrônico: servidores de correio

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 si

- Cliente: servidor de correio que envia
- “servidor”: servidor de correio que recebe

38

Correio eletrônico: SMTP [RFC 821]

- Usa TCP para transferência confiável de mensagens de correio 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

39

Cenário: Alice envia mensagem para Bob

- 1) Alice usa o agente de usuário (*User Agent*) para compor a mensagem “para” bob@someschool.edu
- 2) O agente de usuário dela envia a mensagem para o seu servidor de correio; a mensagem é colocada na fila de mensagens.
- 3) O lado cliente do SMTP abre uma conexão TCP com o servidor de correio do Bob.
- 4) O cliente SMTP envia a mensagem de Alice pela conexão TCP.
- 5) O servidor de correio de Bob coloca a mensagem na caixa de correio de Bob.
- 6) Bob invoca seu agente de usuário para ler a mensagem.

40

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
  
```

41

SMTP: palavras finais

- SMTP usa conexões persistentes
- SMTP exige que as mensagens (cabeçalho e corpo) estejam em ASCII de 7 bits
- Servidor SMTP usa CRLF.CRLF para indicar o final da mensagem

Comparação com HTTP:

- HTTP: pull
- E-mail: push
- Ambos usam comandos e respostas em ASCII, interação comando/resposta e códigos de status
- HTTP: cada objeto encapsulado na sua própria mensagem de resposta
- SMTP: múltiplos objetos são enviados numa mensagem multiparte

42

Formato da mensagem de correio

SMTP: protocolo para trocar mensagens de e-mail
 RFC 822: padrão para mensagens do tipo texto:

- linhas de cabeçalho, ex.:
 - To:
 - From:
 - Subject:
 diferente dos comandos HTTP
- corpo
 - a "mensagem", ASCII somente com caracteres

43

Protocolos de acesso ao correio

- SMTP: entrega e armazena no servidor do destino
- 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, Gmail, etc.

44

Protocolo POP3

Fase de autorização

- comandos do cliente:
 - user: declara nome do usuário
 - pass: password
- respostas do servidor
 - +OK
 - ERR

Fase de transação

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

```

S: +OK POP3 server ready
C: user alice
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
  
```

45

POP3 (continuação) e IMAP

Mais sobre POP3

- O exemplo anterior usa o modo *download-and-delete*
- Bob não pode reler o e-mail se ele trocar o cliente
- download-and-keep*: cópias das mensagens em clientes diferentes
- POP3 é *stateless* através das sessões

IMAP

- Mantém todas as mensagens em um lugar: o servidor
- Permite que o usuário organize as mensagens em pastas
- IMAP mantém o estado do usuário através das sessões:
 - Nomes das pastas e mapeamentos entre os IDs da mensagem e o nome da pasta

46

Camada de aplicação

- Princípios de aplicações de rede
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS
- Programação de socket com TCP
- Programação de socket com UDP

47

DNS: Domain Name System

Pessoas: muitos identificadores:

- RG, nome, passaporte, CPF

Internet hospedeiros, roteadores:

- Endereços IP (32 bits) - usados para endereçar datagramas
- "nome", ex.: www.dsc.ufcg.edu.br - usados por humanos

P.: Como relacionar nomes com endereços IP?

Domain Name System:

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

48

DNS

DNS services

- Nome do hospedeiro para tradução de endereço IP
- Hospedeiro *aliasing* (apelido)
 - Nomes canônicos e *alias*
- mail server *aliasing*
- distribuição de carga
- Servidores Web replicados: estabelece o endereço IP para um nome canônico

Por que não centralizar o DNS?

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

Não é escalável!

49

Base de dados distribuída, hierárquica

Cliente quer o IP para www.amazon.com; 1ª aprox.:

- Cliente consulta um servidor de raiz para encontrar o servidor DNS **.com**
- Cliente consulta o servidor DNS **.com** para obter o servidor DNS **amazon.com**
- Cliente consulta o servidor DNS **amazon.com** para obter o endereço IP para www.amazon.com

50

DNS: servidores de nomes raiz

- São contatados pelos servidores de nomes locais que não podem resolver um nome
- Servidores de nomes raiz:
 - Buscam servidores de nomes autorizados 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

51

Servidores TLD e autoritários

Servidores Top-Level Domain (TLD):

responsáveis pelos domínios com, org, net, edu etc e todos os domínios top-level nacionais uk, fr, ca, jp, br

- Network Solutions mantém servidores para o TLD “com”
- Educause para o TLD “edu”

Servidores DNS autorizados:

servidores DNS de organizações, provêem nome de hospedeiro autorizado para mapeamentos IP para servidores de organizações (ex.: Web e mail).

- Podem ser mantidos por uma organização ou provedor de serviços

52

Servidor de nomes local

- Não pertence estritamente a uma hierarquia
- Cada ISP (ISP residencial, companhia, universidade) possui um
 - Também chamado de “servidor de nomes default”
- Quando um hospedeiro faz uma pergunta a um DNS, a pergunta é enviada para seu servidor DNS local
 - Age como um procurador (proxy), encaminhando as perguntas para dentro da hierarquia

53

Exemplo

- O hospedeiro em cis.poly.edu quer o endereço IP para gala.cs.umass.edu

54

Consultas recursivas

Consulta recursiva:

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

Consulta encadeada:

- Servidor consultado responde com o nome de outro servidor de nomes para contato
- “eu não sei isto, mas pergunte a este servidor”

55

DNS: armazenando e atualizando registros

Uma vez que um servidor de nomes aprende um mapeamento, ele armazena o mapeamento num registro do tipo **cache**

- Registro do **cache** tornam-se obsoletos (desaparecem) depois de um certo tempo
- Servidores TLD são tipicamente armazenados em **cache** nos servidores de nome locais

Mecanismos de atualização e notificação estão sendo projetados pelo IETF

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

56

Registros do DNS

DNS: base de dados distribuída que armazena registros de recursos (RR)

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

- Type = A
 - name é o nome do computador
 - value é o endereço IP
- Type = NS
 - name é um domínio (ex.: foo.com)
 - value é o endereço IP do servidor de nomes autorizados para este domínio
- Type = CNAME
 - name é um “apelido” para algum nome “canônico” (o nome real)
 - www.ibm.com é realmente www.ibm.com.cs186.net
 - value é o nome canônico
- Type = MX
 - value é o nome do servidor de correio associado com name

57

DNS: protocolo e mensagem

Protocolo DNS: mensagem de consulta e resposta, ambas com o mesmo formato de mensagem

Cabeçalho da msg

- Identificação: número de 16 bits para consulta, resposta usa o mesmo número
- Flags:
 - Consulta ou resposta
 - Recursão desejada
 - Recursão disponível
 - Resposta é autorizada

Identificação	Flags	
Número de perguntas	Número de RRs de resposta	12 bytes
Número de RRs com autoridade	Número de RRs adicionais	
Perguntas (número variável de perguntas)	Respostas (número variável de registros de recursos)	Nome, campos de tipo para uma consulta
Autoridade (número variável de registros de recursos)	Informação adicional (número variável de registros de recursos)	RRs de resposta à consulta
		Registros para servidores com autoridade
		Informação adicional "VLI" que pode ser usada

58

Camada de aplicação

Identificação	Flags	
Número de perguntas	Número de RRs de resposta	12 bytes
Número de RRs com autoridade	Número de RRs adicionais	
Perguntas (número variável de perguntas)	Respostas (número variável de registros de recursos)	Nome, campos de tipo para uma consulta
Autoridade (número variável de registros de recursos)	Informação adicional (número variável de registros de recursos)	RRs de resposta à consulta
		Registros para servidores com autoridade
		Informação adicional "VLI" que pode ser usada

DNS: protocolo e mensagens

59

Camada de aplicação

- Exemplo: empresa recém-criada “Network Utopia”
- Registrar o nome networkutopia.com num “registrar” (ex.: Network Solutions)
 - É necessário fornecer ao registrar os nomes e endereços IP do seu servidor de nomes autorizados (primário e secundário)
 - Registrar insere dois RRs no servidor TLD do domínio com:


```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```
- No servidor autorizado, inserir um registro Tipo A para www.networkutopia.com e um registro Tipo MX para networkutopia.com
- Como as pessoas obtêm o endereço IP do seu Web site?

Inserindo registros no DNS

60

Camada de aplicação

- Princípios de aplicações de rede
- Web e HTTP
- Correio electrónico
SMTP, POP3, IMAP
- DNS
- Programação de socket com TCP
- Programação de socket com UDP

61

Sockets

- Um processo envia/recebe mensagens para/de seu **socket**
- O **socket** é análogo a uma porta
- O processo de envio empurra a mensagem para fora da porta
- O processo de envio confia na infra-estrutura de transporte no outro lado da porta que leva a mensagem para o **socket** no processo de recepção.

62

Programação de sockets com TCP

Socket: uma porta entre o processo de aplicação e o protocolo de transporte fim-a-fim (UDP ou TCP)

Serviço TCP: transferência confiável de **bytes** de um processo para outro

63

Programação de sockets com TCP

Cliente deve contatar o servidor

- Processo servidor já deve estar em execução
- Servidor deve ter criado socket (porta) que aceita o contato do cliente

Cliente contata o servidor

- Criando um socket TCP local
- Especificando endereço IP e número da porta do processo servidor
- Quando o cliente cria o socket: cliente TCP estabelece conexão com o TCP do servidor

Quando contatado pelo cliente, o TCP do servidor cria um novo socket para o processo servidor comunicar-se com o cliente

- Permite ao servidor conversar com múltiplos clientes
- Número da porta de origem é utilizada para distinguir o cliente (mais sobre isso quando abordarmos a camada de Transporte)

Ponto de vista da aplicação
TCP fornece a transferência confiável e ordenada entre o cliente e o servidor

64

Jargão: *stream*

- Um **stream** é uma seqüência de caracteres que fluem para dentro ou para fora de um processo
- Um **stream de entrada** é associada a alguma fonte de entrada para o processo. Ex.: teclado ou **socket**
- Um **stream de saída** é associada a uma fonte de saída. Ex.: monitor ou **socket**

65

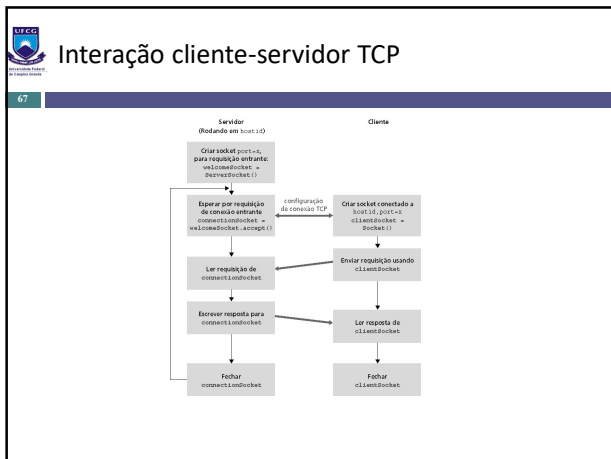
Programação de sockets com TCP

Exemplo de aplicação cliente-servidor:

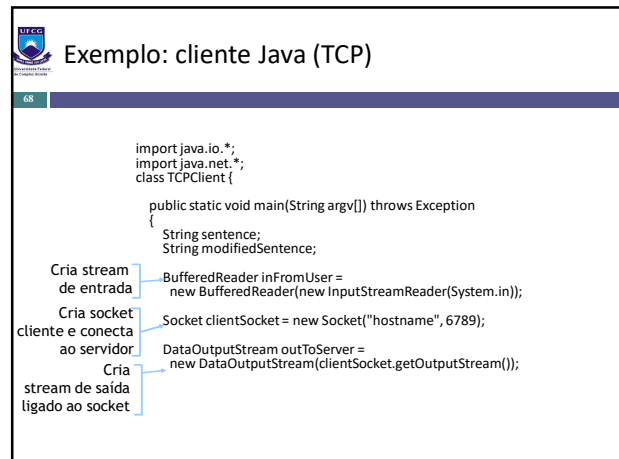
- Cliente lê linha da entrada-padrão do sistema (inFromUser stream), envia para o servidor via socket (outToServer stream)
- Servidor lê linha do socket
- Servidor converte linha para letras maiúsculas e envia de volta ao cliente
- Cliente lê a linha modificada através do (inFromServer stream)

Programação de sockets com TCP

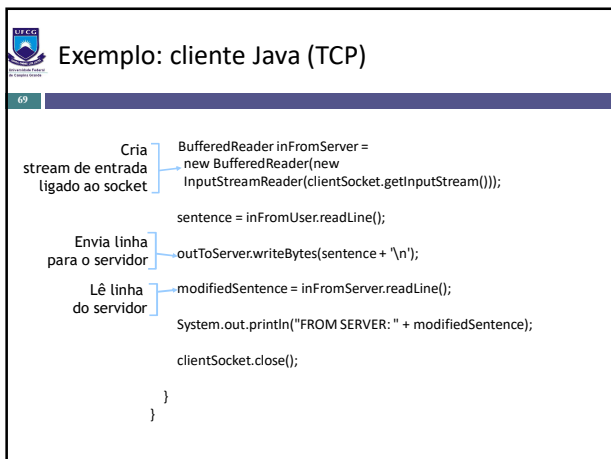
66



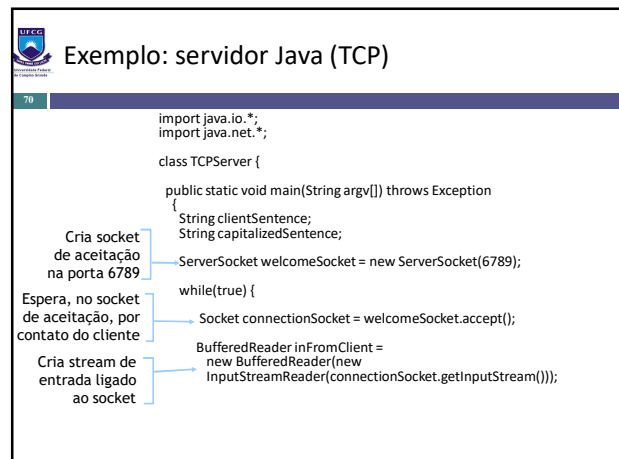
67



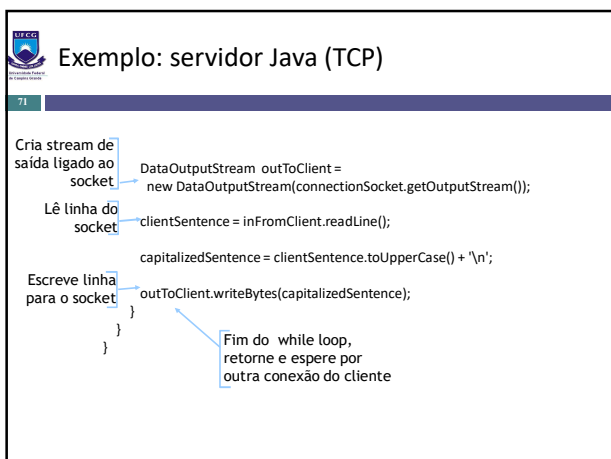
68



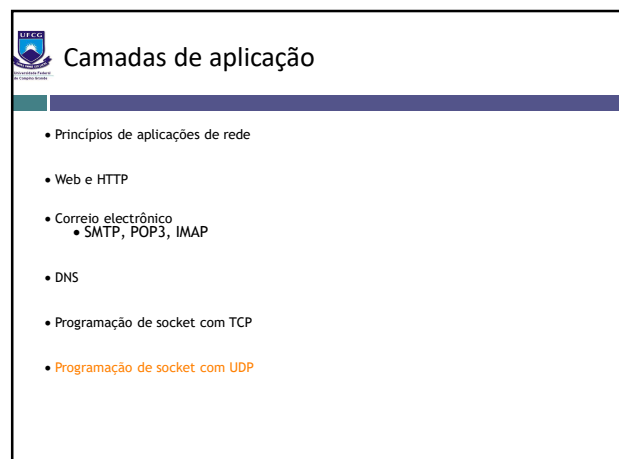
69



70



71



72

Programação de sockets com UDP

UDP: não há conexão entre o cliente e o servidor

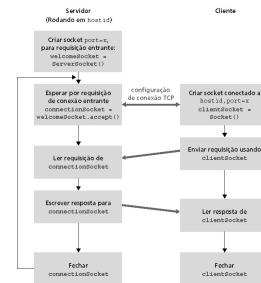
- Não existe apresentação
- Transmissor envia explicitamente endereço IP e porta de destino em cada mensagem
- Servidor deve extrair o endereço IP e porta do transmissor de cada datagrama recebido
- Dados transmitidos podem ser recebidos fora de ordem ou perdidos

Ponto de vista da aplicação:

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

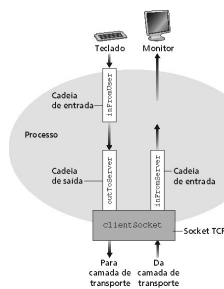
73

Interação cliente-servidor: UDP



74

Exemplo: cliente Java (UDP)



75

Exemplo: cliente Java (UDP)

```

import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception {
        // Cria stream de entrada
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));

        // Cria socket cliente
        DatagramSocket clientSocket = new DatagramSocket();

        // Translada nome do hospedeiro para endereço IP usando DNS
        InetAddress IPAddress = InetAddress.getByName("hostname");

        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
    }
}
  
```

76

Exemplo: cliente Java (UDP)

```

// Cria datagrama com dados a enviar, tamanho, endereço IP porta
DatagramPacket sendPacket =
    new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

// Envia datagrama para servidor
clientSocket.send(sendPacket);

// Lê datagrama do servidor
DatagramPacket receivePacket =
    new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);

String modifiedSentence =
    new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}
  
```

77

Exemplo: servidor Java (UDP)

```

import java.io.*;
import java.net.*;


class UDPServer {
    public static void main(String args[]) throws Exception {
        // Cria socket datagrama na porta 9876
        DatagramSocket serverSocket = new DatagramSocket(9876);

        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];

        while(true) {
            // Cria espaço para datagramas recebidos
            DatagramPacket receivePacket =
                new DatagramPacket(receiveData, receiveData.length);

            // Recebe datagrama
            serverSocket.receive(receivePacket);
        }
    }
}
  
```

78

 Exemplo: servidor Java (UDP)

```
String sentence = new String(receivePacket.getData());
Obtém endereço IP e número da porta do transmissor → InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
String capitalizedSentence = sentence.toUpperCase();
sendData = capitalizedSentence.getBytes();
Cria datagrama para enviar ao cliente → DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);
Escreve o datagrama para dentro do socket → serverSocket.send(sendPacket);
}
Termina o while loop, retorna e espera por outro datagrama
```

79