DNS: Domain Name System

- Há duas maneiras de identificar um hospedeiro – por um nome de hospedeiro e por um endereço IP.

- Para conciliar essas duas formas, é necessário um serviço de diretório que traduza nomes de hospedeiro para endereços IP.

- Esta é a tarefa principal do DNS da Internet.

DNS: Domain Name System

Pessoas: muitos identificadores:

- CPF, nome, nº da identidade

Hosts e roteadores da Internet :

- endereço IP (32 bits) - usado p/ endereçar datagramas
- "nome", ex., jambo.ic.uff.br - usado pelos usuários

DNS: Domain Name System

Problemas:

1. nome dos hosts fornecem muito pouca informação (para os algoritmos de roteamento) sobre a localização do mesmo na Internet:

Ex: www.eurecom.fr

2. tamanho dos nomes é variável e de difícil processamento pelos roteadores. Já o endereço IP tem tamanho fixo.

Ex: 192.164.16.1

Questão: como realizar o mapeamento entre o nome do host e seu endereço IP?

DNS: Domain Name System

Domain Name System:

- *base de dados distribuída* – implementada numa hierarquia de muitos *servidores de nomes*
- *protocolo da camada de aplicação* – permite que hospedeiros, roteadores e servidores de nomes se comuniquem para *resolver* nomes (tradução endereço/nome)
  - função do ***núcleo*** da Internet implementada como protocolo de camada de aplicação, mas não é uma aplicação com a qual o usuário interage diretamente
  - DNS é um exemplo da filosofia: "deixar a complexidade na borda da rede"

DNS: Domain Name System

- Usado pelos protocolos HTTP e SMTP para traduzir os hostnames fornecidos pelo usuário para endereços IP

Exemplo: suponha que um *navegador* queira acessar a URL www.someschool.edu/index.html. O endereço IP é obtido da seguinte maneira:

- O browser extrai o *hostname*: www.someschool.edu da URL e passa o hostname para o lado cliente da aplicação DNS;
- O cliente DNS envia uma solicitação contendo o hostname a um servidor DNS;
- O cliente eventualmente recebe uma resposta contendo o endereço IP referente ao hostname
- Uma vez recebido o endereço IP, o browser abre uma conexão TCP com o servidor HTTP (porta 80) no endereço IP obtido.

DNS: Domain Name System

Portanto:

- DNS introduz um atraso adicional nas aplicações que o utilizam;
- Mas, geralmente, o mapeamento desejado se encontra armazenado no "cache" de um servidor próximo ao host que fez a solicitação;

  ⇒

  1. ajuda a reduzir o atraso do DNS;
  2. ajuda a reduzir o tráfego na rede;

DNS: Domain Name System

- Servidores DNS:
  - máquinas UNIX rodando o software BIND (Berkeley Internet Name Domain)

- DNS roda sobre UDP e usa a porta 53

DNS: Domain Name System

Outros serviços:

- ***apelidos para hosts*** com nomes complicados (aliasing):
  Ex: ***enterprise.com*** – ***relay1.west-coast.enterprise.com***

  Obs: ***relay1.west-coast.enterprise.com*** é chamado de ***nome canônico***

- ***apelido para o servidor de mails*** (normalmente o domínio é um apelido para o nome do servidor)

Obs: pode-se ter o mesmo apelido tanto para um servidor Web quanto para um servidor de email

- ***distribuição da carga*** (servidor fornece um conjunto de IPs que se alternam no topo de uma lista para uma certa URL, por ex., cnn.com)
  → replicação de servidores

Cliente normalmente "pega" o primeiro endereço da lista
(ex.: web e e-mail)

Servidores de nomes DNS

Por que não centralizar o DNS? (projeto mais simples)

- ponto único de falha
- volume de tráfego (centenas de milhões de pedidos)
- base de dados centralizada e distante (não pode estar perto de todos os hosts!)

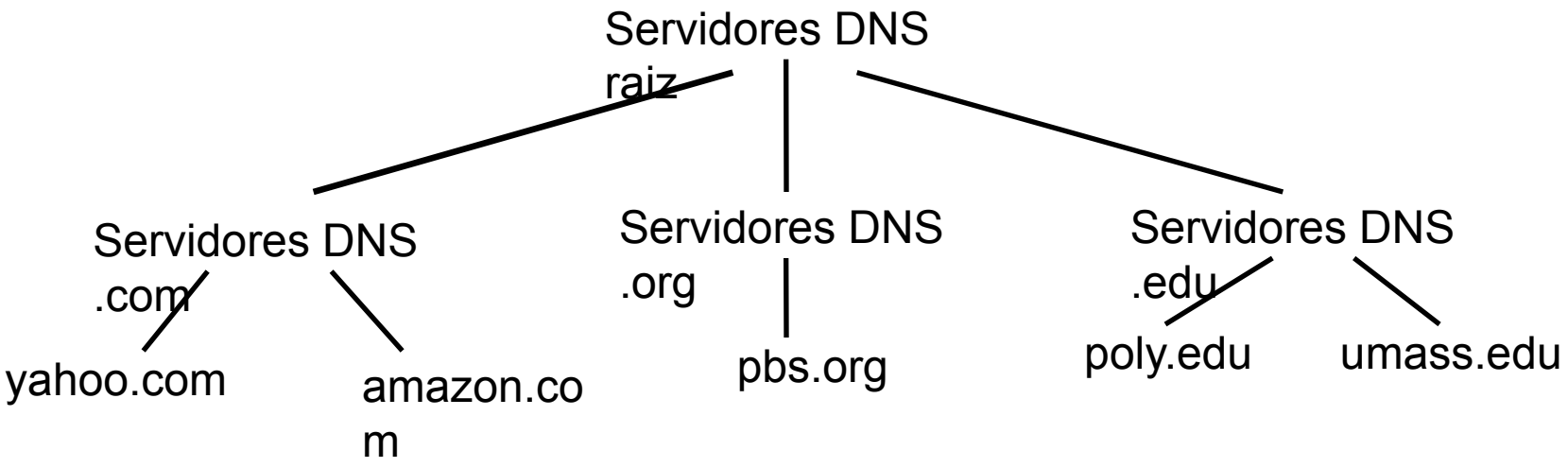
  ⇒ aumento no atraso

- manutenção da BD: grande demais, cuidar da atualização para cada novo *host*

⇒ ***Não é escalável!***

⇒ ***Nenhum servidor de nomes mantém todos os mapeamentos. Mapeamentos são distribuídos pelos servidores***

Banco de Dados Distribuído e Hierárquico

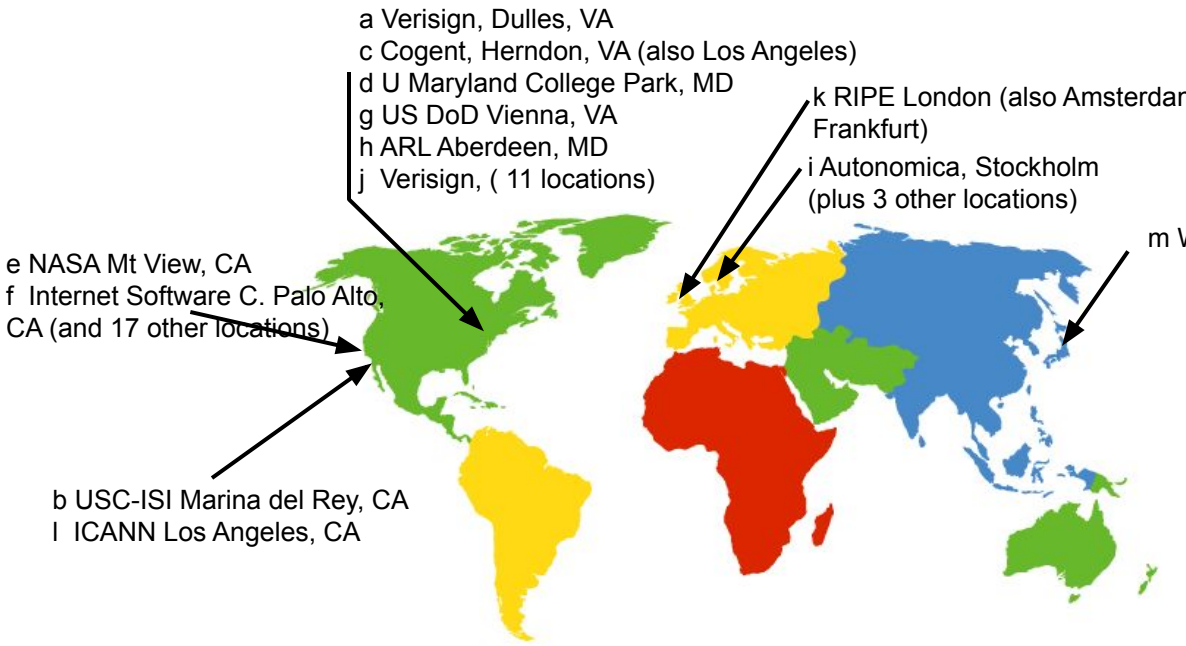


Cliente quer IP para www.amazon.com; 1ª aproximação:

- cliente faz pedido ao servidor raiz p/ encontrar o servidor DNS .com
- cliente faz pedido ao servidor DNS .com p/ obter o servidor DNS de amazon.com DNS
- cliente faz pedido ao servidor DNS de amazon.com para obter o endereço IP p/ www.amazon.com

DNS: Servidores raiz

- Procurado por servidor local que não consegue resolver o nome
- Servidor raiz:
  - procura servidor oficial se mapeamento desconhecido
  - obtém tradução
  - devolve mapeamento ao servidor local



ICANN (Internet Corporation for Assigned Names and Numbers) – ***gerencia os servidores raiz***

13 servidores raiz "lógicos" em todo o mundo (replicados várias vezes)

~200 USA

Servidores de Domínio de Alto Nível (TLD) e Oficiais

Servidores DNS de Domínio de Alto Nível:

- Responsáveis por .com, .org, .net, .edu, etc, e todos os domínios de alto nível de países: br, uk, fr, ca, jp.

  - Network Solutions mantém servidores de alto nível para .com

  - Educause mantém servidores de alto nível para .edu

Servidores DNS Oficiais:

  - Servidores DNS das organizações, que provêem mapeamentos (registros) oficiais de hostname para endereço IP para os servidores de Web e Email das mesmas.

  - Podem ser mantidos pela organização (grandes empresas e universidades, por ex.) ou por um provedor de serviços pago

Servidor de Nomes Local

- não necessariamente pertence à hierarquia de servidores
- cada provedor (residencial, de empresa, universidade) tem um.
  - também chamado de "servidor de nome default"
  - seu IP é fornecido ao host quando é feita a conexão ao provedor (vide "network status", no Windows ou Unix)

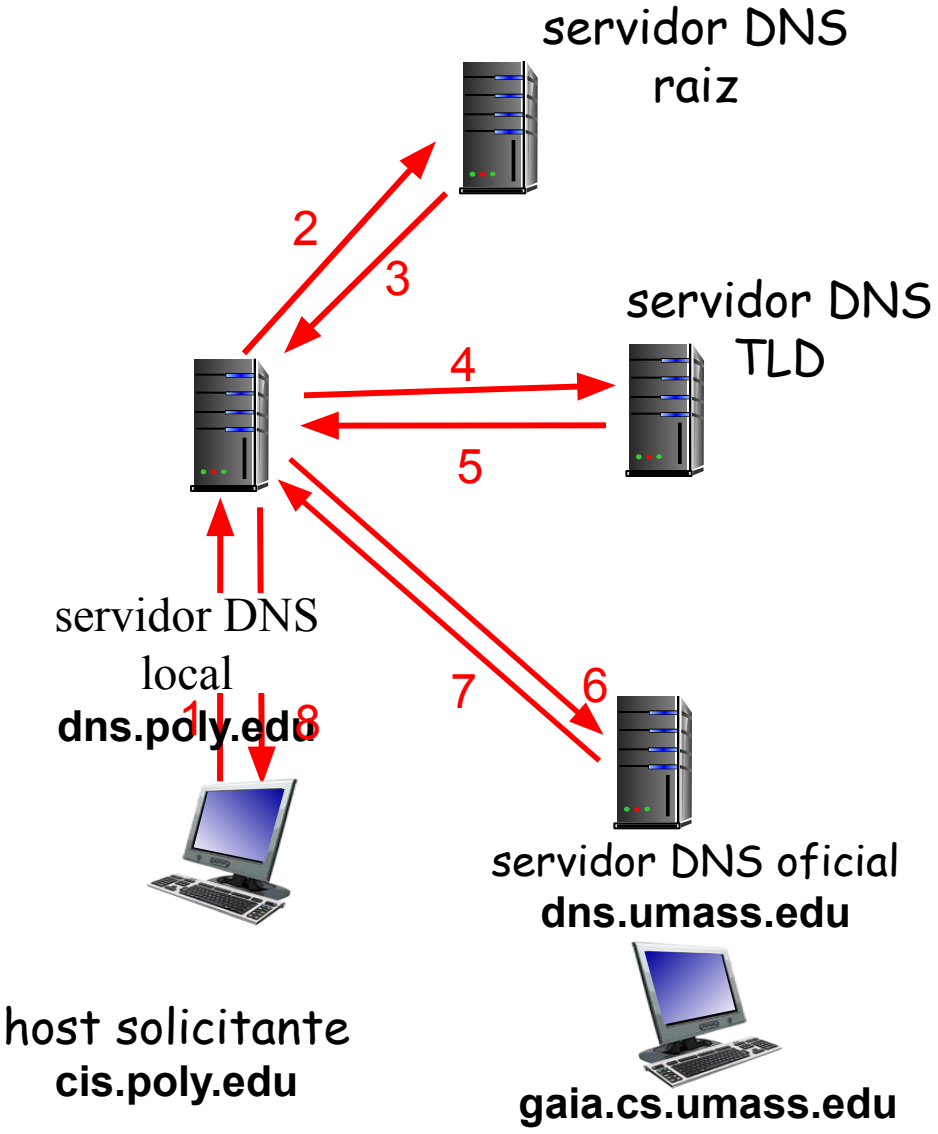
- Normalmente está próximo ao host
- quando um computador faz um pedido DNS, o pedido é enviada para o seu servidor DNS local
  - Possui *cache* local dos recentes pares de traduções nome- IP realizados (pode estar desatualizado!)
  - Age como um proxy, redirecionando pedidos que não conhece para a hierarquia de servidores DNS

Resolução de nomes: pedido iterativo

Host em cis.poly.edu quer o IP address p/ gaia.cs.umass.edu

Pedido iterativo:

- Servidor contatado responde c/ nome do servidor a ser contatado
- "Eu não sei esse nome, mas pergunte a este outro servidor"
- 8 mensagens trocadas!

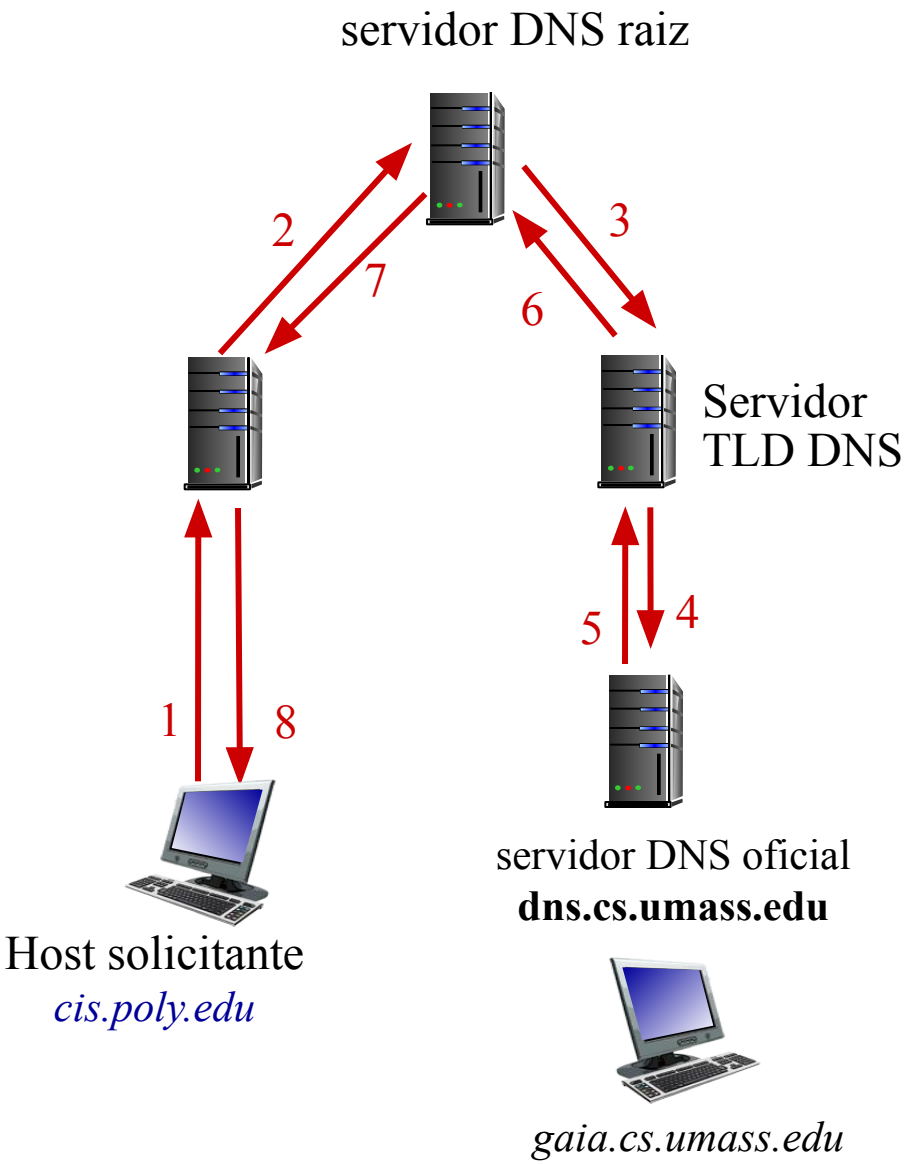


Resolução de nomes: pedido recursivo

host cis.poly.edu requer endereço IP de gaia.cs.umass.edu

Pedido recursivo:

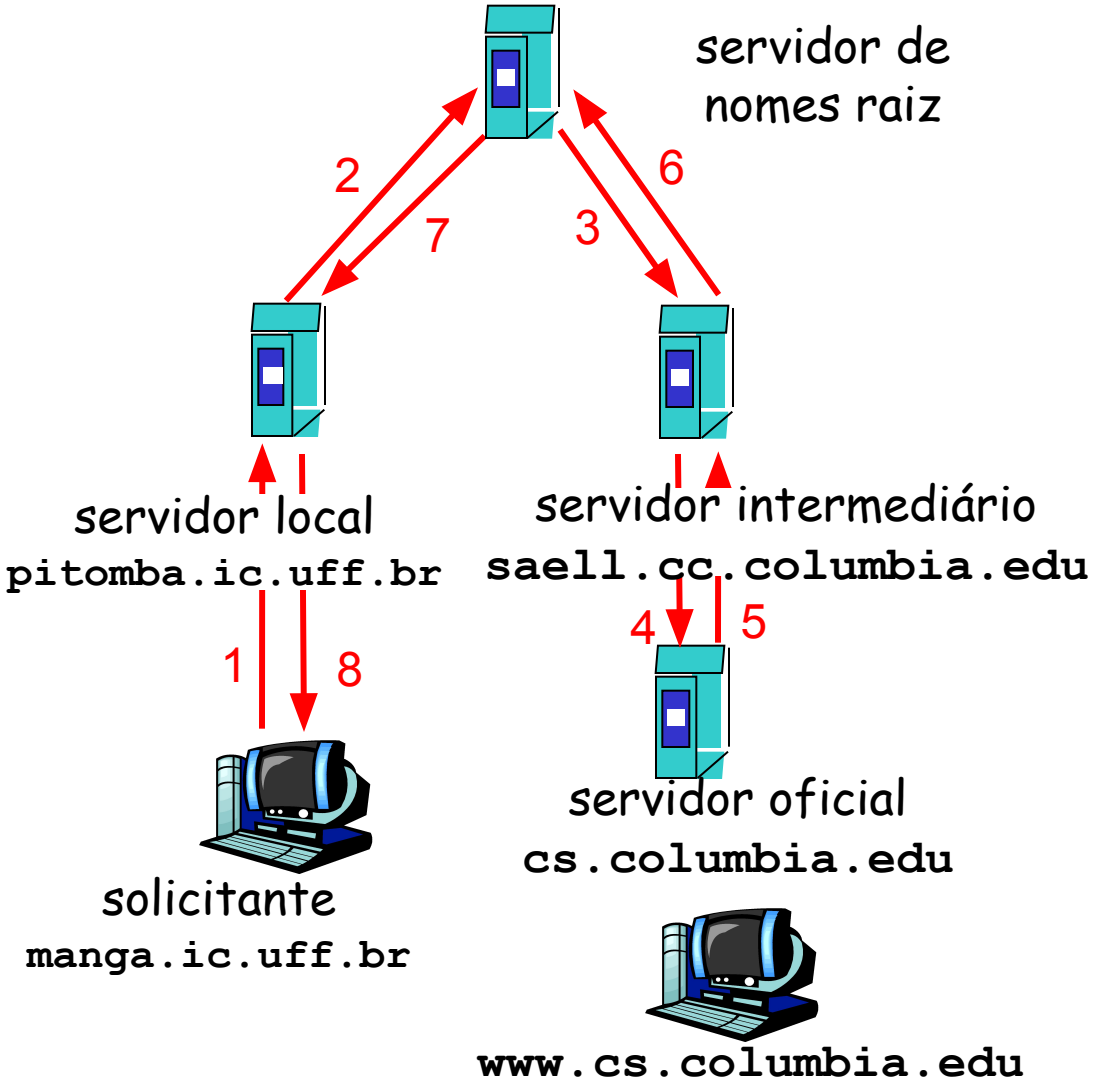
1. Contata servidor DNS local, **dns.poly.edu**
2. **dns.poly.edu** contata servidor raiz, se necessário
3. Servidor raiz contata servidor oficial, se necessário



Resolução de nomes: exemplo (3)

Servidor raiz/TLD:

- pode não conhecer o servidor de nomes oficial
- pode conhecer *servidor de nomes intermediário:* a quem contatar para descobrir o servidor de nomes oficial



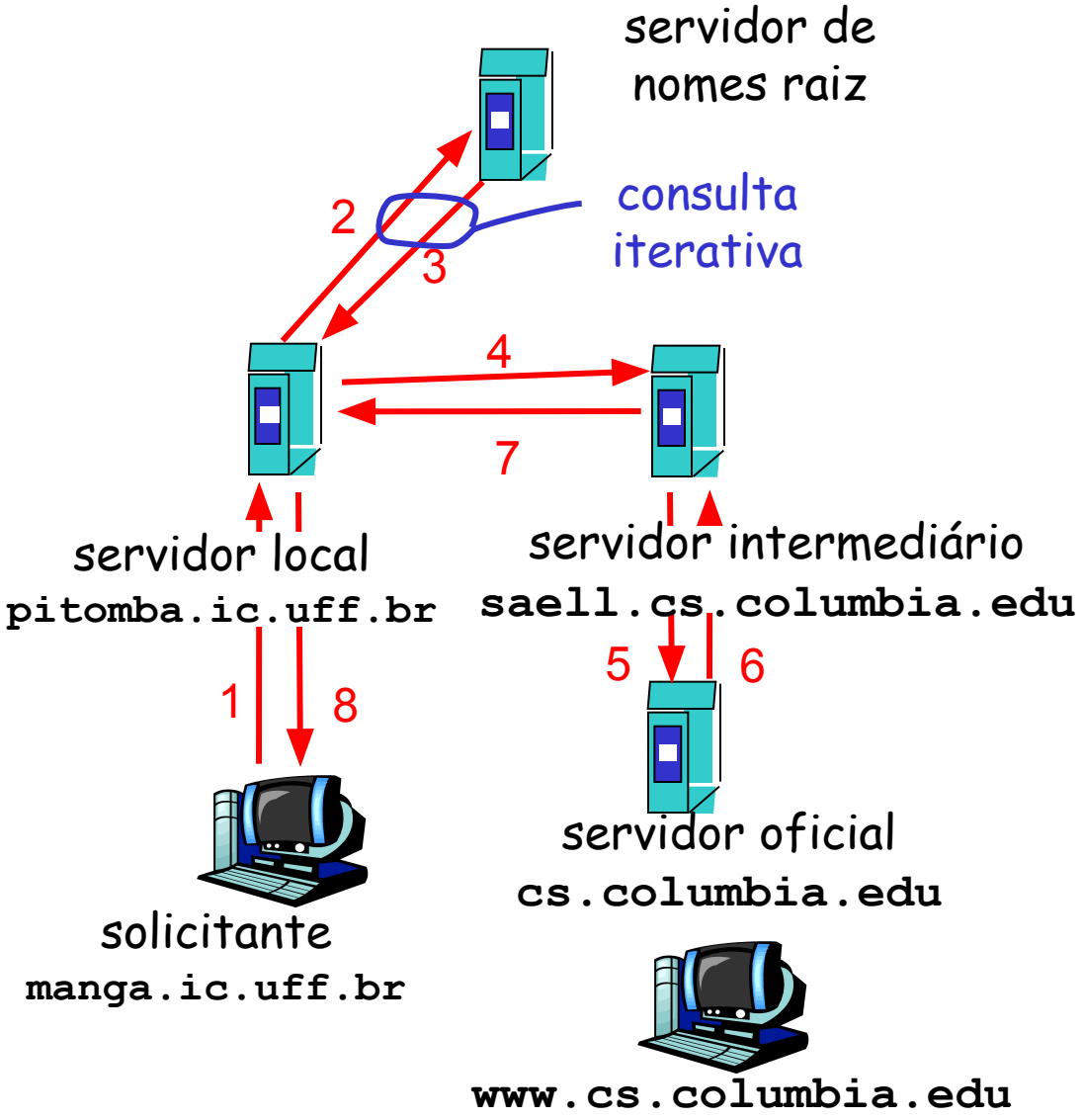
DNS: consultas iterativas e recursivas

consulta recursiva:

- transfere a responsabilidade de resolução do nome para o servidor de nomes contatado
- carga pesada nos serv de maior hierarquia?

consulta iterativa:

- servidor consultado responde com o nome de um servidor de contato
- "Não conheço este nome, mas pergunte para esse servidor"



DNS: uso de cache

1. Uma vez que um servidor qualquer aprende um mapeamento, ele o coloca numa cache local

- futuras consultas são resolvidas usando dados da cache

- entradas na cache são sujeitas a temporização (desaparecem depois de um certo tempo: tipicamente 2 dias)
  ttl = time to live (sobrevida)

DNS: uso de cache

2. IP dos servidores TLD normalmente são armazenados no cache do servidor local (portanto, servidores raiz não são visitados constantemente)

- Como as entradas no cache podem estar desatualizadas o endereço IP correto pode não ser conhecido até que o TTL expire

- O DNS explora extensivamente o *cache* para diminuir o atraso e reduzir o número de mensagens DNS que trafegam pela Internet.

Registros DNS

DNS: BD distribuído contendo *registros de recursos* (RR)

formato RR: `(nome, valor, tipo, ttl)`

- Tipo=*A*
  - **nome** é nome de host
  - **valor** é o seu endereço IP

Ex: (relay1.bar.foo.com, 145.37.93.126, A)

- Tipo=NS
  - **nome** é domínio (p.ex. foo.com)
  - Valor é o nome do servidor oficial (dns.foo.com) para este domínio

Ex: (foo.com, dns.foo.com, NS)

Qdo o servidor devolve um registro deste tipo, também devolve dentro da mensagem um registro tipo *A*

(dns.foo.com, 192.168.50.100, A)

Registros DNS

DNS: BD distribuído contendo *registros de recursos* (RR)

formato RR: **(nome, valor, tipo, ttl)**

- Tipo=CNAME
  - **nome** é o apelido (alias) para algum nome "canônico" (verdadeiro)
  - **valor** é o nome canônico.

  Ex1: www.ibm.com é, na verdade,um apelido para servereast.backup2.ibm.com

  Ex2: (foo.com, relay1.bar.foo.com, CNAME)

- Tipo=MX
  - **nome** é domínio
  - **valor** é nome canônico do servidor de correio para este domínio

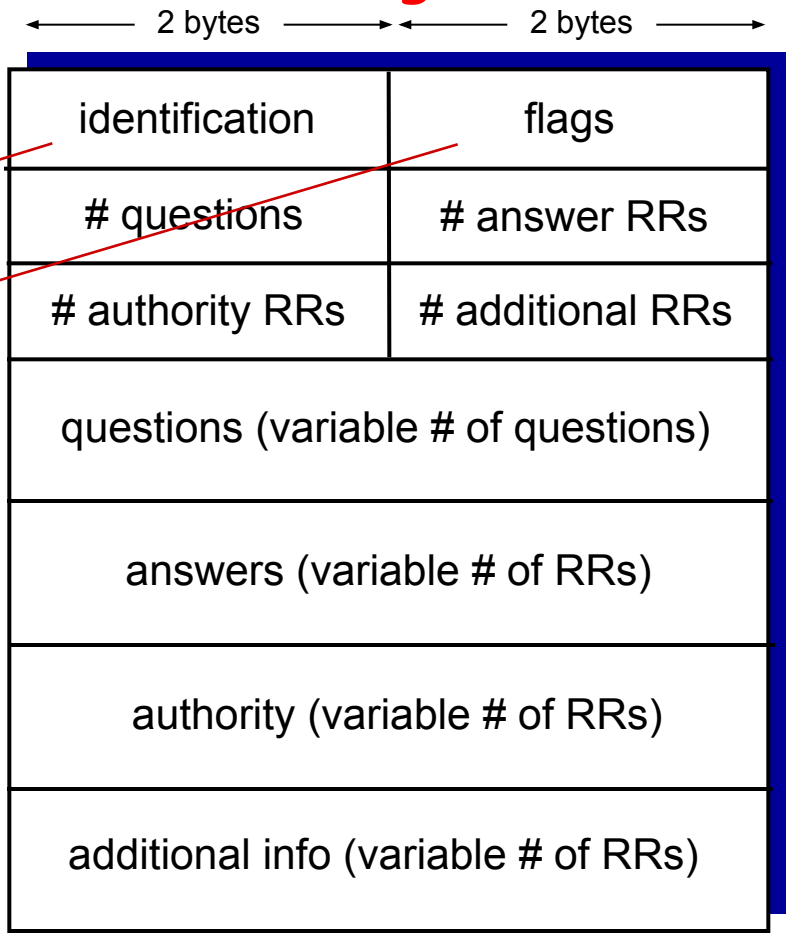
  Ex: (foo.com, mail.bar.foo.com, MX)

DNS: protocolo e mensagens

protocolo DNS: mensagens de *pedido* e *resposta*, ambas com o mesmo *formato de mensagem*

cabeçalho de msg

- identificação: ID de 16 bit para pedido. Resposta ao pedido usa mesmo ID
- flags:
  - pedido ou resposta
  - recursão desejada
  - recursão suportada
  - resposta é de um servidor oficial



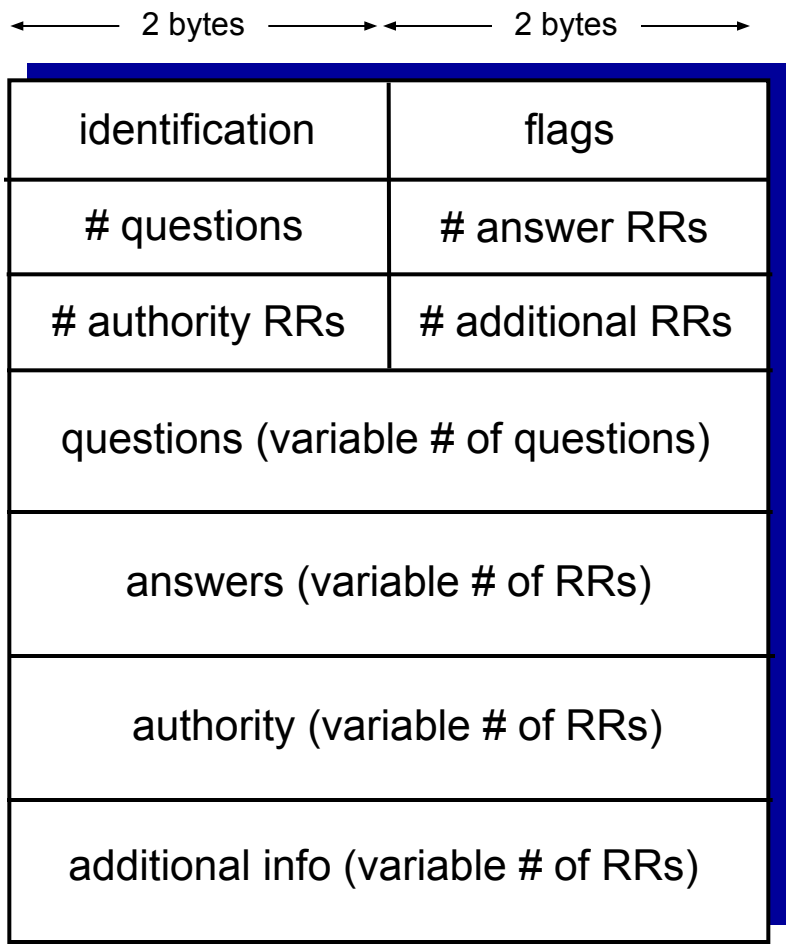
DNS: protocolo e mensagens

campos de nome e de tipo num pedido

RRs em resposta ao pedido

registros para outros servidores oficiais

informações adicionais



Programa ***nslookup*** permite que façamos consultas DNS

Inserindo registros no DNS

exemplo: novo site "Network Utopia"

- registrar nome networkuptopia.com em "*registrador DNS*" (por ex., Network Solutions: .com, .net e .org) – verifica a unicidade do domínio e o coloca no banco de dados
  - ✔ provê nomes e endereços IP de servidores oficiais (primário e secondário)
  - ✔ ***Network Solutions*** insere dois RRs no servidor TLD .com:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- deve-se criar também, no servidor oficial, um registro Tipo A p/ www.networkuptopia.com (servidor Web) e um registro Tipo MX p/ networkutopia.com (serv. Mail)

A partir daí pode-se acessar o servidor Web de Network Utopia.

Ataques ao DNS

Ataque DdoS (negação de serviço distribuída)

Bombardear os servidores raiz com excesso de tráfego

- Não é eficiente:
  - Sem sucesso até o momento
  - Filtragem de tráfego
  - Servidores DNS locais armazenam no cache os IPs dos servidores TLD, evitando acesso ao servidor raiz

Bombardear os servidores TLD: potencialmente mais perigoso

Ataque do homem do meio (man-in-middle): intercepta pedidos e retorna respostas falsas

DNS poisoning

✔ Enviar falsas traduções para o servidor DNS, que as coloca no cache

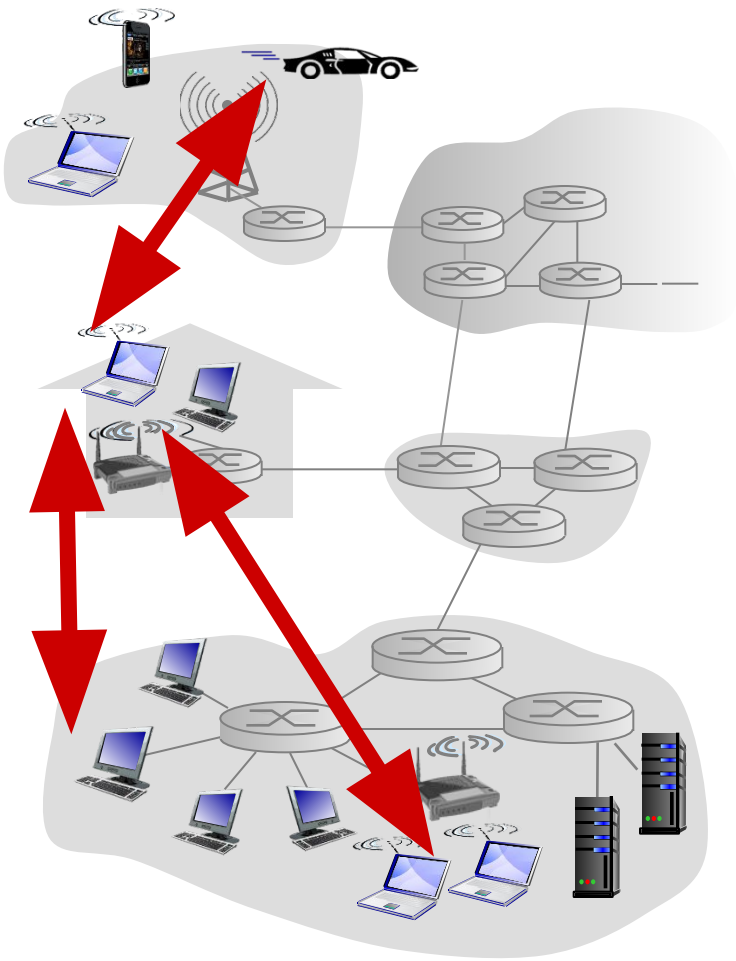
Explorar DNS com DDoS

✔Enviar pedidos com falso endereço IP de origem

DNSSEC (RFC 4033): versão Segura do DNS

Arquitetura P2P

- *Não existe servidor "sempre online"*
- Sistemas finais arbitrários se comunicam diretamente
- Peers (pares) requisitam serviço de outros peers e, por sua vez, provêem serviço a estes outros peers
  - ⇒ ***Autoescalabilidade:*** novos pares trazem novas demandas por serviço, mas tbm trazem nova capacidade de serviço
- Pares estão conectados intermitentemente e mudam seus endereços IP ⇒ ***gerenciamento complexo***



- ***Aplicações:*** distribuição de arquivos (BitTorrent), ***video streaming*** (Kankan), VoIP (Skype)

Aplicações P2P

Distribuição de arquivos P2P

- **Cada par pode redistribuir qualquer parte do arquivo recebido para outros pares, auxiliando, assim, o servidor no processo de distribuição.**

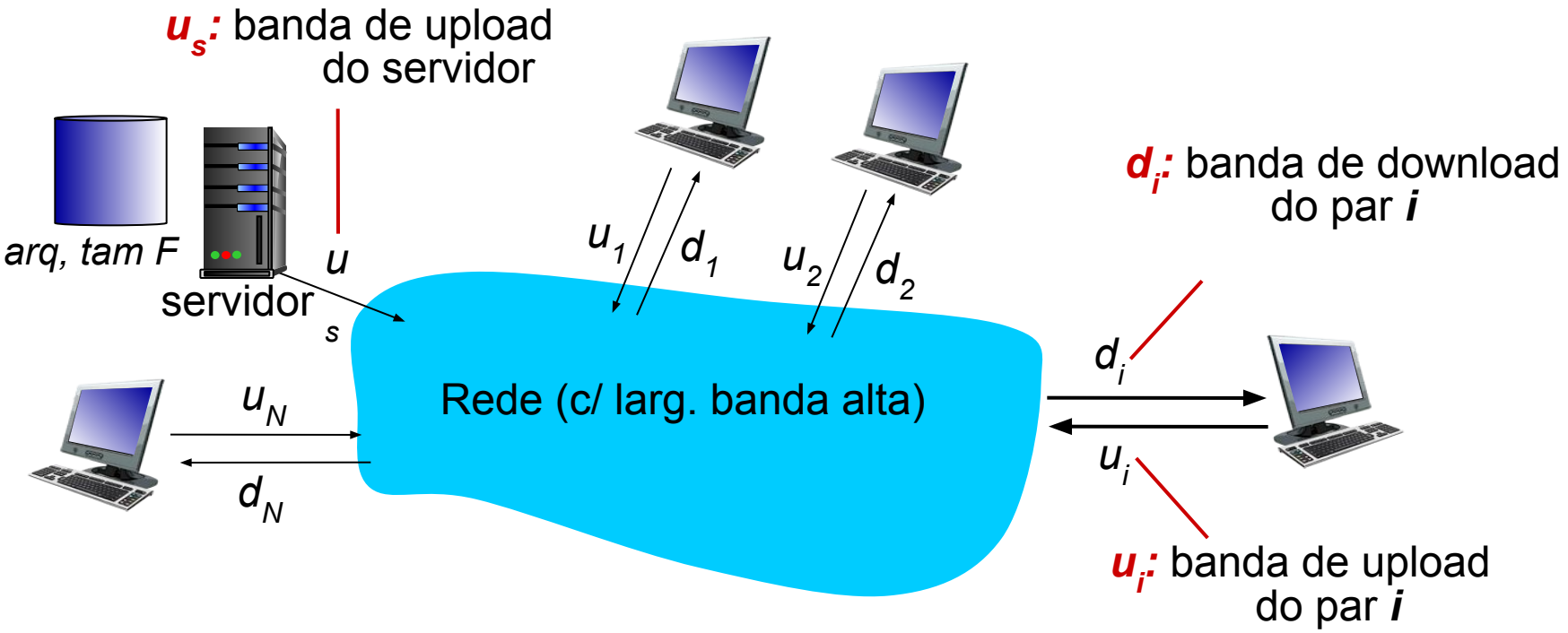
- **O tempo de distribuição é o tempo necessário para que todos os *N* pares obtenham uma cópia do arquivo.**

Distribuição de arquivos: C-S vs P2P

Questão:

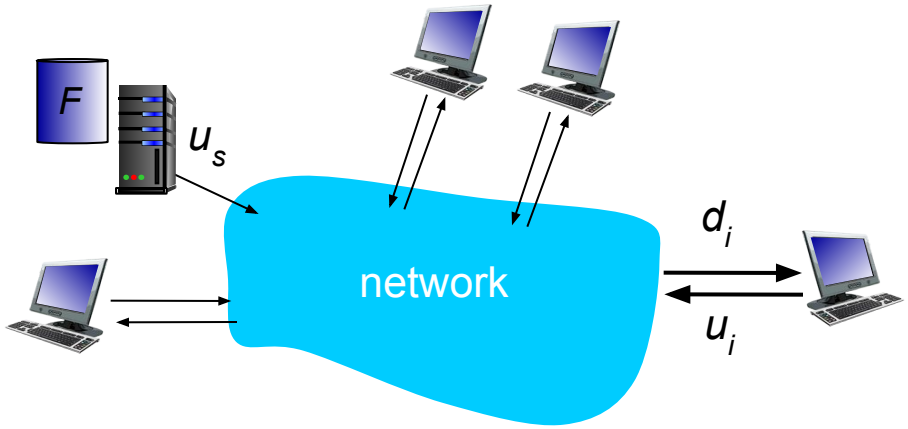
Quanto tempo é necessário para distribuir um arquivo de tamanho *F* de um servidor para *N* pares?

- Capacidade de upload/download do par é um recurso limitado



Tempo de distribuição do arquivo: modelo C-S

- *Transmissão do arquivo pelo servidor:* deve enviar (upload) N cópias do arquivo:
  - tempo p/ enviar uma cópia: F/u_s
  - tempo para enviar N cópias: NF/u_s



- *cliente:* *cada* cliente deve baixar uma cópia
  - d_{min} = taxa min de download entre os clientes
  - Tempo de download para o usuário com menor taxa: F/d_{min}

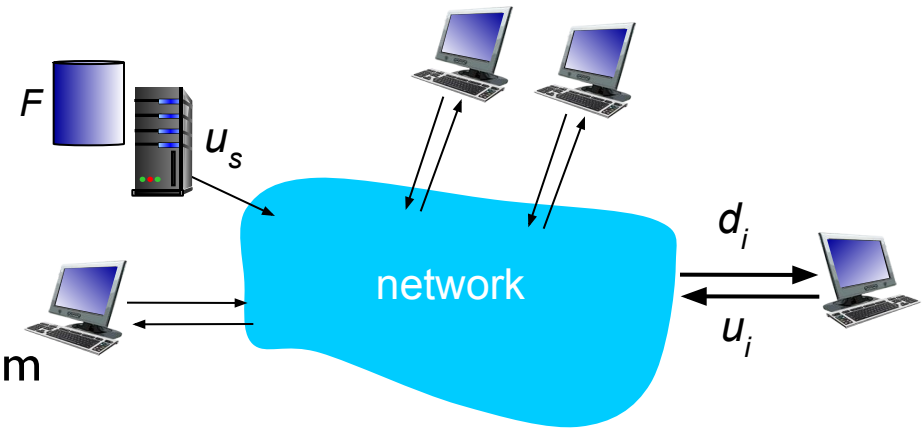
tempo p/ distribuir F p/ N clientes usando o modelo cliente-servidor

$$T_{c\text{-}s} > max\{NF/u_s, F/d_{min}\}$$

Aumenta linearmente c/ N

Tempo de distribuição do arquivo: modelo P2P

- *servidor:* *deve fazer* upload de uma cópia (*F* bits)
  - Tempo p/ enviar uma cópia: F/u_s
- *cliente:* cada cliente deve fazer download de uma cópia
  - Tempo de download para usuário com menor taxa: F/d_{min}



- *clientes:* devem fazer download de *NF* bits (total)

Taxa max de upload da rede (limitando a taxa max de download) é $u_s + \Sigma u_i$

tempo p/ distribuir F p/ N clientes usando o modelo P2P

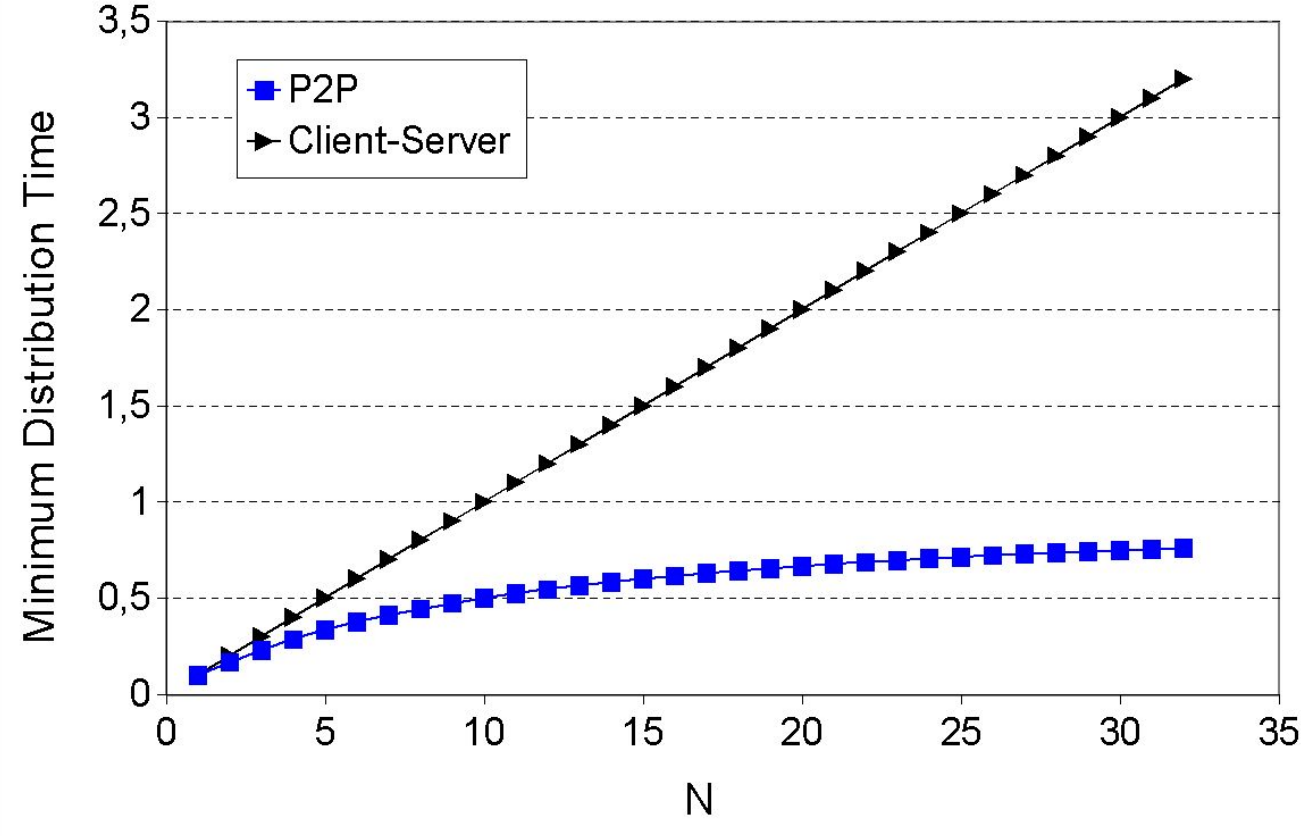
$$T_{P2P} > \underline{m}ax\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

Aumenta linearmente c/ *N* …

… no entanto, cada par traz capacidade de serviço

Cliente-servidor x P2P: um exemplo

Taxa de upload do cliente = u, F/u = 1 hora, $u_s = 10u$, $d_{min} \geq u_s$

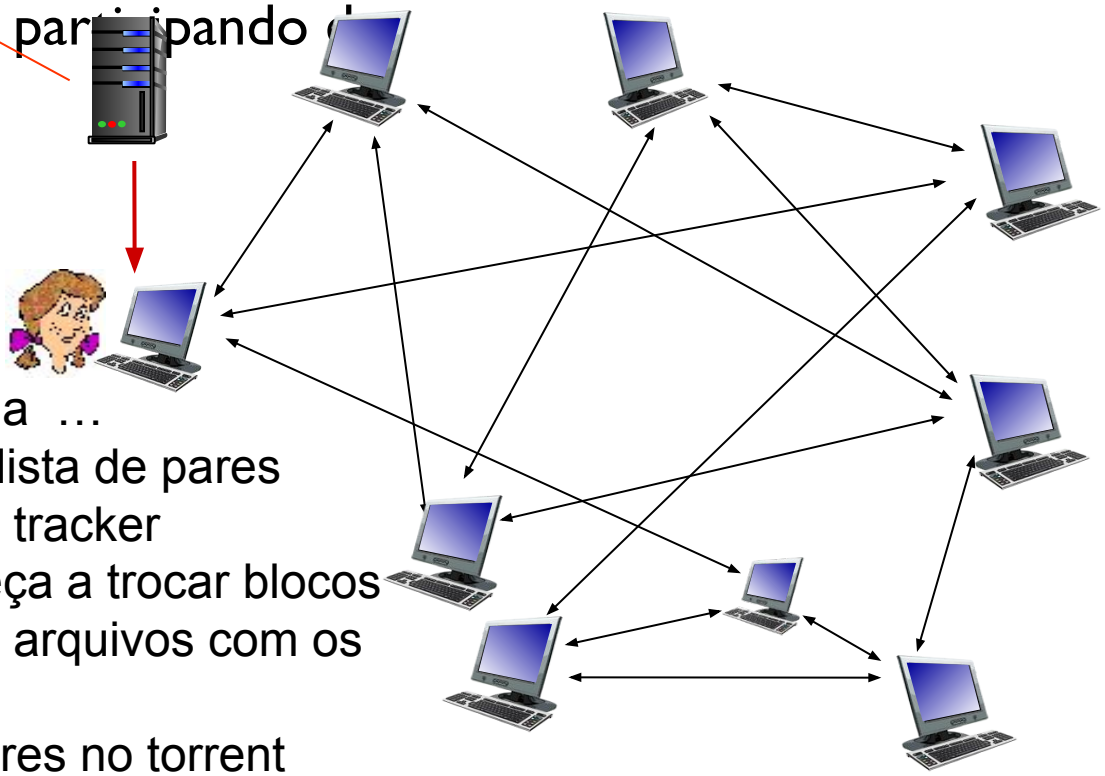


BitTorrent: distribuição de arquivos P2P

- arquivo dividido em blocos de 256 KB
- pares no *torrent* enviam/recebem *blocos do arquivo*

tracker: mantém registro dos pares participando do torrent

torrent: grupo de pares trocando *blocos de arq.*



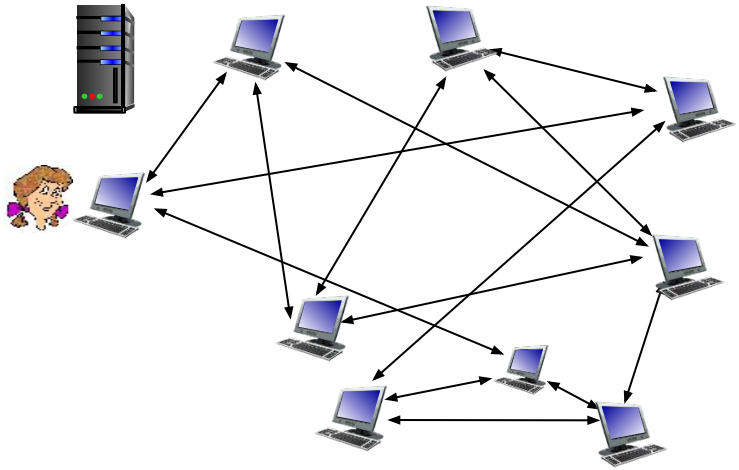
Alice chega …

… obtém lista de pares do tracker

… e começa a trocar blocos de arquivos com os outros pares no torrent

BitTorrent: distribuição de arquivos P2P

- Par chegando ao torrent:
  - Não tem nenhum bloco, mas vai obtê-los à medida que se comunica com outros pares
  - Se registra no tracker para obter lista de pares, se conecta a um subconjunto (normalmente até 50 pares, denominados "vizinhos")



- Enquanto faz download, par faz upload de blocos p/ outros pares
- Um par pode mudar os pares com os quais troca blocos
- Pares podem entrar e sair
- Quando o par já obteve todo o arquivo, ele pode ser egoísta, deixando o torrent, ou altruísta, permanecendo no torrent

BitTorrent: pedindo e enviando blocos de arquivos

Obtendo blocos:

- Num determinado instante, pares diferentes possuem diferentes subconjuntos de blocos do arquivo
- Periodicamente, um par (Alice) solicita a cada par vizinho a lista de blocos que eles têm…
- …e pede a eles os pedaços dos quais ela precisa, os mais raros primeiramente

Enviando blocos: ***tit-for-tat*** *(toma lá, dá cá!)*

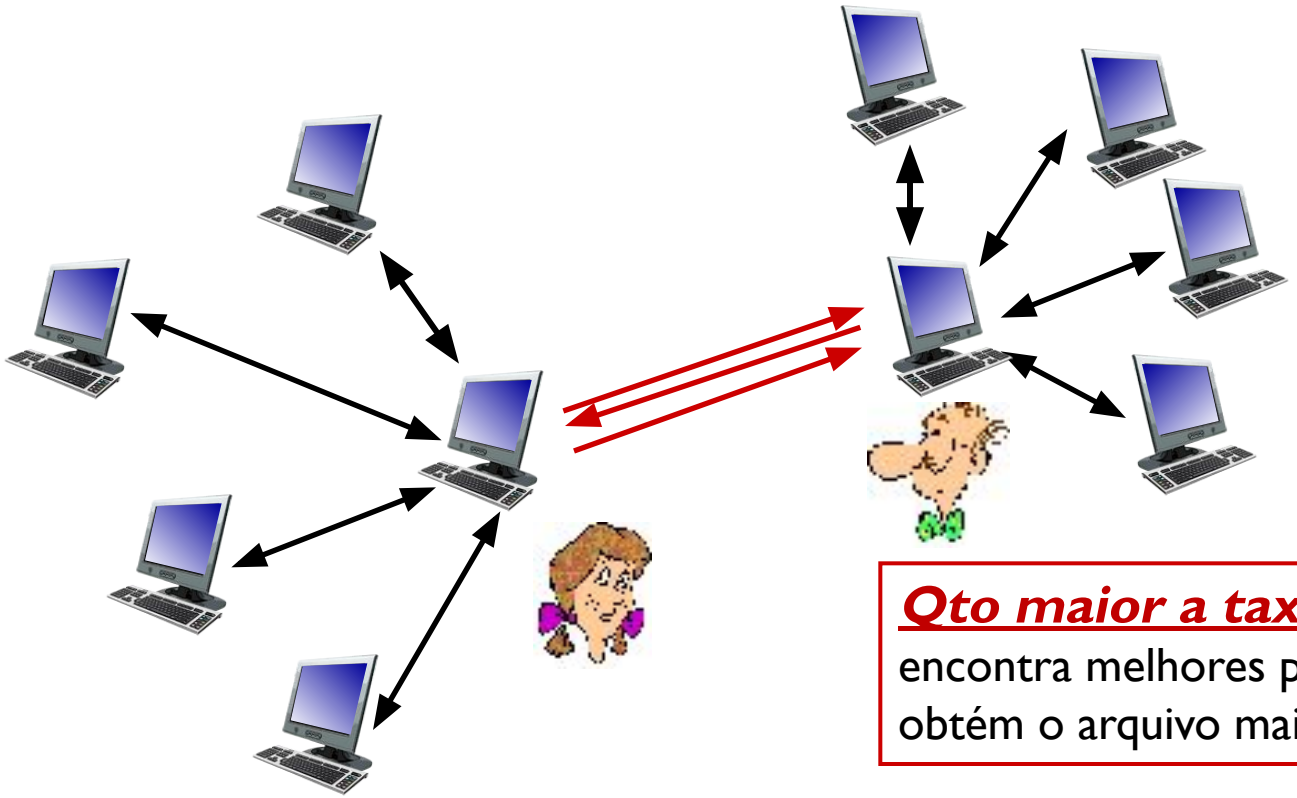
- Alice envia blocos para os quatro pares vizinhos (“top 4”) que estão lhe enviando blocos à taxa mais alta
  - Outros pares são bloqueados por Alice (não recebem blocos dela)
  - Reavalia os top 4 a cada 10 segs
- A cada 30 segs: seleciona aleatoriamente um outro par e começa a enviar-lhe blocos
  - Provavelmente irá desbloquear esse par
  - Talvez o par recém escolhido pode se juntar aos top 4

BitTorrent: *tit-for-tat*

(1) Alice desbloqueia Bob

(2) Alice se torna uma das quatro melhores provedoras ("top-4") de Bob; Bob retribui, usando o princípio da reciprocidade (***tit-for-tat***)

(3) Bob se torna um dos quatro melhores provedores ("top-4 ") de Alice



Qto maior a taxa de upload: encontra melhores parceiros e obtém o arquivo mais rapidamente