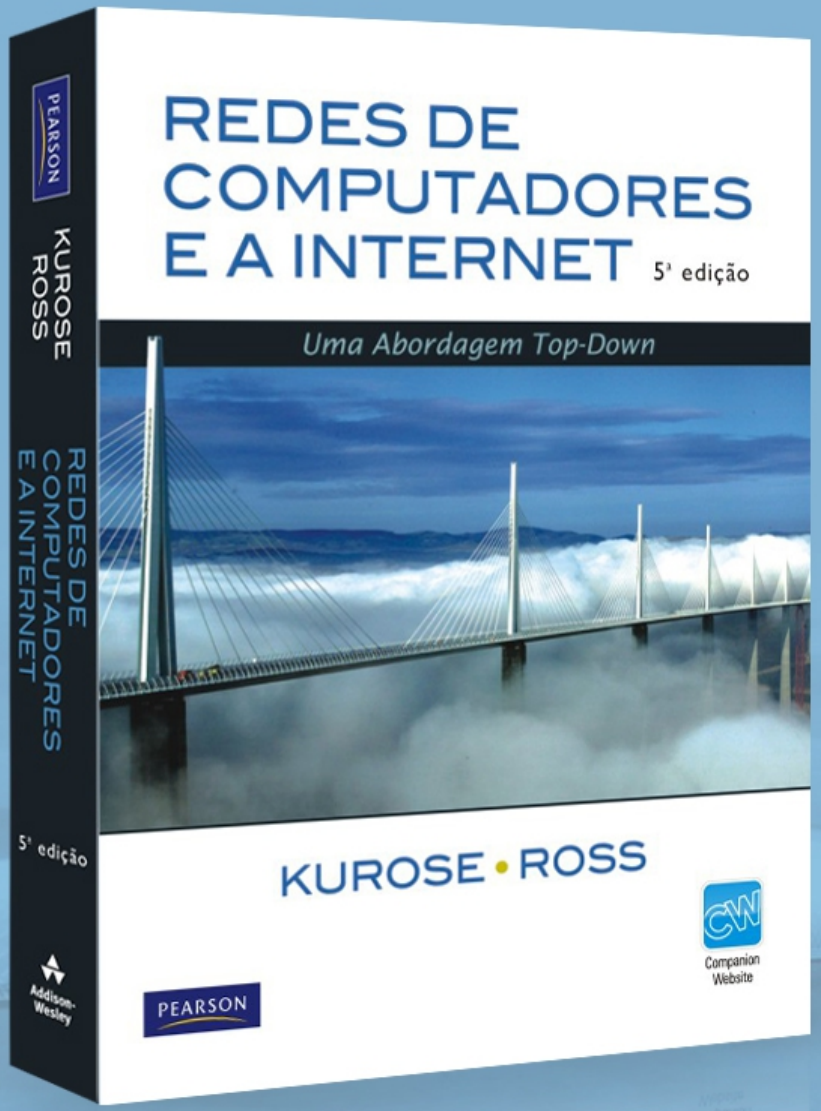
Capítulo 2
Camada de aplicação



Nota sobre o uso destes slides ppt:

Estamos disponibilizando estes slides gratuitamente a todos (professores, alunos, leitores). Eles estão em formato do PowerPoint para que você possa incluir, modificar e excluir slides (incluindo este) e o conteúdo do slide, de acordo com suas necessidades. Eles obviamente representam *muito* trabalho da nossa parte. Em retorno pelo uso, pedimos apenas o seguinte:

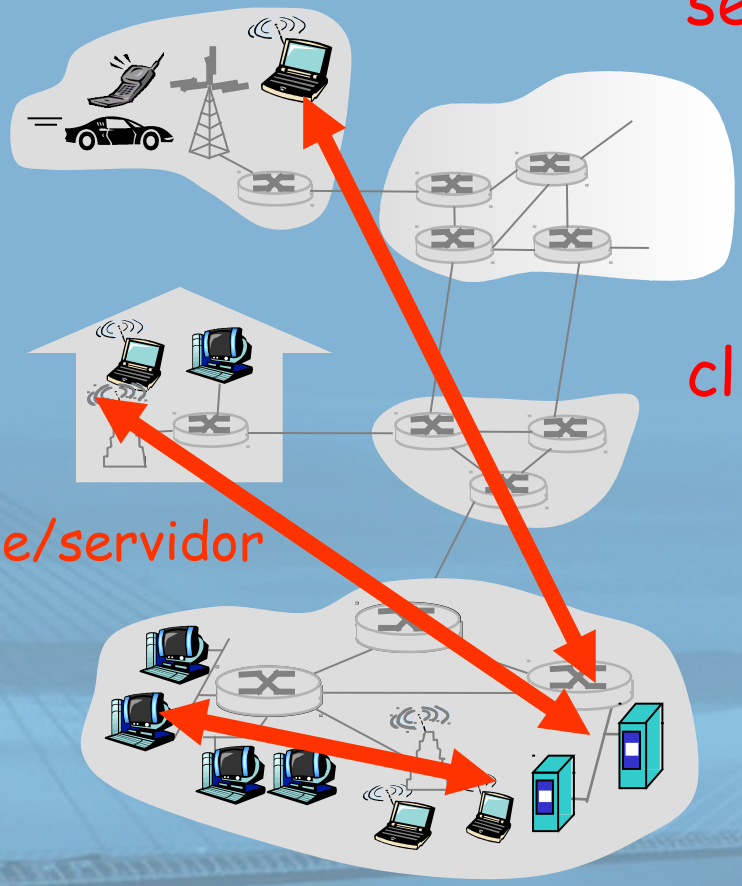
- Se você usar estes slides (por exemplo, em sala de aula) sem muita alteração, que mencione sua fonte (afinal, gostamos que as pessoas usem nosso livro!).
- Se você postar quaisquer slides sem muita alteração em um site Web, que informe que eles foram adaptados dos (ou talvez idênticos aos) nossos slides, e inclua nossa nota de direito autoral desse material.

Obrigado e divirta-se! JFK/KWR

Arquiteturas de aplicação

- Cliente-servidor
  - Incluindo centros de dados/cloud computing
- Peer-to-peer (P2P)
- Híbrida de cliente-servidor e P2P

Arquitetura cliente-servidor



cliente/servidor

servidor:

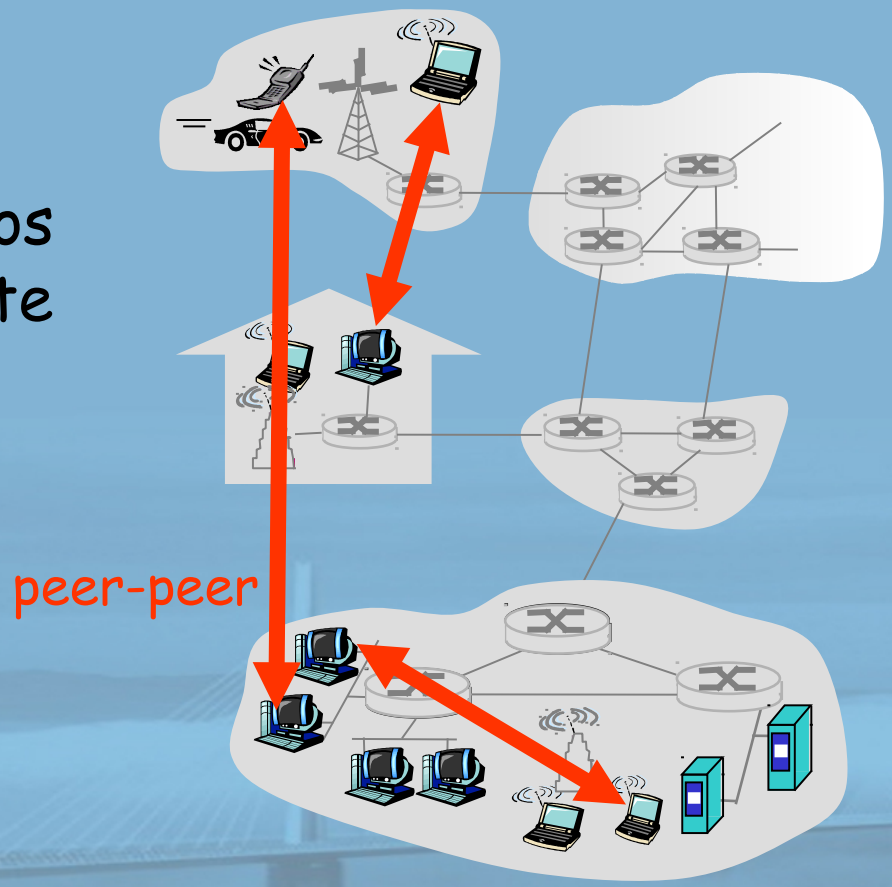
- hospedeiro sempre ligado
- endereço IP permanente
- server farms por expansão

clientes:

- comunicam-se com o servidor
- podem estar conectados intermitentemente
- podem ter endereços IP dinâmicos
- não se comunicam diretamente entre si

Arquitetura P2P pura

- *nenhum* servidor sempre ligado
- sistemas finais arbitrários se comunicam diretamente
- pares são conectados intermitentemente e mudam endereços IP



altamente escalável, mas difícil de administrar

Híbrido de cliente-servidor e P2P

Skype

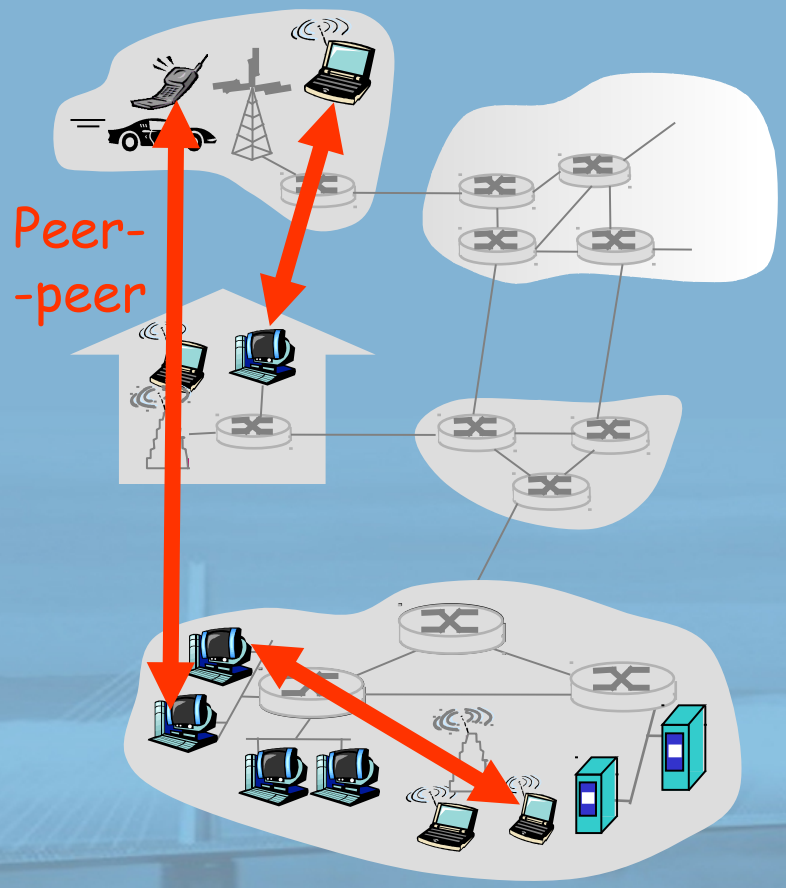
- aplicação P2P voice-over-IP P2P
- servidor centralizado: achando endereço da parte remota:
- conexão cliente-cliente: direta (não através de servidor)

Mensagem instantânea

- bate-papo entre dois usuários é P2P
- serviço centralizado: detecção/localização da presença do cliente
  - usuário registra seu endereço IP com servidor central quando entra on-line
  - usuário contacta servidor central para descobrir endereços IP dos parceiros

Arquitetura P2P pura

- *sem* servidor sempre ligado
- sistemas finais arbitrários se comunicam diretamente
- pares estão conectados intermitentemente e mudam de endereços IP



- Três tópicos:
  - distribuição de arquivos
  - procura de informações
  - estudo de caso: Skype

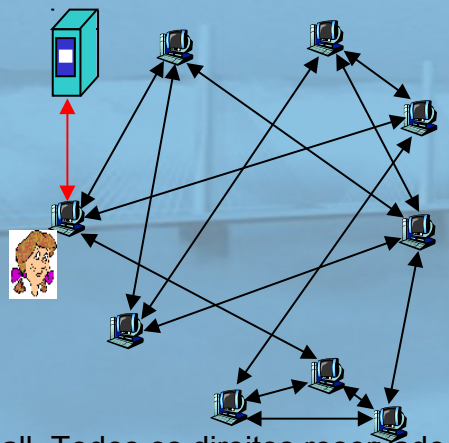
BitTorrent

Criado pelo desenvolvedor norte-americano Bram Cohen em 2001;

Quando um arquivo está sendo baixado para um computador, "pedacinhos" deste são obtidos de várias outras máquinas simultaneamente, não apenas de uma;

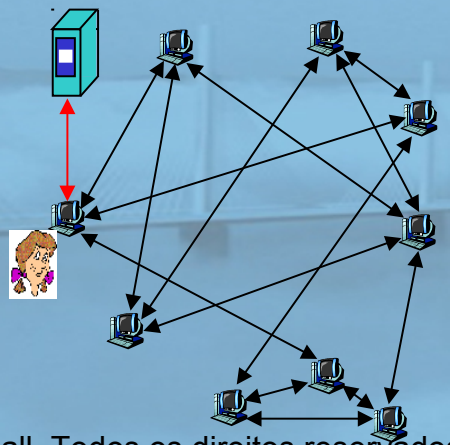
Exemplos de BitTorrent websites:

The Pirate Bay, Rarbg (Tracker), Rutracker (Tracker), Torrents.me,



BitTorrent

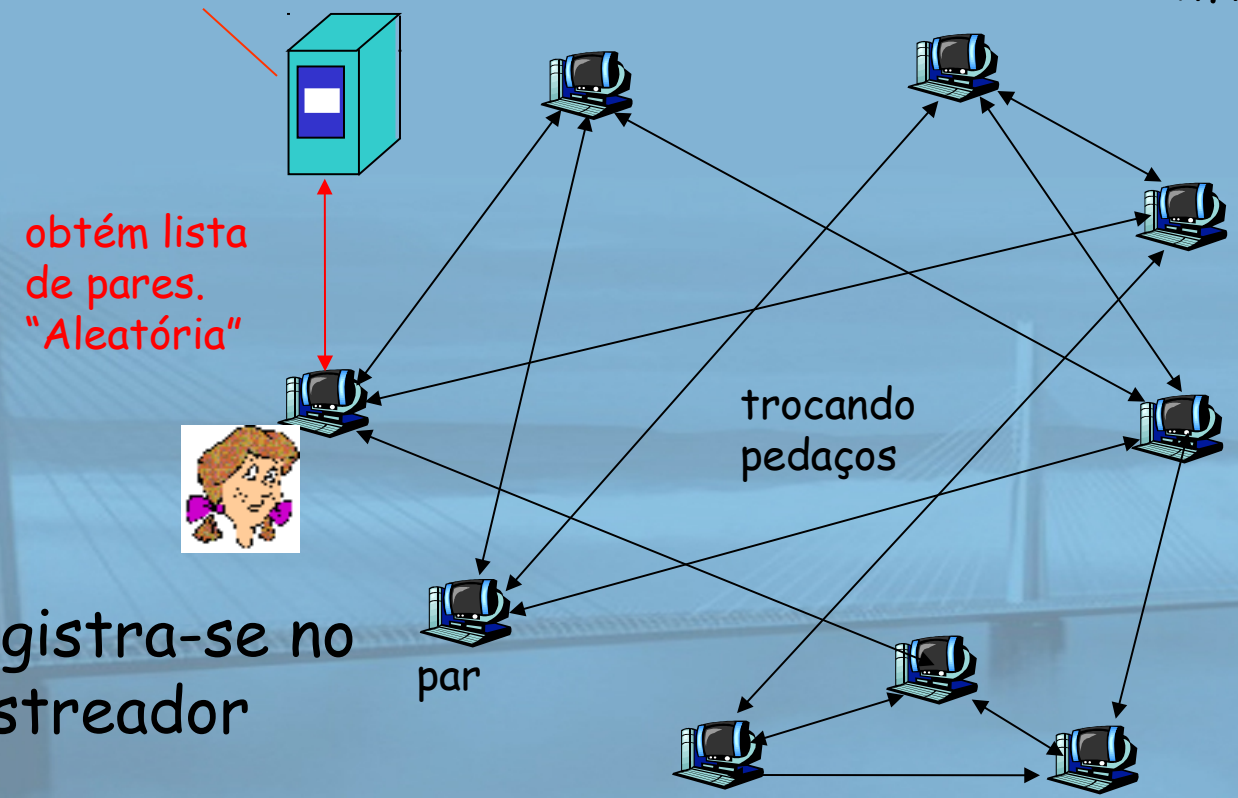
- Torrent: coleção de todos os pares que participam da distribuição de um determinado arquivo.
- Os pares em um torrent fazem download de blocos de tamanho igual do arquivo entre si.
- Tamanho típico de 256 Kbytes.
- Quando um par entra em um Torrent ele não tem nenhum bloco.



Distribuição de arquivos: BitTorrent

rastreador: verifica pares que participam do torrent

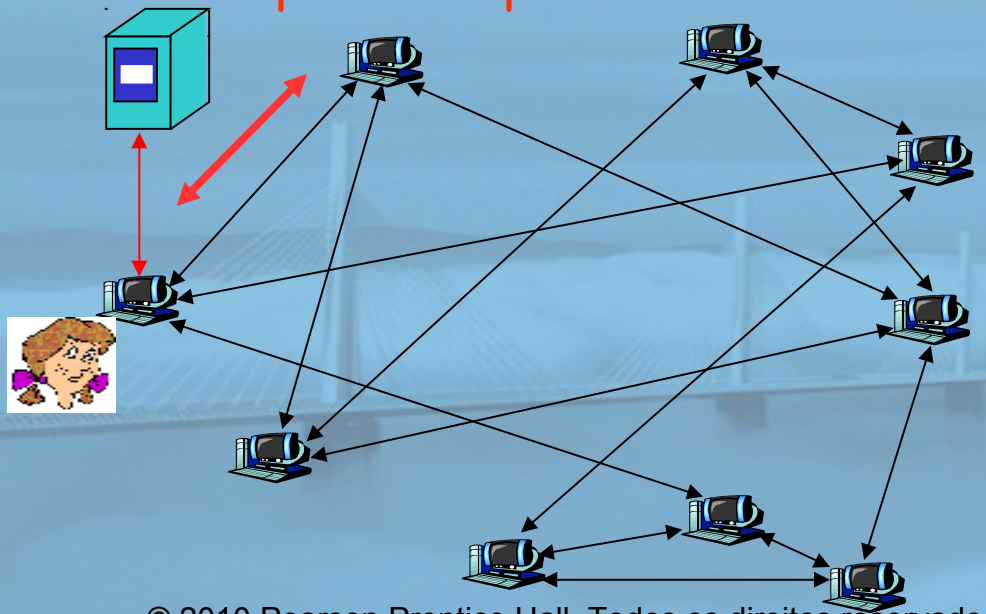
- Cada torrent tem um nó de infraestrutura



BitTorrent

-Alice tenta estabelecer conexões TCP com os pares da lista;
-Pede lista de pares vizinhos de tempos em tempos.

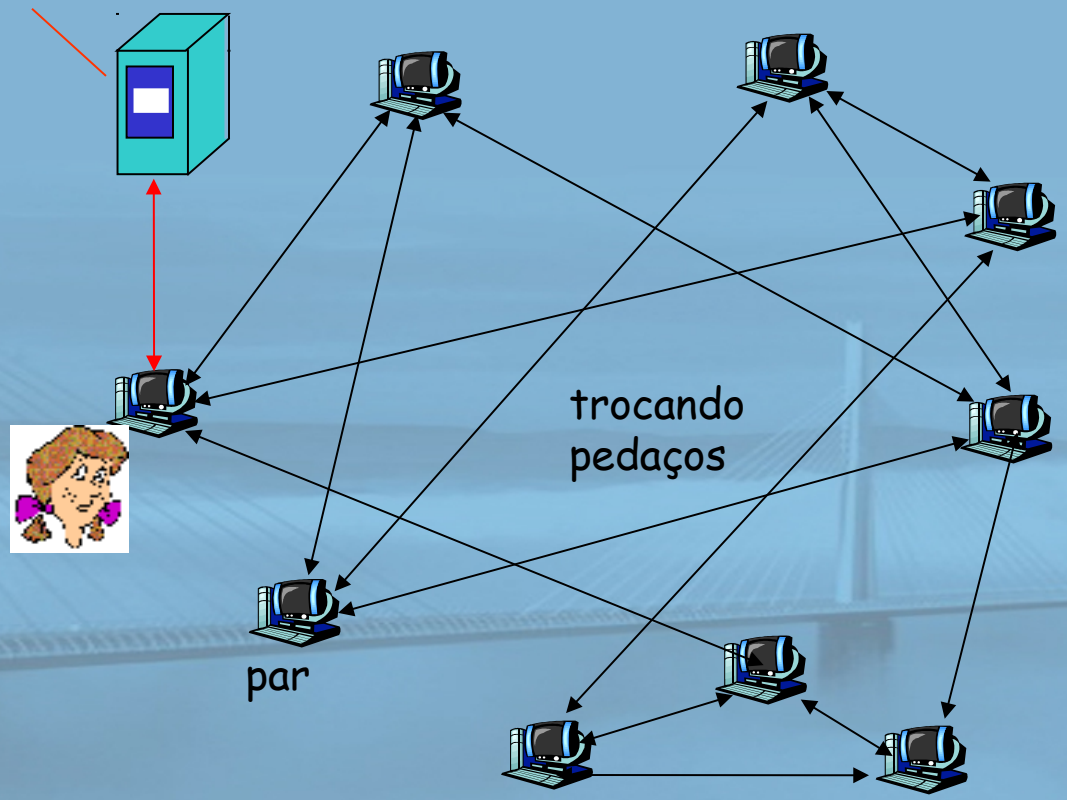
- Acúmulo de Blocos:
  - Par acumula blocos com o tempo;
  - Par faz download de blocos de outros pares e upload de blocos para outros pares.



Conexão com Pares: BitTorrent_

1-Quais blocos eu solicito primeiro?

2-Para quais vizinhos eu envio os blocos que tenho?



Empurrando pedaços

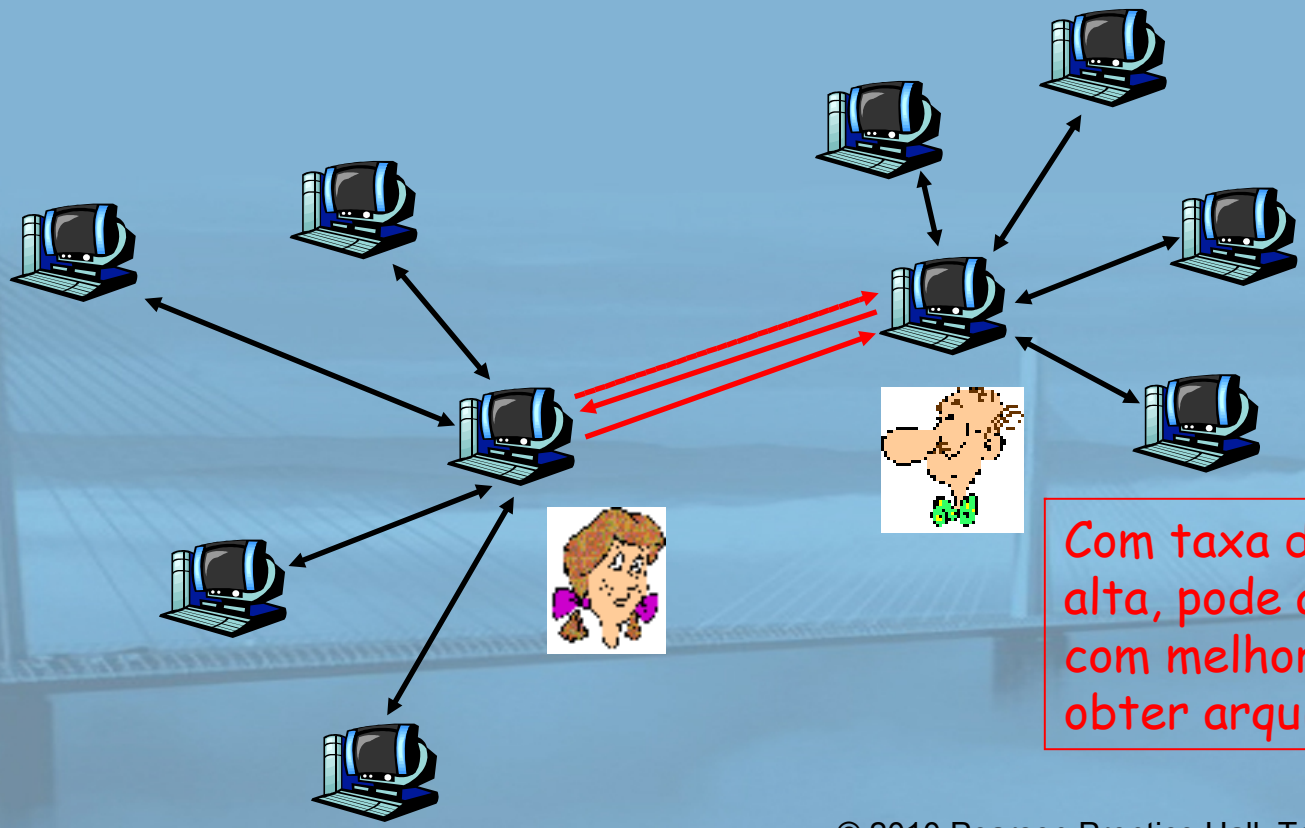
- periodicamente, um par (Alice) pede a cada vizinho a lista de pedaços que eles têm
- Alice envia requisições para seus pedaços que faltam
  - mais raros primeiro
  - Alice verifica quais pedaços são mais raros entre os vizinhos (menor no de cópias repetidas).

Enviando pedaços: olho por olho

- Alice envia pedaços a quatro vizinhos atualmente enviando seus pedaços na *velocidade mais alta*
  - reavalia 4 maiores a cada 10 s
- a cada 30 s: seleciona outro par aleatoriamente, começa a enviar pedaços
  - par recém-escolhido pode se juntar aos 4 maiores
  - "desafoga" de forma otimista

BitTorrent: Olho por olho

(1) Alice "desafoga" Bob de forma otimista
(2) Alice um dos quatro maiores provedores de Bob; Bob recíproco
(3) Bob torna-se um dos quatro maiores provedores de Alice



Com taxa de upload mais alta, pode achar parceiros com melhor negociação & obter arquivo mais rápido!

Distributed Hash Table (DHT)

- DHT = banco de dados P2P distribuído
- banco de dados tem duplas (chave, valor);
  - chave: número ss; valor: nome humano
  - chave: tipo conteúdo; valor: endereço IP
- pares consultam BD com chave
  - BD retorna valores que combinam com a chave
- pares também podem inserir duplas (chave, valor)
- Como distribuir duplas entre todos os pares (subconjunto de duplas)?

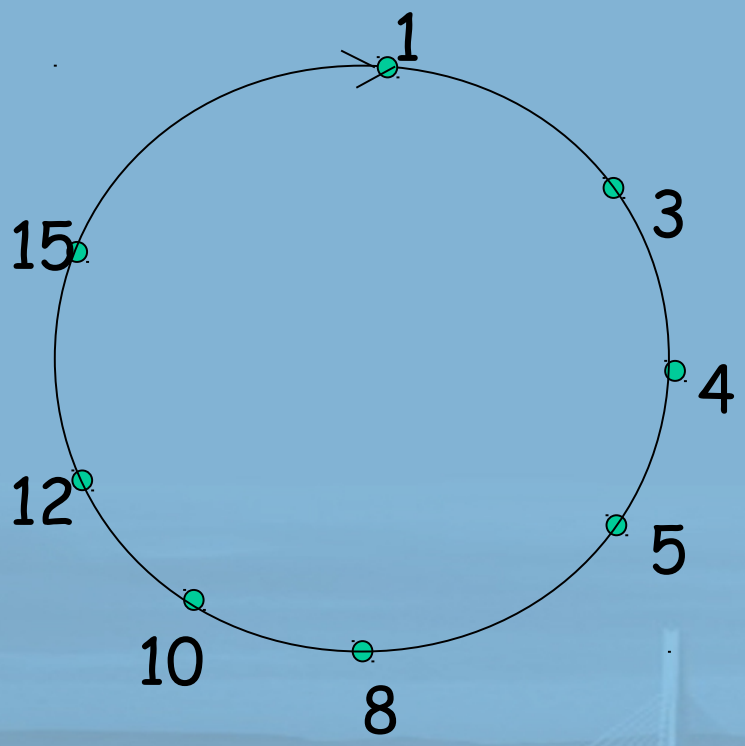
Identificadores DHT

- atribuem identificador inteiro a cada par no intervalo $[0,2^n - 1]$.
  - cada identificador pode ser representado por n bits (max=160 bits).
- A cada recurso é atribuído um identificador (mesmo espaço de identificadores dos pares (*n é fixo*).).
  - Identificador do recurso é o hash de alguma propriedade do recurso:
    - p. e., chave = h("Led Zeppelin IV"), nome do arquivo
  - É por isso que a chamamos de tabela "hash" distribuída

Como atribuir chaves aos pares?

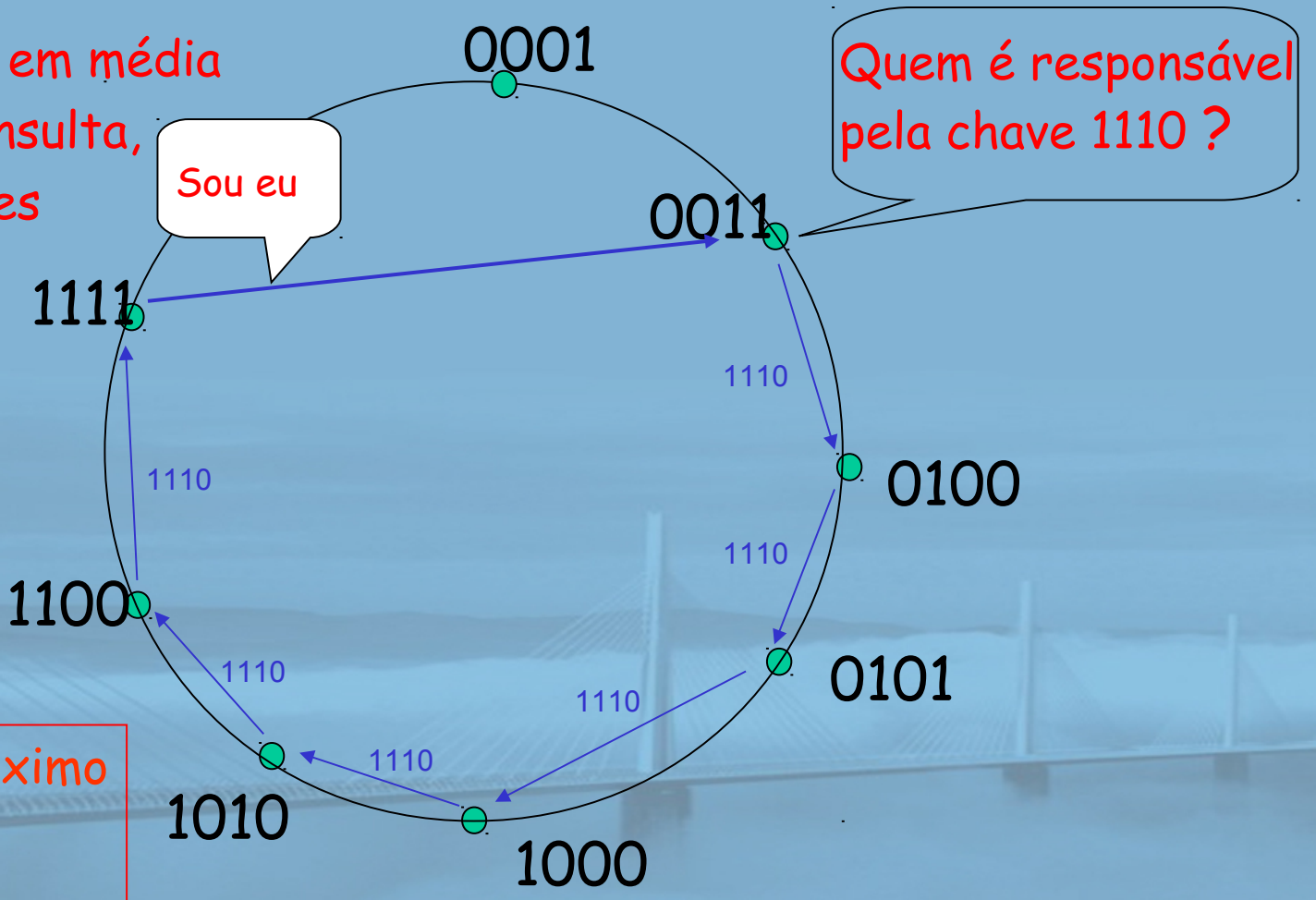
- questão central:
  - atribuir duplas (chave, valor) aos pares.
- regra: atribuir chave ao par que tem o ID mais próximo.
- convenção na aula: mais próximo é o sucessor imediato da chave.
- ex.: n = 4; pares: 1,3,4,5,8,10,12,14;
  - chave = 13, então par sucessor = 14
  - chave = 15, então par sucessor = 1

DHT circular



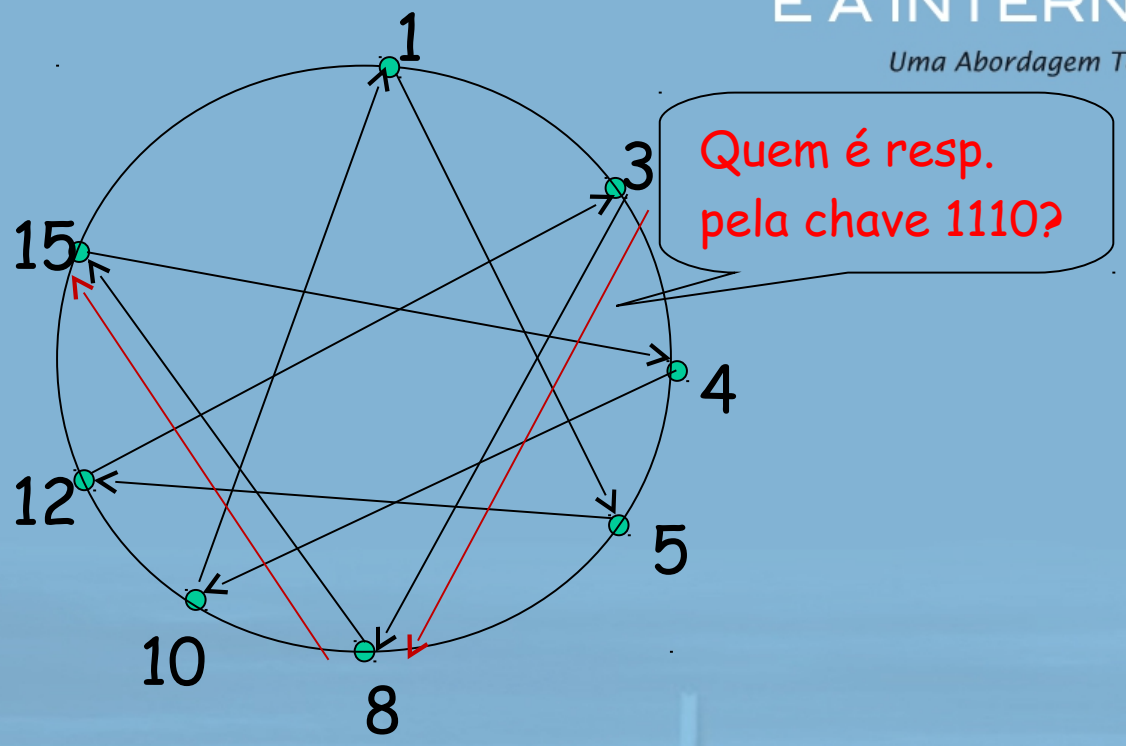
- cada par *só* conhece sucessor e predecessor imediato.
- "rede de sobreposição"

O(N) mensagens em média para resolver consulta, quando há N pares



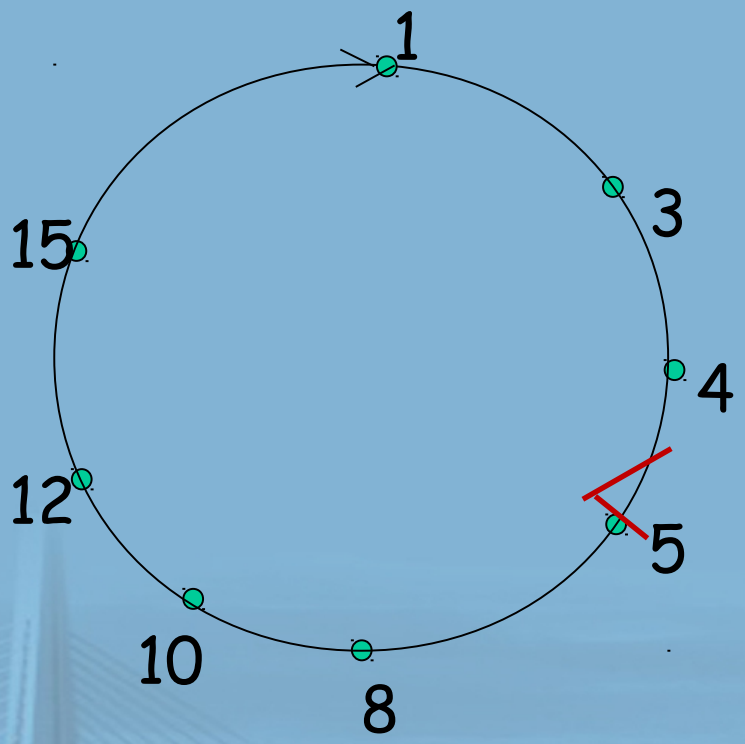
Define mais próximo como sucessor mais próximo

DHT circular com atalhos



- cada par registra endereços IP do predecessor, sucessor, atalhos
- reduzido de 6 para 2 mensagens
- possível criar atalhos de modo que O(log N) vizinhos, O(log N) mensagens na consulta

Peer Churn (Pares Dinâmicos)

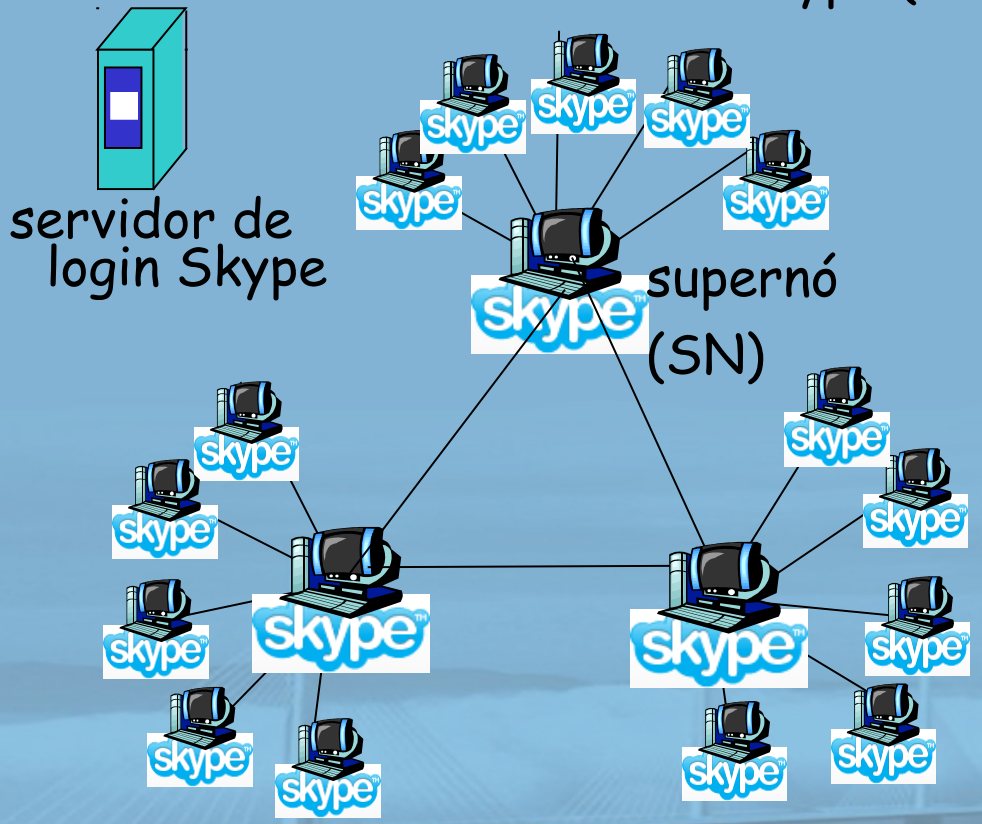


- para manejar o peer churn, é preciso que cada par conheça o endereço IP de seus dois sucessores.
- cada par periodicamente envia 'ping' aos seus dois sucessores para ver se eles ainda estão vivos.

- par 5 sai abruptamente
- par 4 detecta; torna 8 seu sucessor imediato; pergunta(endereço IP) a 8 quem é seu sucessor imediato; torna o sucessor imediato de 8 seu segundo sucessor.
- e se o par 13 quiser se juntar?

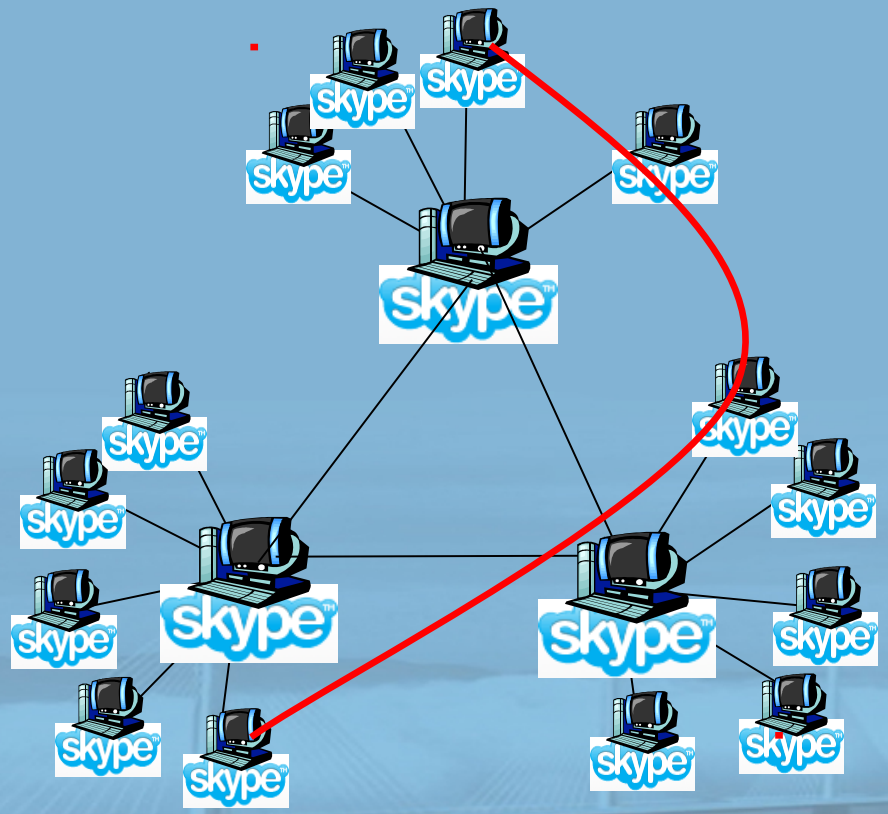
Estudo de caso do P2P: Skype

- inerentemente P2P: pares de usuários se comunicam.
- protocolo próprio da camada de aplicação (deduzido por engenharia reversa)
- sobreposição hierárquica com SNs
- índice compara usernames com endereços IP; distribuído por SNs



Pares como retransmissores

- problema quando Alice e Bob estão atrás de "NATs"
  - NAT impede que um par de fora inicie uma chamada para um par de dentro da rede
- solução:
  - usando os SNs de Alice e de Bob, o retransmissor é escolhido
  - cada par inicia a sessão com retransmissão.
  - pares agora podem se comunicar através de NATs com retransmissão



Capítulo 2: Camada de aplicação

REDES DE COMPUTADORES E A INTERNET 5ª edição
Uma Abordagem Top-Down

- 2.1 Princípios de aplicações de rede
- 2.2 A Web e o HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Aplicações P2P
- 2.7 Programação de sockets com UDP
- 2.8 Programação de sockets com TCP