

Aula 7 - DNS, P2P

Diego Passos

Universidade Federal Fluminense

Redes de Computadores I

Material adaptado a partir dos slides
originais de J.F Kurose and K.W. Ross.

Revisão da Última Aula...

- **Cookies:**

- Artificio para armazenar estado.
- **Pequenos** pedaços de informação **pelo cliente**.
- Servidor pede ao cliente que armazene cookie.
- Cliente envia cookies em todas as requisições para o servidor.
- Normalmente, cookie guarda ID do usuário.

- **Web caches:**

- Cache para conteúdos web.
- Evita requisição para fora da rede.
- Reduz uso do enlace de acesso.
- Reduz atraso na resposta à requisição.
- Pode ocasionar **inconsistência dos dados**.

- **FTP:** transferência de arquivos.

- **Comunicação fora-de-banda.**

- i.e., controle separado dos dados.

- **E-mail:**

- Componentes: *user agent*, servidor, SMTP.
- SMTP: protocolo de envio de e-mail.
 - Cliente para servidor e servidor para servidor.
 - Mensagens ASCII.
 - Roda sobre TCP, porta 25.

- **POP3 e IMAP.**

- Acesso, gerenciamento da caixa de entrada.
- POP3 é mais simples, IMAP mais complexo.

DNS

DNS: Domain Name System

- **Pessoas:** muitos identificadores.
 - CPF, RG, # de passaporte, ...
- **Hosts e roteadores na Internet:**
 - Endereço IP (# de 32 bits) usado para endereçar datagramas.
 - “Nome”, e.g., `www.yahoo.com`, usado por humanos.
- **Pergunta:** como mapear nomes para seus respectivos IPs e vice-versa?
- **Domain Name System (DNS):**
 - **Base de dados distribuída:** implementada em uma hierarquia de múltiplos **servidores de nomes**.
 - **Protocolo da camada de aplicação:** hosts e servidores de nome se comunicam para **resolver** nomes (tradução entre nome e endereço IP).
 - Note: função fundamental da Internet implementada como protocolo de aplicação.
 - Complexidade nas bordas.

DNS: Serviços, Estrutura

- **Serviços do DNS:**

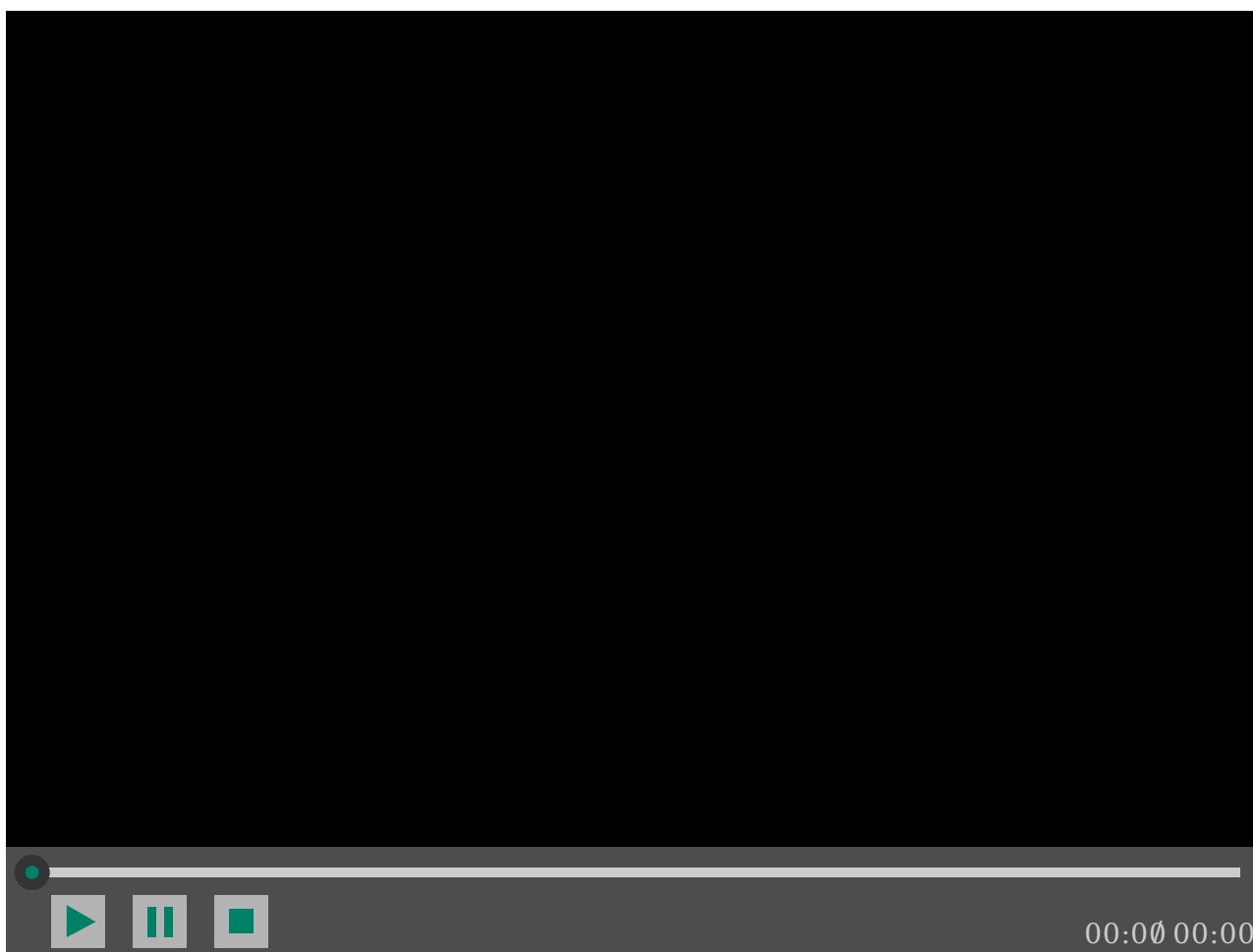
- Tradução de nomes de *hosts* para endereços IP.
- *Host aliasing*.
 - Atribuição de “apelidos”.
 - Host possui **nome canônico** e, possivelmente, vários apelidos.
- *Aliasing* de servidores de e-mail.
- Balanceamento de carga.
 - Servidores web replicados.
 - Cada servidor com seu IP.
 - Mas o mesmo nome associado a vários IPs.

- **Por que não um DNS centralizado?**

- Ponto único de falha.
- Concentração de grande volume de tráfego.
- Base de dados centralizada distante.
- Manutenção.

Em suma: **não escala!**

DNS: Balanceamento de Carga



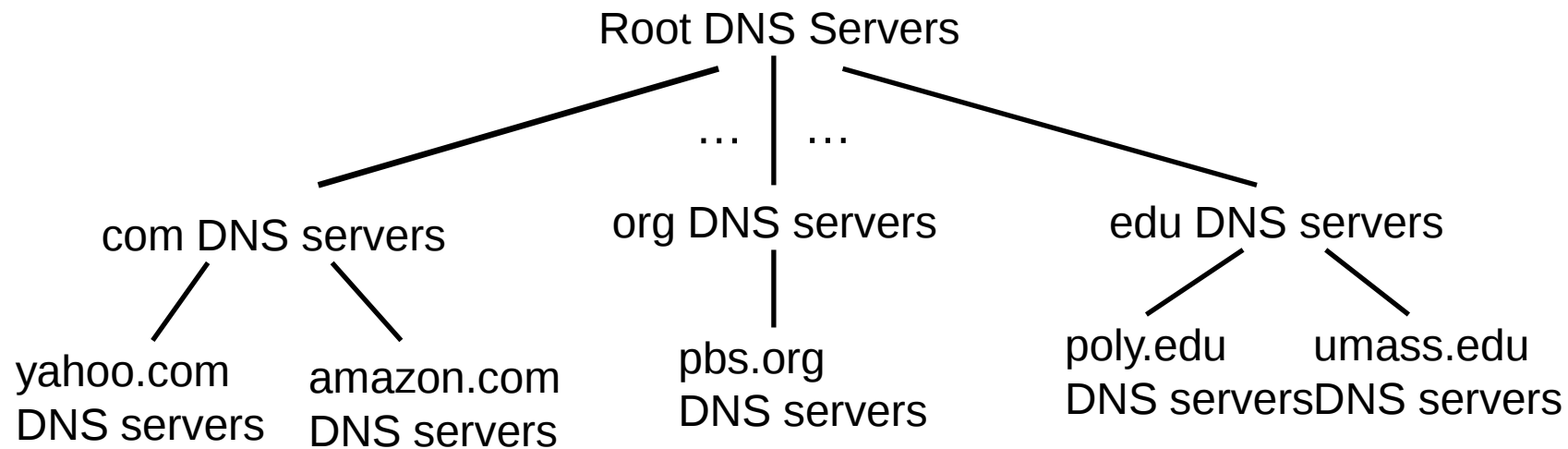
- Aplicação (ping) requisita resolução de nome `www.google.com`:
 - Primeiras execuções associam nome a `216.58.222.100`.
 - Eventualmente, respostas diferentes: `216.58.222.68`.

Alias (Apelidos)



- Primeira resolução para o nome `www.midiacom.uff.br`:
 - Nome associado ao IP `200.20.10.93`.
- Segunda resolução para `mesbla.midiacom.uff.br`:
 - Associado ao mesmo IP!.

DNS: Uma Base de Dados Hierárquica e Distribuída

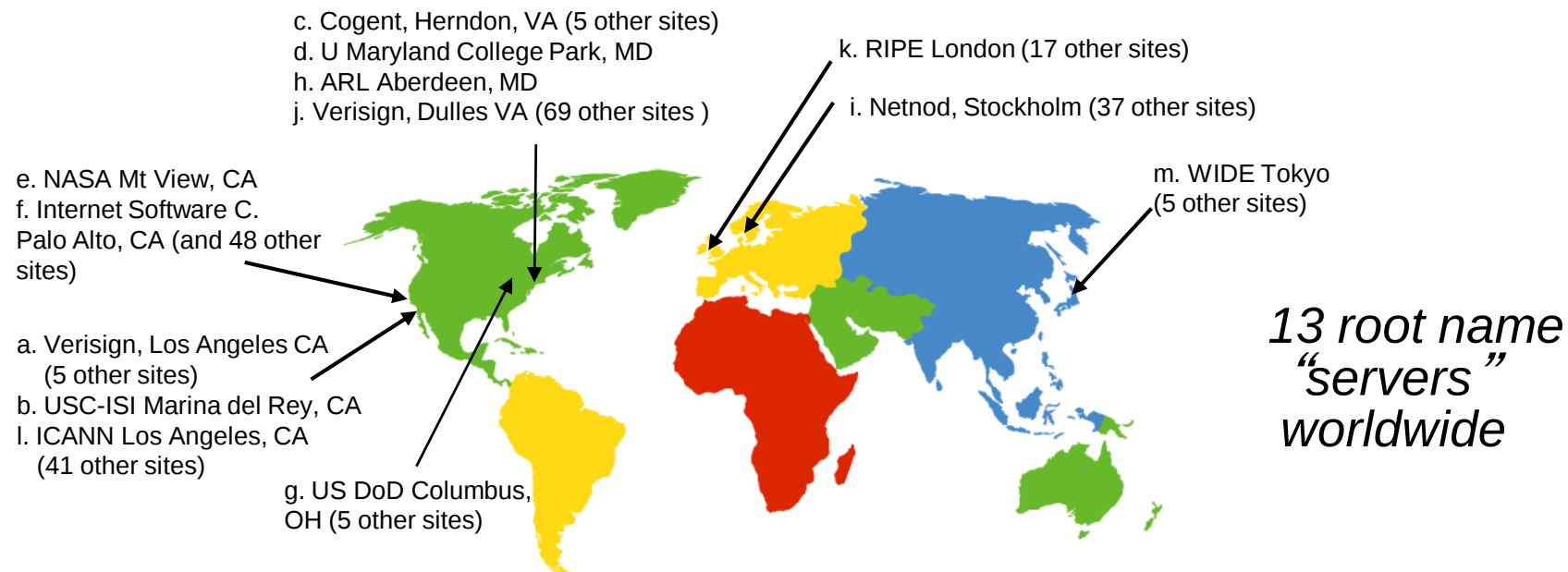


- **Cliente quer IP de `www.amazon.com`. Primeira abordagem:**

- Cliente pergunta ao servidor raiz a localização do DNS do **domínio** .com.
- Cliente pergunta ao servidor DNS do domínio .com a localização do servidor DNS do domínio `amazon.com`.
- Cliente pergunta ao servidor DNS do domínio `amazon.com` pelo endereço IP do host `www.amazon.com`.

DNS: Servidores Raiz

- Conhecem os servidores TLD.
- Contactados (principalmente) quando se deseja saber o DNS de um TLD.
- Poucos “servidores” no mundo.
 - Embora cada um seja composto por vários computadores espalhados pelo mundo.



TLD, Servidores Autoritativos

- **Servidores Top-Level Domain (TLD):**

- TLD: .org, .net, .com, .edu, ..., .br, .uk, .jp, ...
- Cada TLD tem seu servidor DNS específico.
- A Network Solutions mantém servidores DNS para o TLD .com.
- A Registro.br mantém o DNS para o TLD .br.

- **Servidores autoritativos:**

- Servidor de DNS de uma organização específica.
- Provê mapeamentos para os endereços IP da organização e seus nomes de *host*.
- Pode ser gerenciado pela própria organização ou por um provedor de serviço.



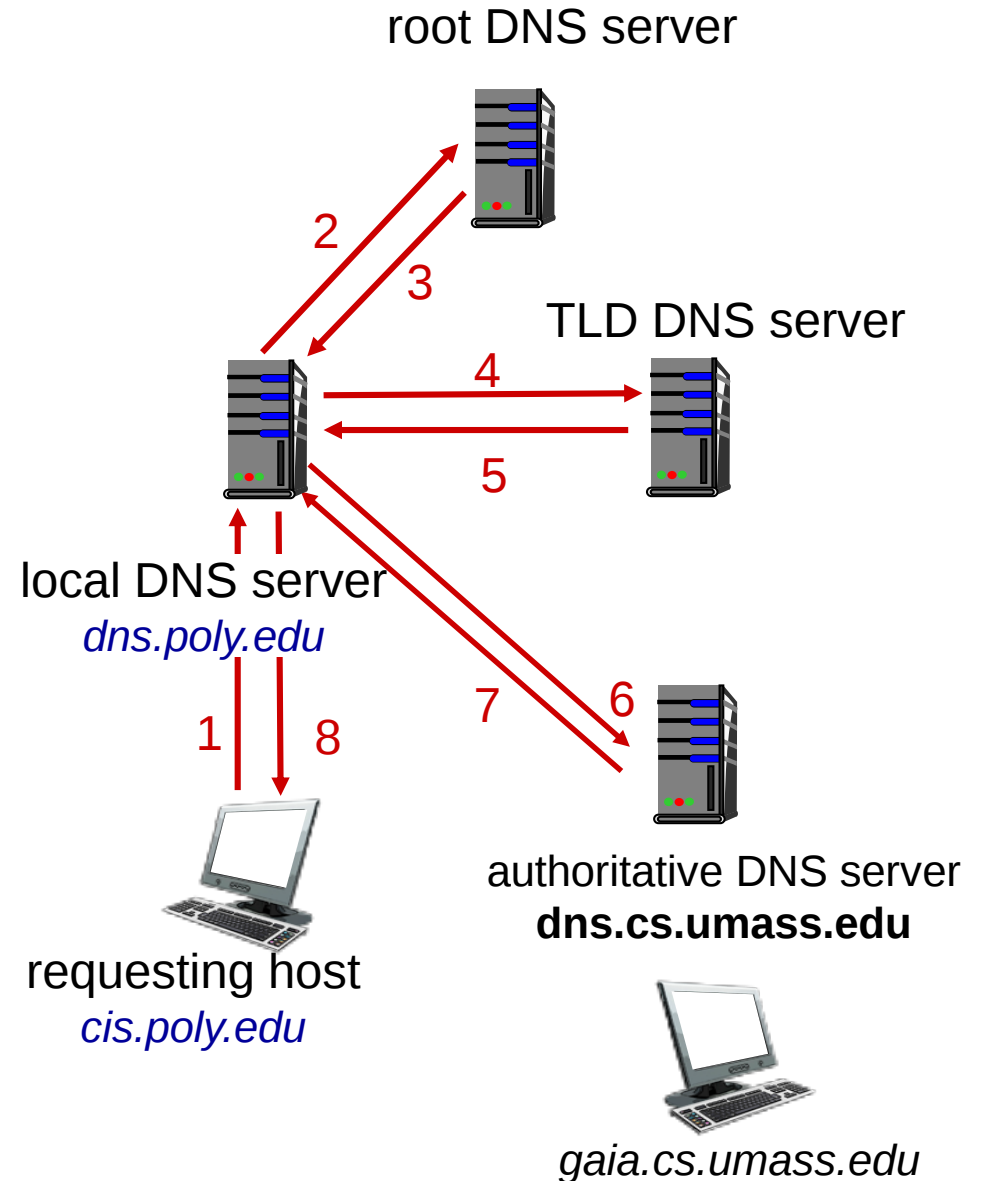
- Primeira resolução para o nome `www.google.com`:
 - Como esperado, bem sucedido.
 - TLD `.com`
- Segunda resolução para `com.google`:
 - Nome parece invertido: primeiro TLD, depois nome da organização.
 - Mas resolução é bem sucedida de qualquer forma!
 - Por que?
 - Atualmente, `.google` é um TLD.
 - Dentro deste TLD, há um *hostname* chamado `com`.

Servidores DNS Locais

- Estritamente falando, não fazem parte da hierarquia.
- Cada ISP (residencial, empresas, universidades) normalmente tem um.
 - Também chamado de “DNS padrão”.
- Quando o *host* faz uma requisição DNS, esta é enviada para o seu servidor DNS local.
 - Geralmente, possui um cache para mapeamentos realizados recentemente (mas o mapeamento pode não ser mais válido!).
 - Atua como um *proxy*, encaminhando requisições para a hierarquia.

Exemplo de Resolução de Nome Usando DNS (I)

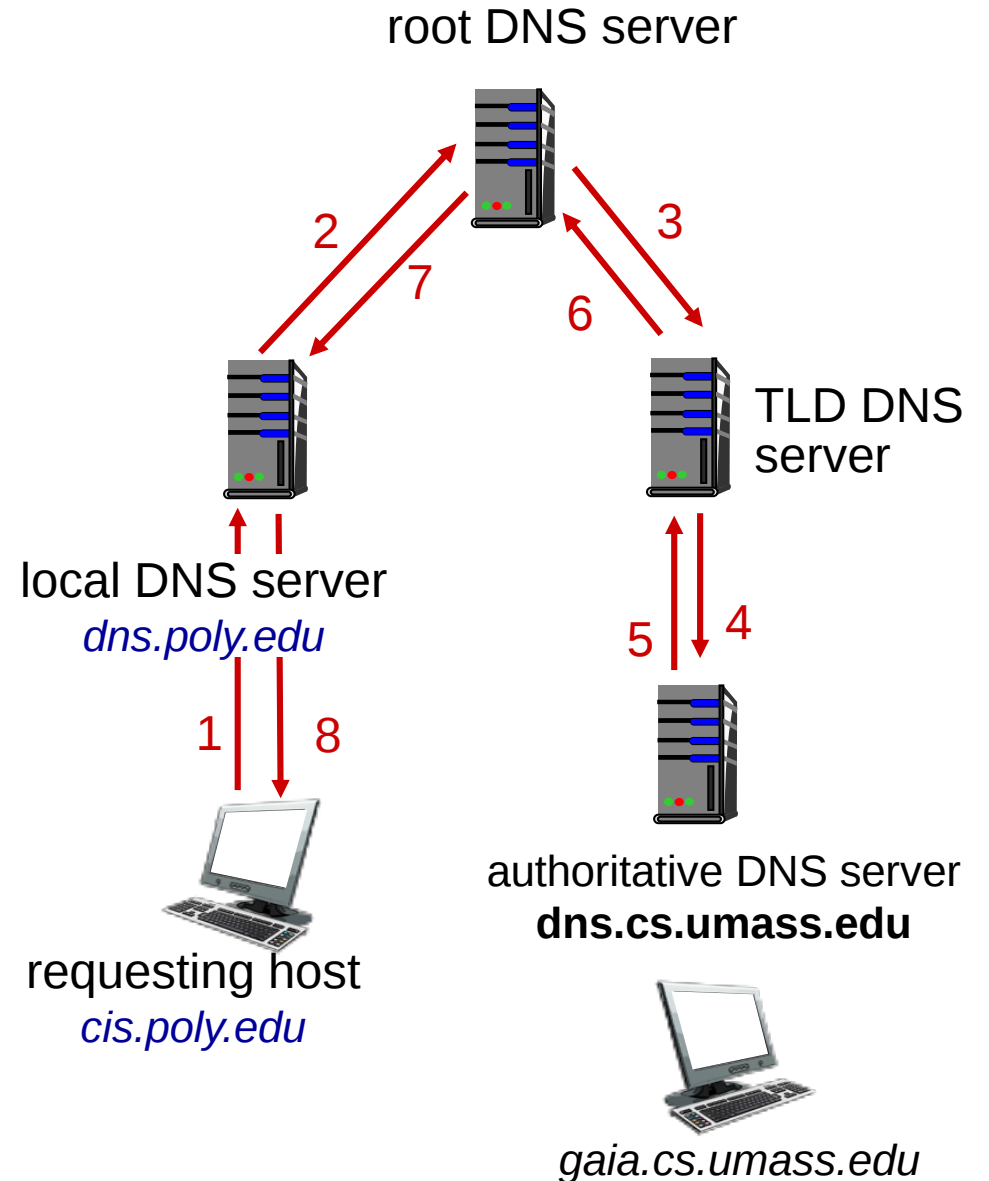
- Host em `cis.poly.edu` quer o IP de `gaia.cs.umass.edu`.
- **Método iterativo:**
 - Servidor contactado retorna nome de outro servidor a ser contactado.
 - “Eu não conheço este nome, mas pergunte para este outro servidor”.



Exemplo de Resolução de Nome Usando DNS (II)

- **Modo recursivo:**

- Coloca o fardo da resolução do nome no servidor contactado.
- Alta carga nos níveis mais altos da hierarquia?



DNS: Cache, Atualização de Registros

- Uma vez aprendido um mapeamento, um servidor de nomes (qualquer) **o armazena em cache.**
 - Entradas na cache têm uma data de expiração (TTL).
 - i.e., são jogadas fora depois de algum tempo.
 - Servidores TLD tipicamente presentes na cache.
 - Logo, servidores raiz raramente visitados.
- Entradas na cache podem ficar **desatualizadas.**
 - Serviço de tradução de melhor esforço!
 - Se host tem IP alterado, restante da Internet pode não ficar sabendo até que TTLs expirem.
- Há propostas para mecanismos de atualização/notificação.
 - e.g., RFC 2136.

Registros de DNS

- **DNS:** base de dados distribuída que armazena Resource Records (**RR**).

Formato de um RR: (nome, valor, tipo, TTL)

- **Tipo=A**

- **nome** é um nome de um *host*.
- **valor** é o endereço IP.

- **Tipo=NS**

- **nome** é um domínio (*e.g.*, `foo.com`).
- **valor** é o **nome do host** do servidor DNS autoritativo para este domínio.

- **Tipo=CNAME**

- **nome** é um apelido para um *host*.
- **value** é o **nome canônico**.
- *e.g.*, `www.midiacom.uff.br` é um apelido para `mesbla.midiacom.uff.br`.

- **Tipo=MX**

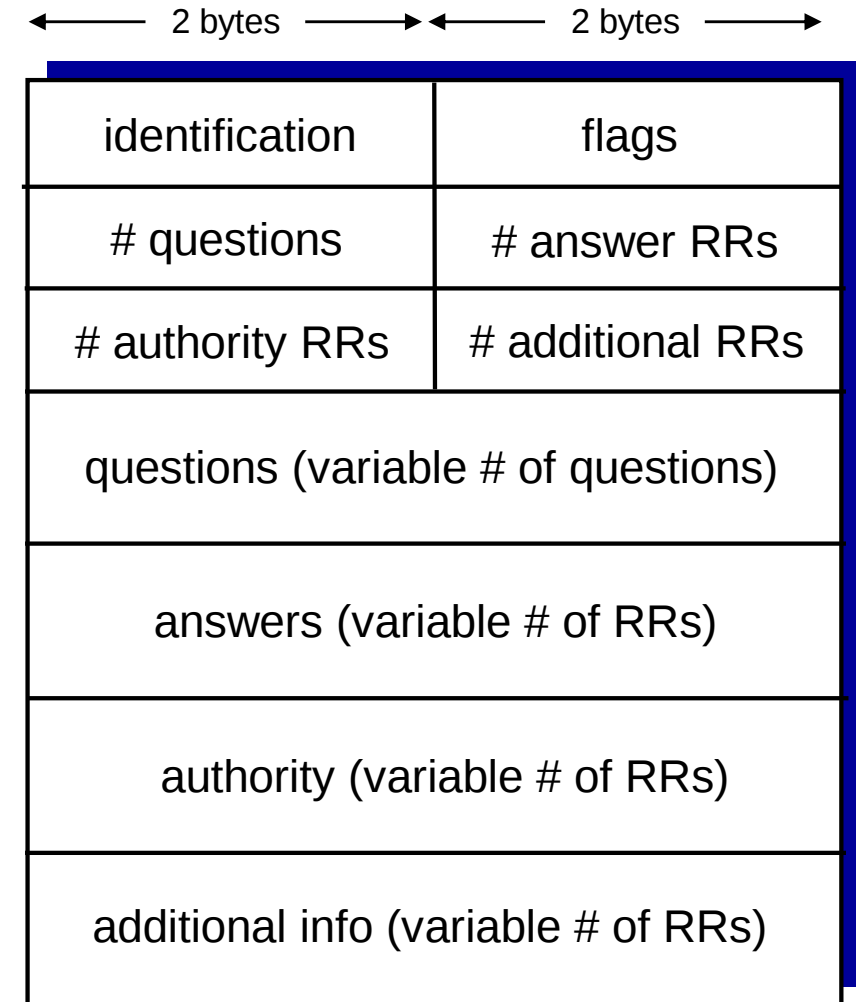
- **valor** é o **nome do host** que funciona como servidor de e-mail do domínio associado ao **nome**.

DNS: Protocolo e Mensagens (I)

- Mensagens de **requisição** e **resposta** têm o mesmo **formato**.

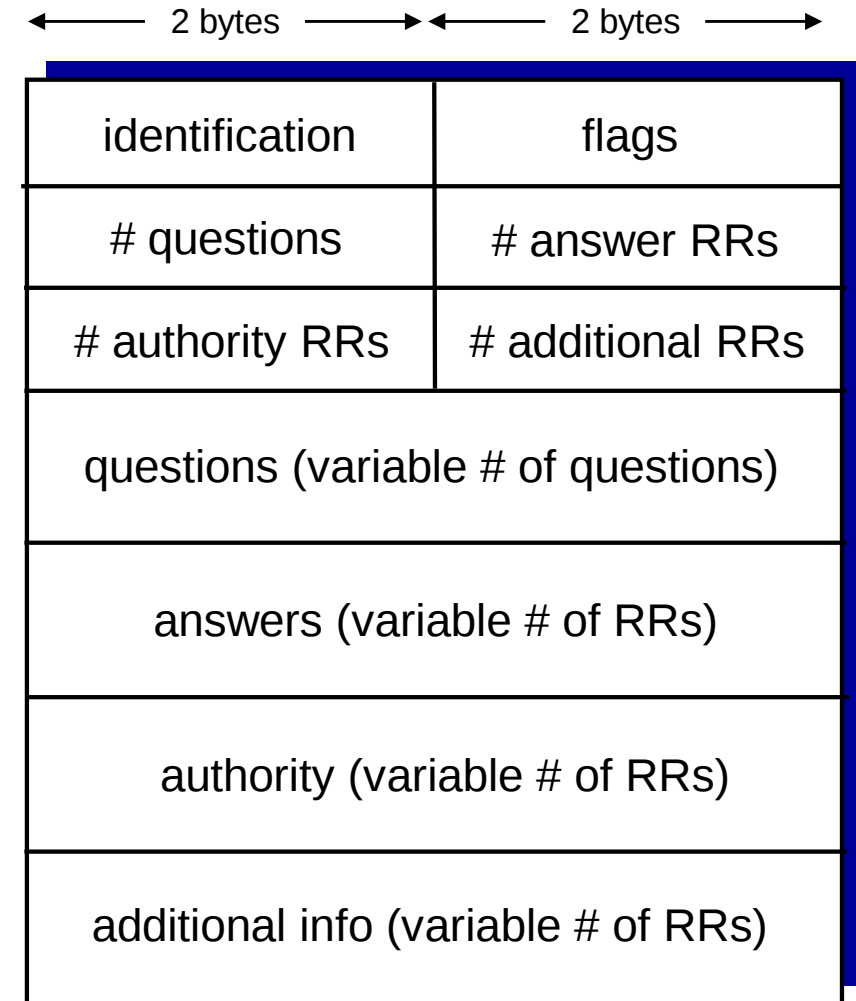
- Cabeçalho das mensagens:

- **Identificação:** # de 16 bits para requisição; resposta utiliza mesmo # da requisição a que responde.
- **Flags:**
 - Requisição ou resposta.
 - Modo recursivo é desejado.
 - Modo recursivo está disponível.
 - Resposta é autoritativa.



DNS: Protocolo e Mensagens (II)

- Campo das consultas:
 - Múltiplas consultas possíveis em uma mesma requisição.
 - Informa nomes, tipos dos campos nas requisições.
- Campo das respostas:
 - Múltiplas respostas possíveis em uma mesma mensagem.



Inserindo Registros no DNS

- Exemplo: nova empresa chamada “Network Utopia”.
- Registro do domínio networkutopia.com com a entidade de registro de nomes.
 - e.g., Network Solutions.
 - Necessário prover nomes e IPs dos servidores de nome autoritativos do novo domínio (primário e secundário).
 - Entidade de registro insere dois RRs na base do servidor de DNS TLD .com:
 - (networkutopia.com, dns1.networkutopia.com, NS).
 - (dns1.networkutopia.com, 212.212.212.1, A).
- No DNS autoritativo, são criadas RRs do tipo A para www.networkutopia.com e do tipo MX para o domínio.

Registro de um Domínio

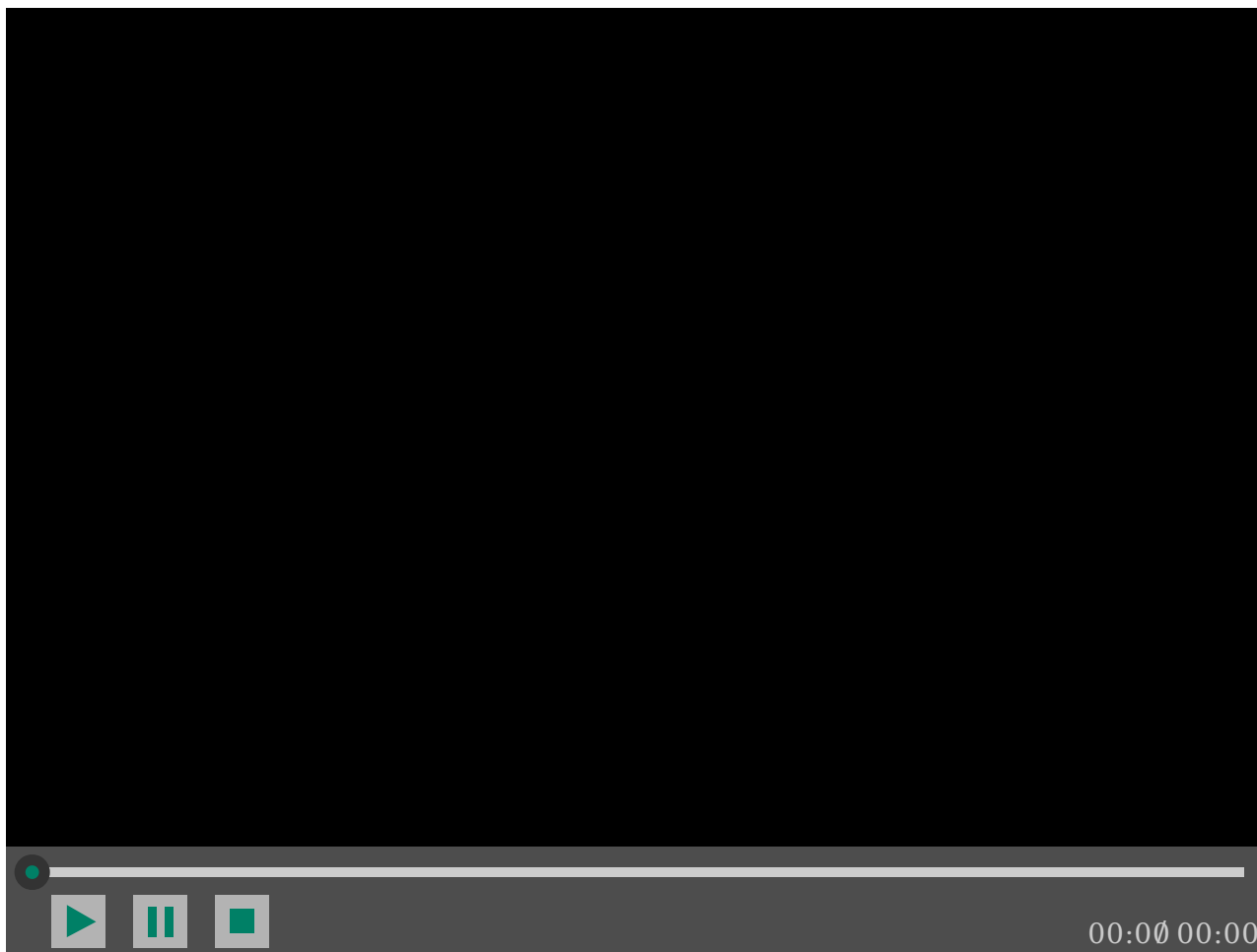


- Usamos a ferramenta `dig` para fazer consultas ao DNS.
- Consulta inicial do tipo NS ao domínio `uff.br`.
 - Resultado: três entradas listando servidores de DNS autoritativos.
- Segunda consulta: entrada do tipo A para `server.uff.br`.
 - Resultado: endereço IP do servidor.
- Note que respostas **não são autoritativas**.
 - Possivelmente um cache do servidor DNS local utilizado.

Respostas Autoritativas vs. Não-autoritativas



- Mesma consulta repetida duas vezes:
 - Do tipo A para nome `www.midiacom.uff.br`.
- Inicialmente, usamos um servidor local qualquer.
 - Resultado: resposta **não-autoritativa**.
 - Possivelmente cache (pode estar desatualizada!).
- Segunda tentativa: usamos um dos servidores de DNS de `uff.br` como “servidor local”
 - Resultado: **resposta autoritativa**.



- Como servidor de e-mail do remetente sabe qual o servidor de e-mail do destinatário?
 - Endereço de e-mail associado a um domínio.
 - e.g. `user@exemplo.com`.
 - Servidor do remetente faz consulta do tipo MX a domínio do destinatário.
- Consulta do tipo MX retorna um **nome**
 - Ainda precisa de uma nova resolução.
 - Consulta do tipo A.

Atacando o DNS

- **Ataques de DDoS:**

- Bombardear servidores raiz com tráfego.
 - Até hoje, não foi bem sucedido.
 - Técnicas de filtro de tráfego.
 - Servidores de DNS locais fazem cache dos IPs dos servidores TLD, evitam acessos ao servidores raiz.
- Bombardear servidores TLD.
 - Potencialmente mais perigoso.

- **Ataques de redirecionamento:**

- Homem-no-meio.
 - Intercepta requisições.
- Envenenamento do DNS.
 - Envia respostas adulteradas para servidor de DNS, que faz cache das informações.

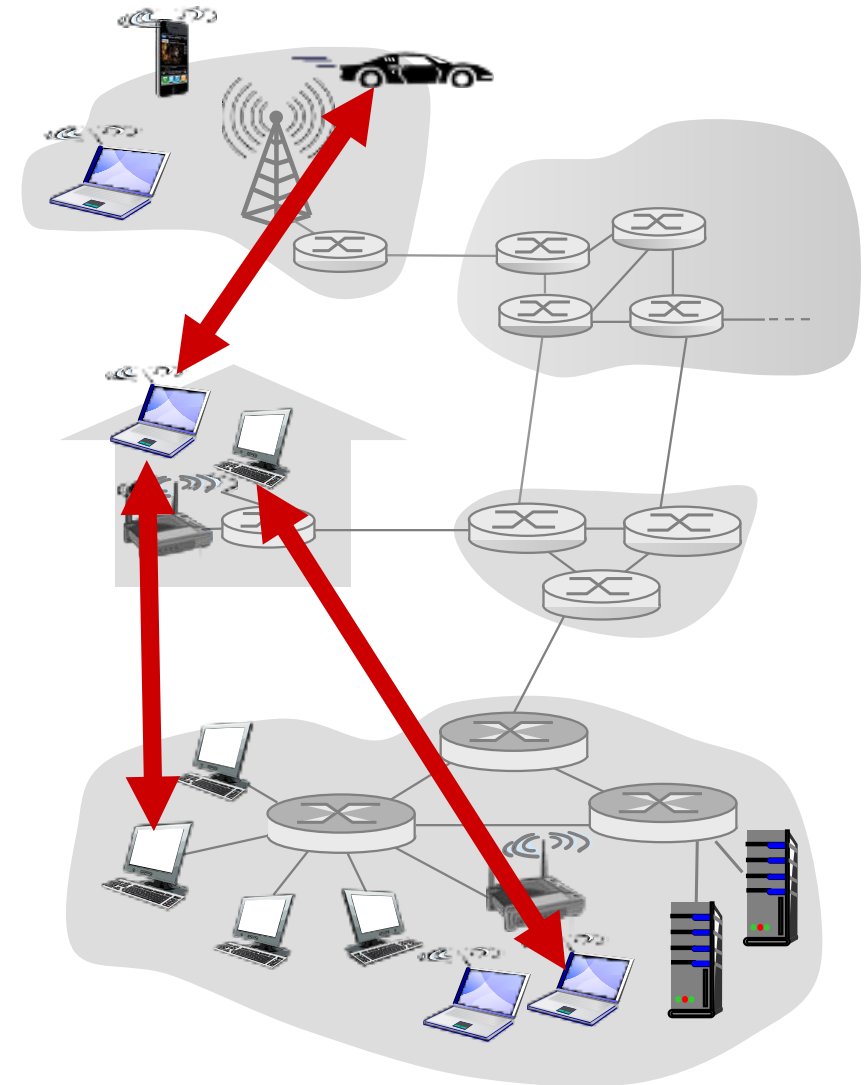
- **Exploração do DNS para DDoS:**

- Envia requisições com IP de origem forjado (IP da vítima).
- Requer amplificação.

Aplicações P2P

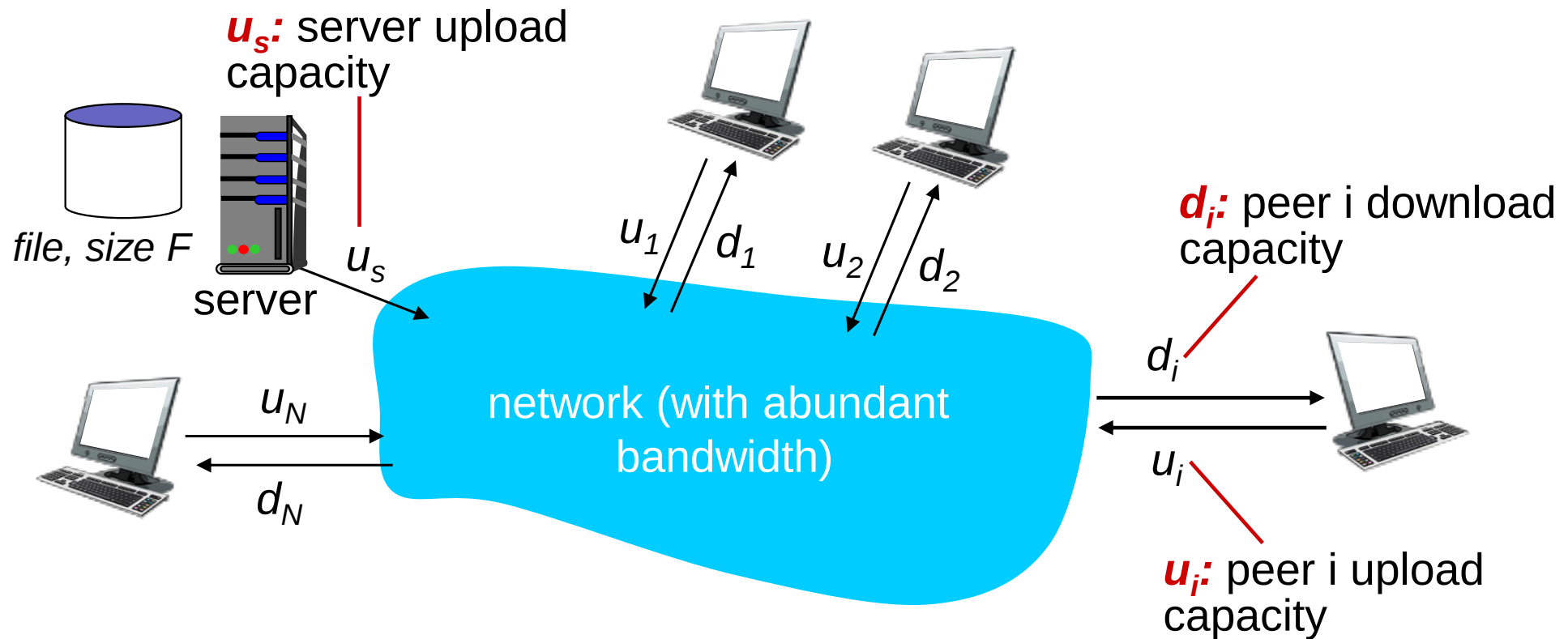
Arquitetura P2P Pura

- Não há servidor sempre ligado.
- Sistemas finais arbitrários se comunicam diretamente.
- Pares se conectam à rede P2P de forma intermitente, podem trocar de endereço IP.
- **Exemplos:**
 - Distribuição de arquivos (e.g., BitTorrent).
 - Streaming (e.g., KanKan).
 - VoIP (e.g., Skype).



Distribuição de Arquivos: Cliente—Servidor vs. P2P

- **Pergunta:** quanto tempo é necessário para distribuir um arquivo (tamanho F) de um servidor para N clientes?
 - Capacidades de *download/upload* dos clientes é um recurso limitado.



Distribuição de Arquivos: Cliente—Servidor

- **Transmissão pelo servidor:** precisa enviar (*upload*) **sequencialmente** N cópias do arquivo.

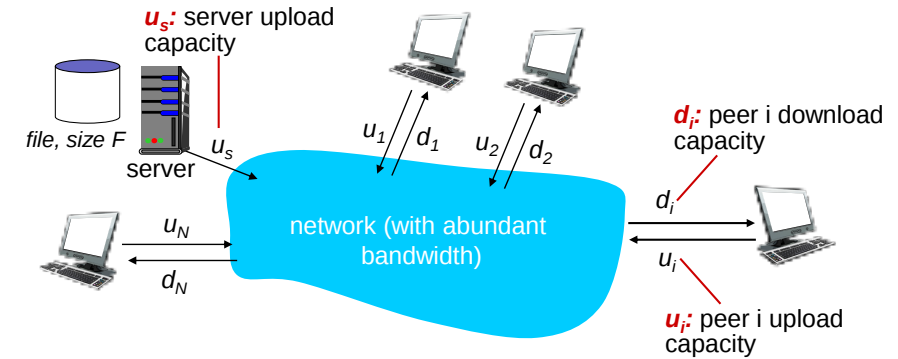
- Tempo para enviar uma cópia: $\frac{F}{u_s}$.
- Tempo para enviar N cópias: $\frac{N \cdot F}{u_s}$.

- **Cliente:** cada cliente precisa receber (*download*) sua cópia do arquivo.
- d_{min} = capacidade de *download* mínima entre todos os clientes.
- Tempo máximo de *download* entre os clientes:

$$\frac{F}{d_{min}}.$$

Tempo para distribuir arquivo de tamanho F para N clientes utilizando abordagem Cliente—Servidor

$$D_{C-S} = \max \left\{ \frac{N \cdot F}{u_s}, \frac{F}{d_{min}} \right\}$$



Distribuição de Arquivos: P2P

- **Transmissão pelo servidor:** precisa enviar (*upload*) **ao menos uma cópia**.

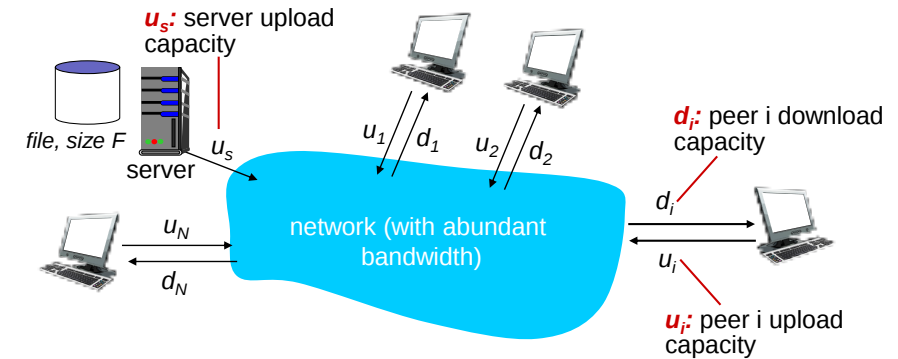
- Tempo para enviar uma cópia: $\frac{F}{u_s}$.

- **Cliente:** cada cliente precisa receber (*download*) sua cópia do arquivo.

- Tempo máximo de *download* entre os clientes:

$$\frac{F}{d_{min}}.$$

- **Clientes:** em conjunto, clientes farão *download* de $N \cdot F$ bits.
- Taxa máxima de *upload* (que limita taxa de *download*) é $u_s + \sum u_i$.



Tempo para distribuir arquivo de tamanho F para N clientes utilizando abordagem P2P

$$D_{P2P} = \max \left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{N \cdot F}{u_s + \sum u_i} \right\}$$

Distribuição de Arquivos: Comparação (I)

- Caso Cliente—Servidor:

$$D_{C-S} = \max \left\{ \frac{N \cdot F}{u_s}, \frac{F}{d_{\min}} \right\}$$

- Demanda cresce linearmente com N .
- Capacidade do servidor é fixa.
- Caso P2P:

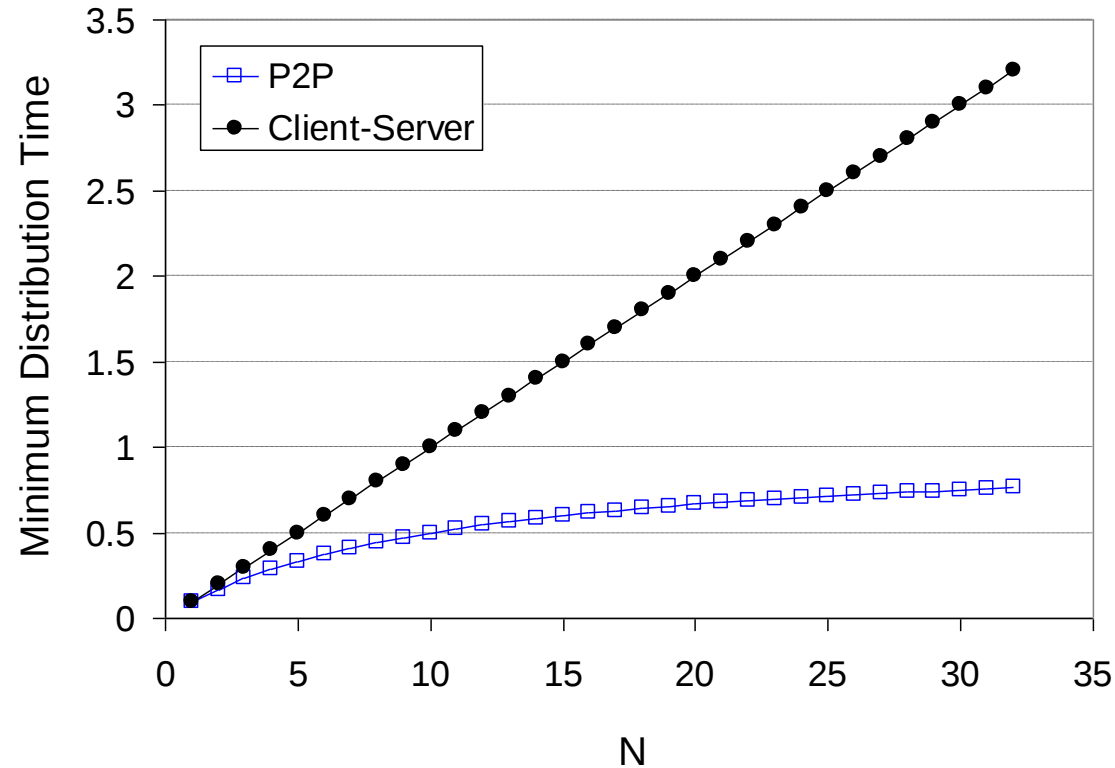
$$D_{P2P} = \max \left\{ \frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{N \cdot F}{u_s + \sum u_i} \right\}$$

- Demanda aumenta linearmente com N .
- Mas também a capacidade de *upload*.
- Resultado: **tempo de distribuição cresce, mas de forma mais escalável.**

Distribuição de Arquivos: Comparação (II)

- Exemplo numérico:

- $\frac{F}{u} = 1$ hora.
- Capacidade de *upload* do servidor 10x maior que dos clientes.
- $d_{min} \geq us$.

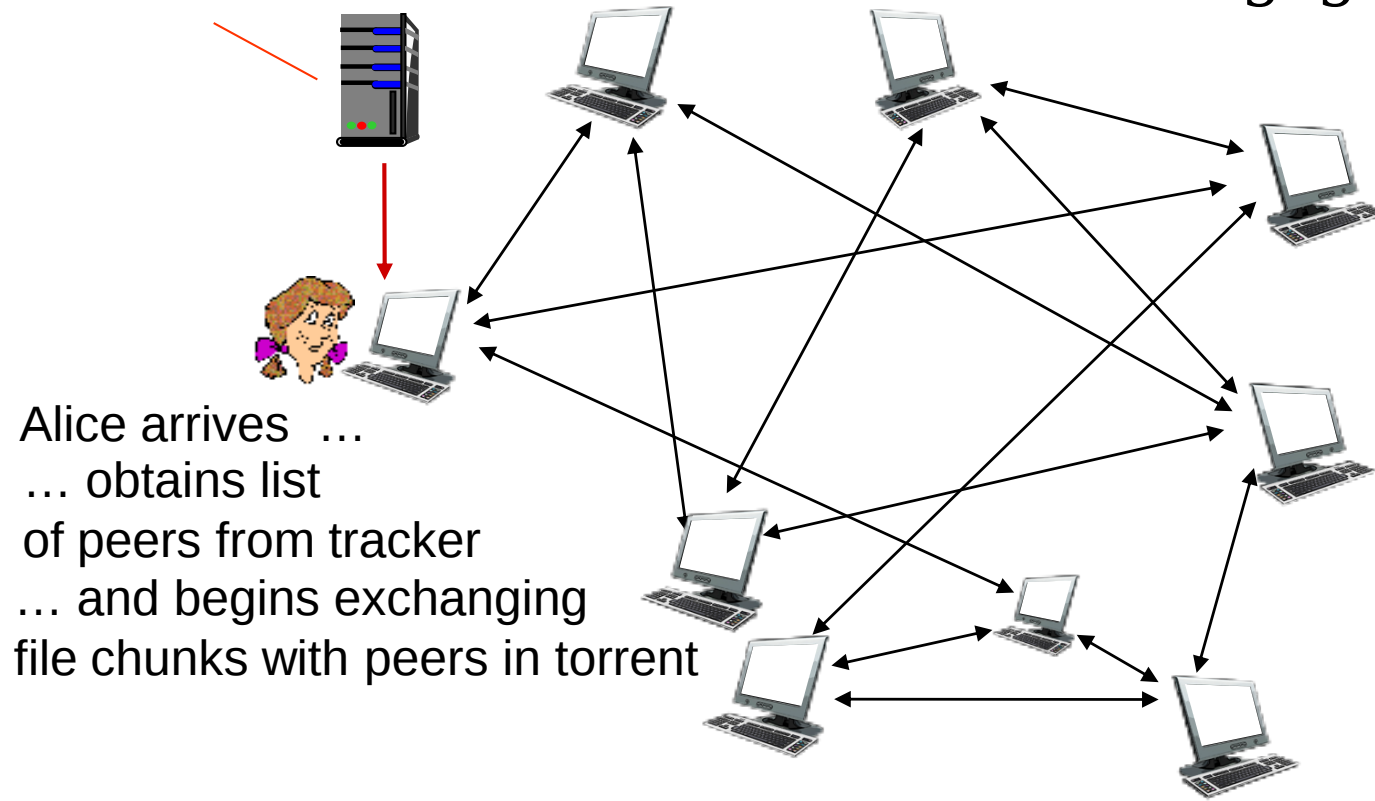


Distribuição de Arquivos P2P: Exemplo do BitTorrent (I)

- Arquivo dividido em pedaços de (normalmente) 256KB.
- Pares no torrent enviam/recebem pedaços do arquivo.

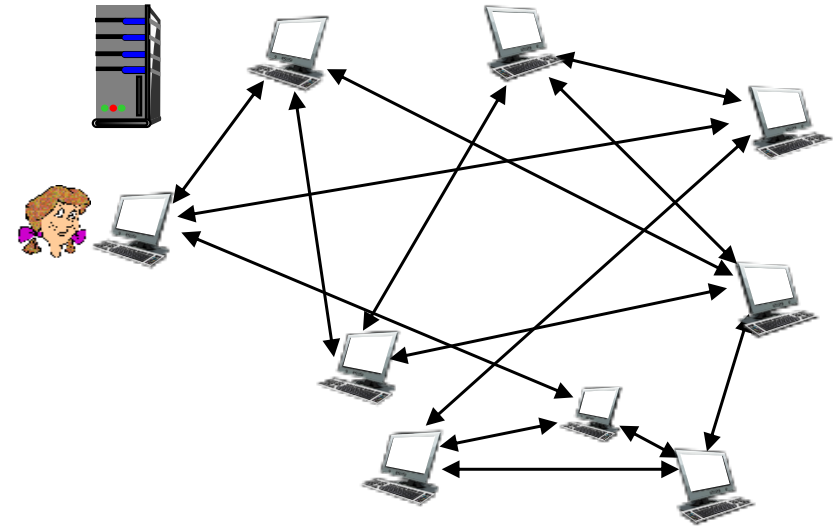
tracker: tracks peers
participating in torrent

torrent: group of peers
exchanging chunks of a file



Distribuição de Arquivos P2P: Exemplo do BitTorrent (II)

- Par se junta ao torrent:
 - Não possui nenhum pedaço, mas os acumulará com o tempo de outros pares.
 - Se registra com o *tracker* para obter uma lista de pares, se conecta a um subconjunto dos pares (“vizinhos”).
- Enquanto baixa, pares fazem *upload* para outros pares.
- Par pode alterar os pares com que troca pedaços.
- **Churn:** pares vem e vão.
- Quando um par tem o arquivo inteiro (todos os pedaços), ele pode (de forma egoísta) sair ou (de forma altruísta) ficar no torrent.



BitTorrent: Requisitando, Enviando Pedacos do Arquivo

● **Requisitando pedacos:**

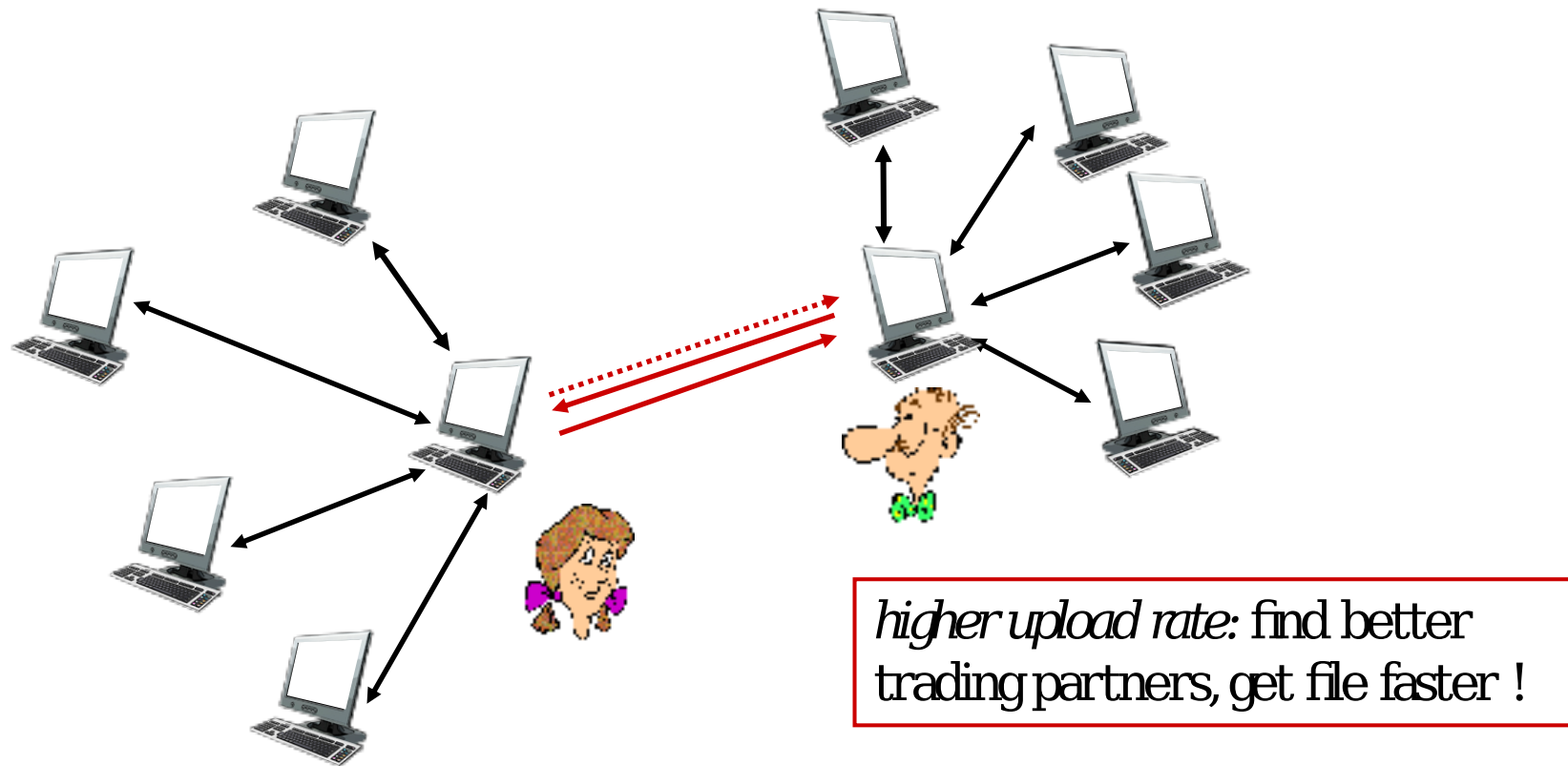
- Em um dado momento, diferentes pares possuem diferentes subconjuntos dos pedacos do arquivo.
- Periodicamente, um par pede aos outros pares uma lista dos pedacos que possuem.
- Par então requisita pedacos que não possui aos pares, começando pelos mais raros.

● **Enviando pedacos: tit-for-tat.**

- Par envia pedacos aos 4 pares que atualmente o enviam pedacos na taxa mais alta.
 - Outros pares sofrem *choking* (i.e., não recebem pedacos).
 - Uma nova avaliação dos 4 melhores pares é feita a cada 10 segundos.
- A cada 30 segundos: par seleciona aleatoriamente outro par, começa a enviar pedacos.
 - *Optimistically unchoke*.
 - Dá oportunidade de pares demonstrarem que são bons *uploaders*.
 - Par escolhido de forma aleatória pode se tornar um dos 4 melhores.

BitTorrent: Tit-For-Tat

- (1) Alice “optimistically unchokes” Bob
- (2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice’s top-four providers



Resumo da Aula...

- **DNS: objetivo.**

- Sistema que mapeia IPs a nomes.
- Simplifica identificação de *hosts*.

- **DNS: características.**

- Base de dados distribuída.
- Nomeação hierárquica.
 - Domínios, subdomínios, ...
- **Evita ponto único de falha.**
- Evita concentração do tráfego.
- Evita distância excessiva de certos clientes.

- **DNS: tipos de servidores.**

- Raiz, TLD, autoritativo, local.

- **DNS: métodos de resolução.**

- **Iterativo:** servidor responde com próximo servidor a ser consultado.
- **Recursivo:** servidor assume responsabilidade de achar o mapeamento.

- **DNS: resgistros.**

- Tipo=A: definição de **nome canônico**.
- Tipo=NS: definição de servidor autoritativo para o domínio.
- Tipo=CNAME: definição de apelidos para *hosts*.
- Tipo=MX: definição de servidor de e-mail para o domínio.

- **P2P: escalabilidade.**

- Mais demanda, mais oferta.
- Desde que pares contribuam.
 - i.e., evitar **free-riders**.
 - Bit-torrent: *tit-for-tat*.

Próxima Aula...

- Encerraremos a discussão sobre camada de aplicação.
- Falaremos sobre a API de sockets.
 - Funções disponíveis.
 - Organização tradicional de um programa com sockets.
 - Sockets TCP vs. sockets TCP.
 - Alguns exemplos.