Blockchain, Criptomoedas & Tecnologias Descentralizadas

Tecnologias descentralizadas:

Buscas e DHT

Prof. Dr. Marcos A. Simplicio Jr. – mjunior@larc.usp.br
Escola Politécnica, Universidade de São Paulo

Objetivos

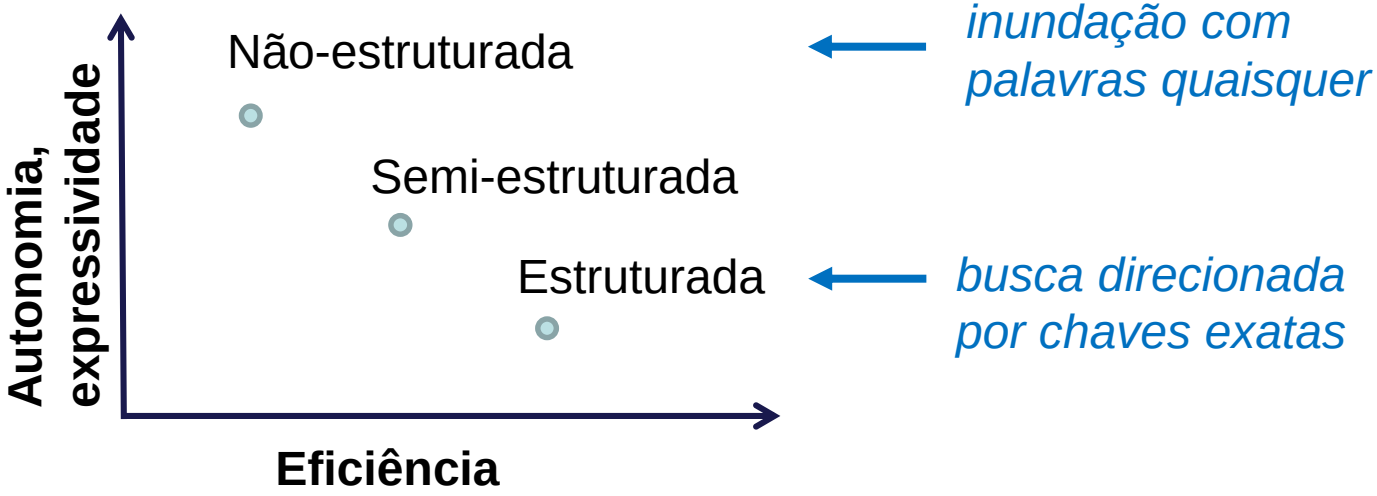
- Explicar os desafios envolvidos em sistemas de busca distribuída
- Apresentar algumas soluções para esses desafios
  - Busca às cegas
  - Busca usando informação local apenas
  - Busca usando informação distribuída na rede (DHT)

Busca em redes P2P: Desafios

- **Eficiência**: recursos (principalmente banda) utilizados
- **Autonomia**: nível de controle de cada nó sobre localização de (índice para) dado, conectividade e roteamento de mensagens
- **Expressividade**: por chave exata ou aproximada, por palavra, por faixa de valores, ...
- **Robustez**: estabilidade frente a alterações no estado da rede (ex.: entrada e saída de nós)
- **Escalabilidade**: acomodar milhões de nós sem degradação de eficiência
- **Completude**: garantia de sucesso e determinação de todas as referências

Busca em redes P2P: Desafios

- **Eficiência**: recursos (principalmente banda) utilizados
- **Autonomia**: nível de controle de cada nó sobre localização de (índice para) dado, conectividade e roteamento de mensagens
- **Expressividade**: por chave exata ou aproximada, por palavra, por faixa de valores, ...

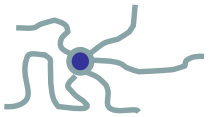


Taxonomia de buscas

- Busca cega
- Busca informada
  - Índice Local
    - Proativa
    - Reativa
  - **Índice Distribuído**
    - Sistemas baseados em DHT

Taxonomia: busca cega

- **Breadth-First Search (BFS) / Inundação (Gnutella)**
  - Encaminha busca para todos os vizinhos
  - Inundação limitada pelo TTL (time-to-live)
- **BFS modificado**
  - Inundação de parte dos vizinhos
  - Reduz tráfego de busca ao custo de menor taxa de acertos.
- **Random walk (Passeio Aleatório)**
  - Encaminha busca a conjunto aleatório de vizinhos escolhidos
  - Usa alguns passeios em paralelo
- **Inundação com profundidade iterativa**
  - BFS com TTL crescente, até atingir número desejado de nós encontrados

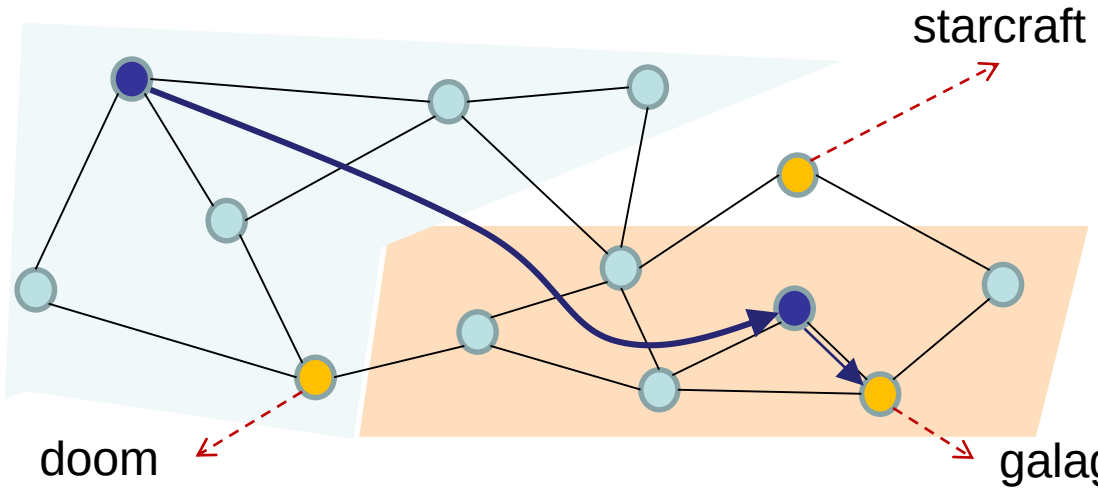


Taxonomia: índice local

Ex. de busca **proativa**: construção de índices vizinhos

- Busca em vizinhança
- Cada nó mantém um conjunto de hashes para palavras chave armazenadas em nós dentro de um pequeno raio (**r**)
- Busca é encaminhada para nó fora da vizinhança (raio > **r**) caso dado não seja encontrado na vizinhança

Nó A está procurando por "galaga"

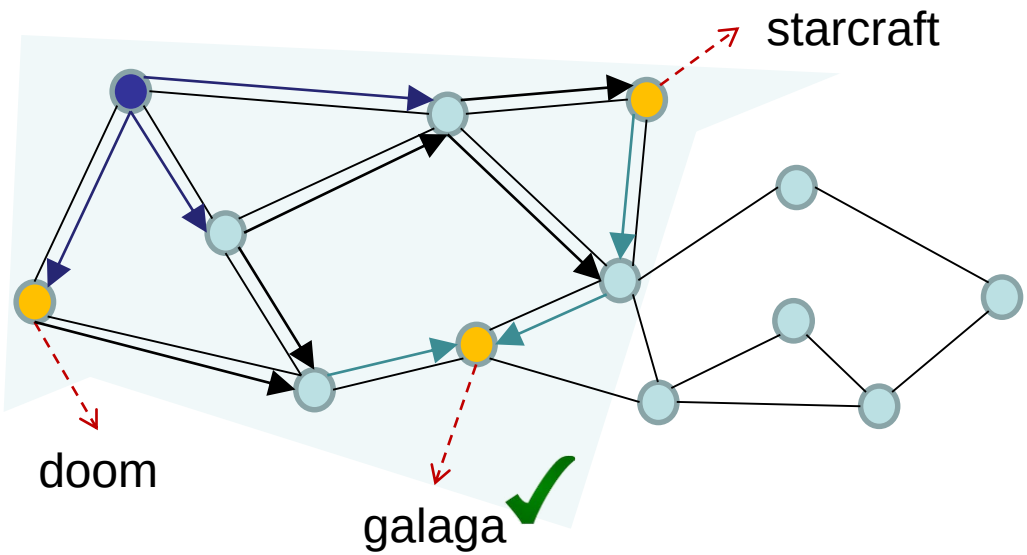


Taxonomia: índice local

Ex. de busca **proativa**: estratégia associativa

- Busca em vizinhança
- Nós se organizam em grupos de interesses
  - Ex.: dicionário de palavras chave, como filmes ou jogos
- Busca é inundada em um ou mais grupos de interesse

Nó A está procurando por "galaga": grupo de interesse por "jogos"



Taxonomia: índice local

- Exemplos de busca **reativa**
- Protocolo de Localização de Recursos Distribuídos
  - Passeio aleatório ao não encontrar informação localmente
  - Se dado for encontrado, sua localização é armazenada em cada nó no **caminho reverso** até o nó que fez a busca
- BFS Inteligente:
  - Versão informada do BFS modificado (com inundação de parte dos vizinhos)
  - Roteamento da busca é guiado por **decisões passadas**
  - Nós armazenam pares (busca, ID_nó): identifica nós interessados em certas chaves
  - Se dado for encontrado: informação de roteamento em todos os nós no **caminho reverso** é atualizada

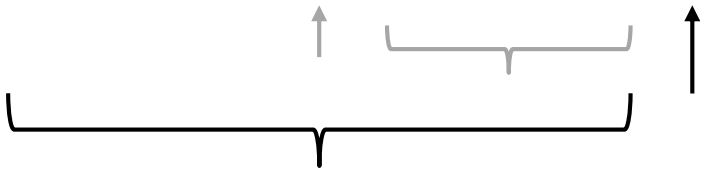
Taxonomia: DHT

- Recap: busca em memória local
  - Busca não ordenada: O(n)

33	45	43	17	91	15	71	86	6	62	21

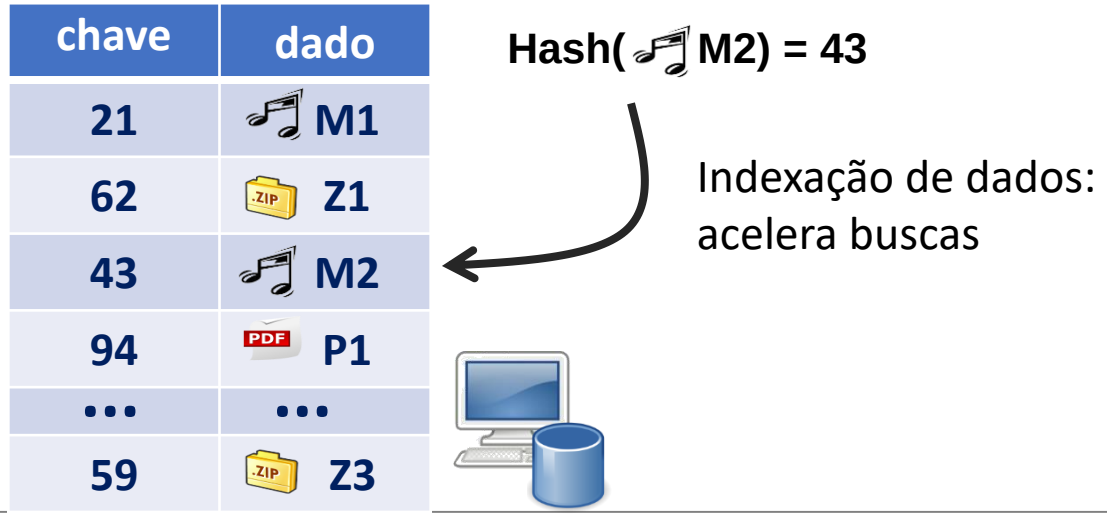
  - Busca binária: O (lg n)

6	15	17	21	33	43	45	62	71	86	91



Taxonomia: DHT

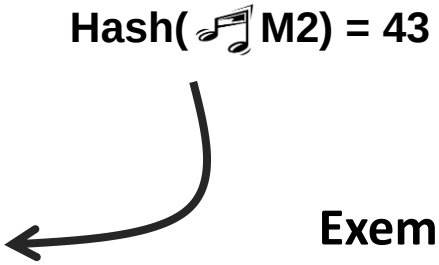
- Recap: tabela hash
  - Mapeia “chaves” em “valores”: buscas em O(1)
- Chamadas:
  - chave = hash(dado)
  - put(chave, valor): insere valor na tabela
  - get(chave): recupera valor da tabela



Taxonomia: DHT

- Tabela hash distribuídas (DHT): similar a tabela hash, mas espalhada na rede
  - chave = hash(dado)
  - buscar(chave) : IP_nó ➔ O(lg n)
  - rotear(IP_nó, PUT, chave, dado)
  - rotear(IP_nó, GET, chave) : dado

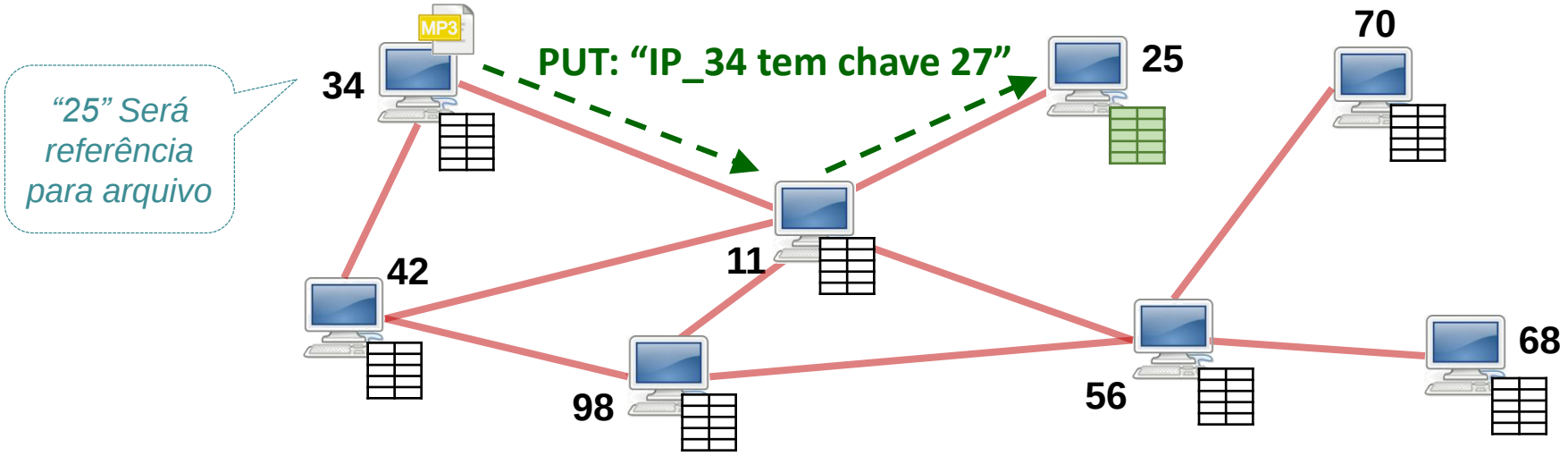
chave	dado
21	M1
62	Z1
43	M2
94	P1
...	...
59	Z3



Exemplos de algoritmos de "Rotear": Kademlia, CAN, Chord, Kelips, Pastry & Tapestry

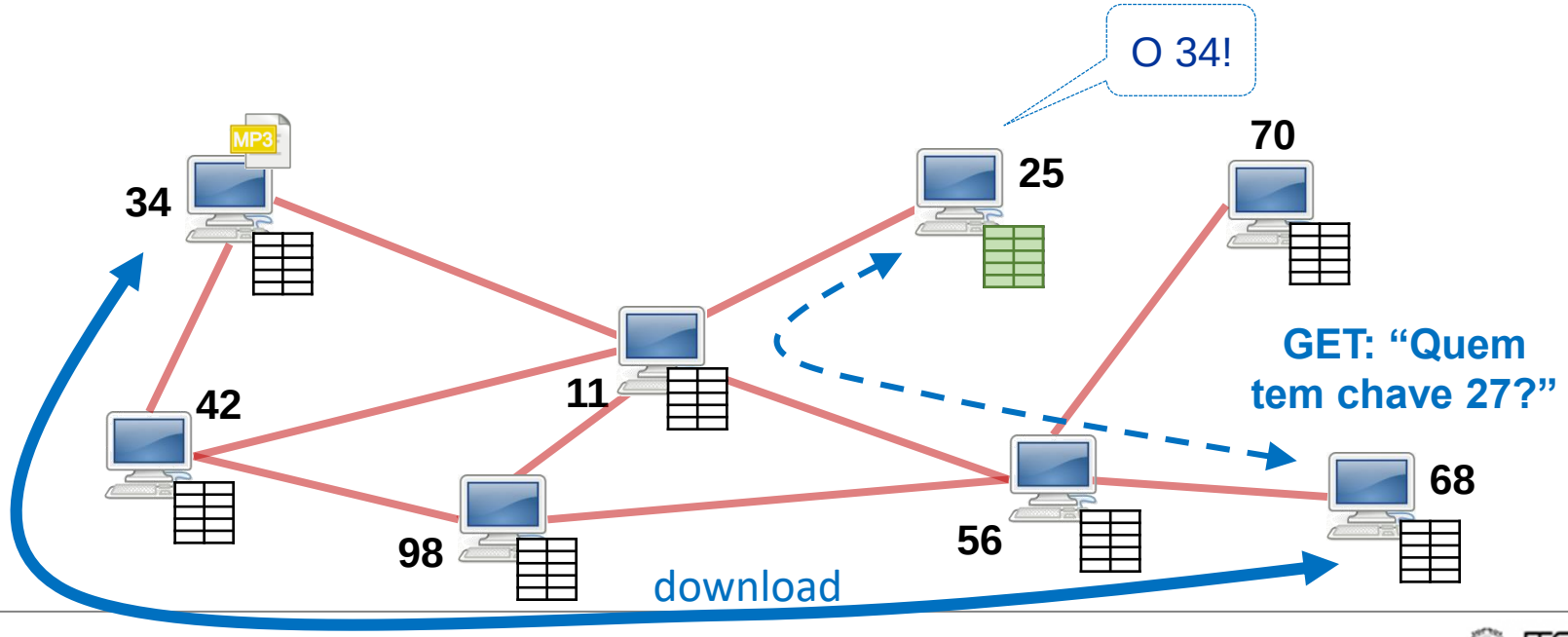
Taxonomia: DHT

- Tabela hash distribuídas (DHT): similar a tabela hash, mas espalhada na rede
  - chave = hash( ) = 27
  - buscar(chave) : IP_25
  - rotear(IP_25, PUT, chave, IP_34)



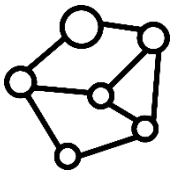
Taxonomia: DHT

- Tabela hash distribuídas (DHT): similar a tabela hash, mas espalhada na rede
  - chave = hash( ) = 27
  - buscar(chave) : IP_25
  - rotear(IP_25, GET, chave) : IP_34

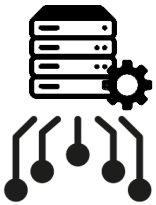


DHT: quem gerencia?

- Abordagem 1: DHT baseada em peers
  - Nós executando aplicação são também os nós da DHT
  - Exemplo comum em compartilhamento de arquivos



- Abordagem 2: DHT infraestruturada
  - Conjunto de nós gerenciados fornecem o serviço de DHT
  - Em alguns casos, pode dar suporte a mais de um tipo de aplicação

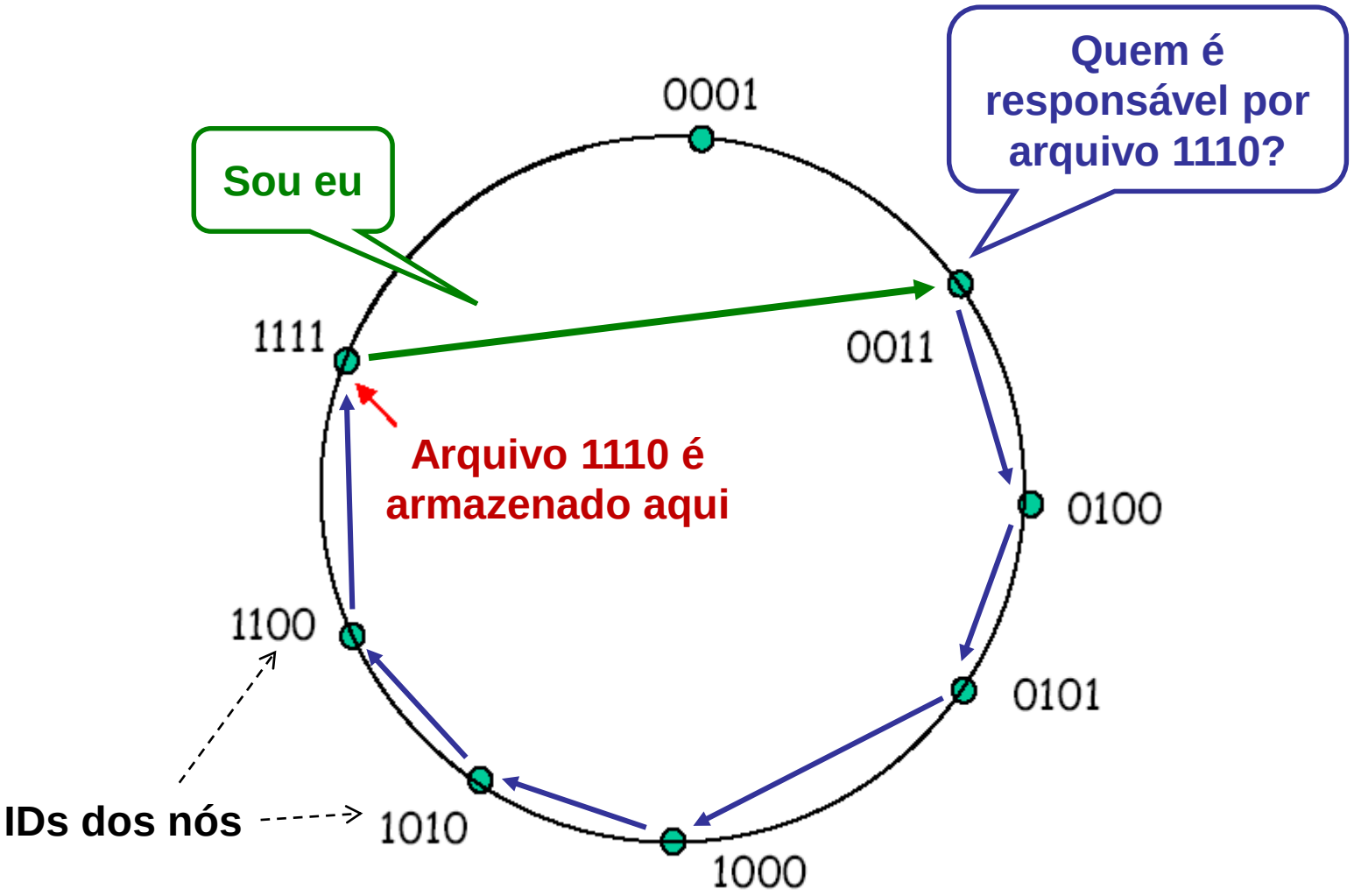


DHT: Hashing consistente

- **Problema**: associar **arquivo a nó na rede**
- Ideia: calcular hash **h** do arquivo (chave) e associá-lo ao **nó "mais próximo"** de **h**
  - Usando alguma métrica de proximidade

- Uma abordagem possível: rede sobreposta é um círculo
  - Cada nó tem um ID escolhido aleatoriamente: hash (end. IP)
  - Chaves no mesmo espaço de IDs
  - Sucessor de nó A: nó cujo ID seja imediatamente maior que ID_A
  - Cada nó sabe IP de seu sucessor

DHT: Hashing consistente



DHT: Hashing consistente

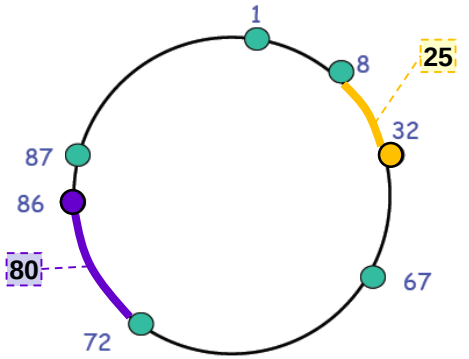
- Funcionamento básico

Saída de um nó	**Entrada de um nó**
• Se o sucessor de A deixa a rede, A precisa escolher próximo sucessor • **Logo:** cada nó mantém referência para **s ≥ 2 sucessores** • Quando o sucessor de A deixa a rede, A pede a seu novo sucessor a sua lista de sucessores; A então atualiza sua própria lista de **s** sucessores	• Você é um novo nó, e seu ID é **k** •Peça a qualquer nó **N** para encontrar o nó **N'** que é sucessor de **k** • Obtenha sua lista de sucessores de **N'** •Diga a seus predecessores que atualizem suas listas de sucessores • **Logo:** cada nó deve saber quem é seu **predecessor**

- #vizinhos = s+1 = O(1)
  - Tabela de roteamento: (ID_vizinho, IP_vizinho)
- Número médio de mensagens para encontrar chave: O(n)

Podemos fazer melhor?!

DHT: Chord

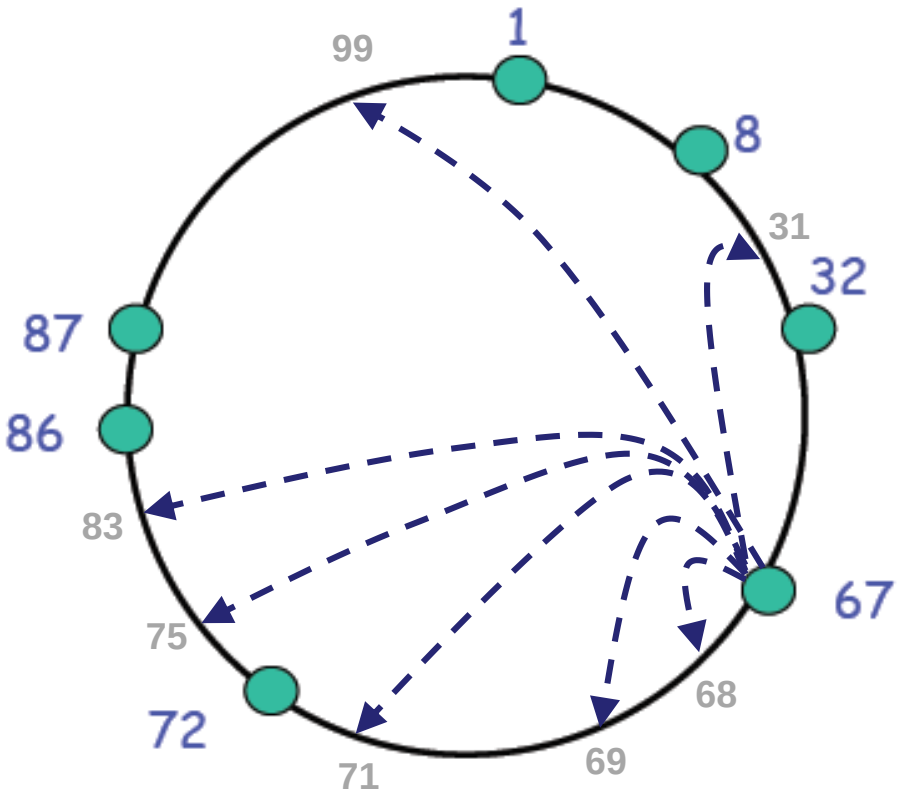


- Baseado em hashing consistente
  - Rede organizada de forma circular
  - ID aleatório unidimensional no espaço de hashes
  - Cada nó responsável por IDs na região que antecede seu ID
    -] ID_anterior , ID_próprio] (mod |espaço de IDs|)
- Tabela de derivação (*finger table*)
  - Conjunto de vizinhos conhecidos, em passos exponenciais
  - O i-ésimo vizinho (sentido horário) do nó de ID **n** tem o ID mais próximo de (e é maior que) $\mathbf{n+2^i}$ (mod |espaço de IDs|), $i \geq 0$
- Roteamento
  - Para alcançar o nó responsável pelo ID **n'**, envie a mensagem para o vizinho número $\mathbf{\log_2(n' - n)}$

DHT: Chord

- Finger table para nó 67 ➔ n=67
- Espaço de IDs de [0, 99] ➔ N = 100

i	$(n+2^i) \bmod N$	Finger table
0	68	
1	69	
2	71	
3	75	
4	83	
5	99	
6	31	
7	95	x



DHT: Chord

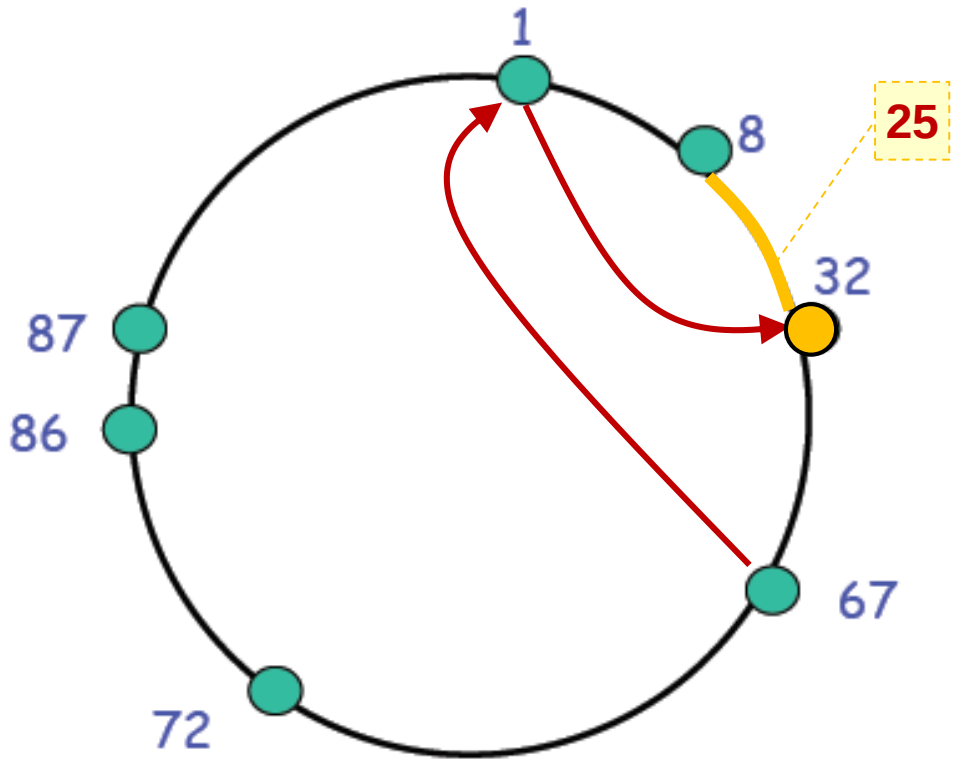
- Roteamento: busca de K = 25
  - **Nó 67**: destino = $\log_2((25 - 67) \bmod 100) = \log_2(58) \sim 5$
  - **Nó 1**: destino = $\log_2((25 - 1) \bmod 100) = \log_2(24) \sim 4$

n = 67

i	Finger table
0	72
1	72
2	72
3	86
4	86
5	1
6	32

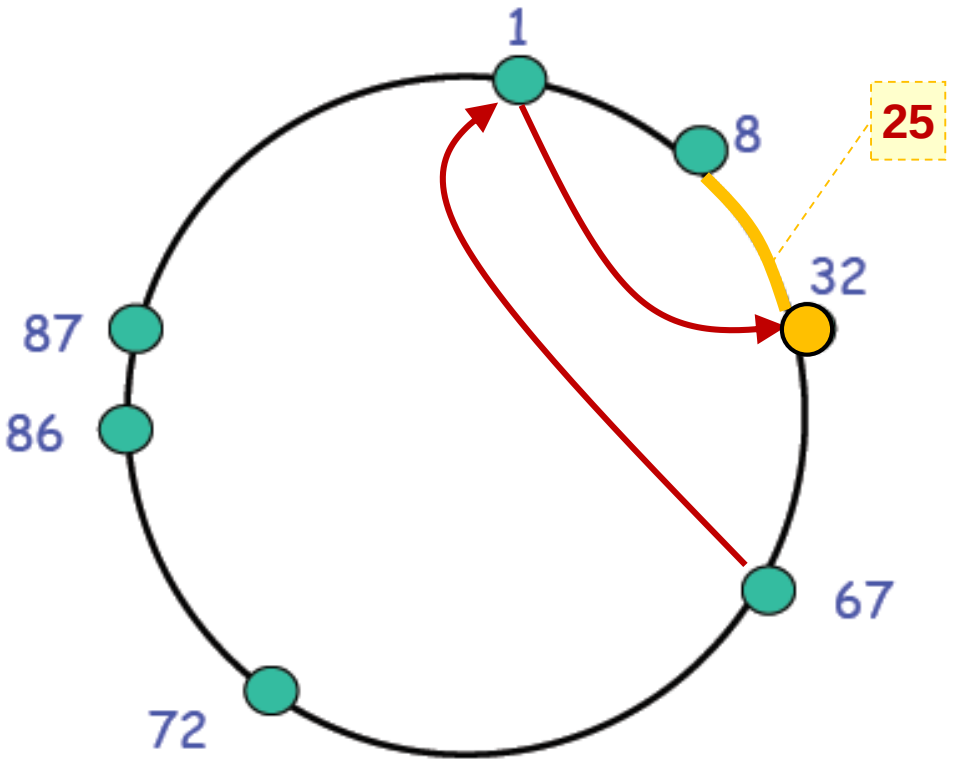
n = 1

i	Finger table
0	8
1	8
2	8
3	32
4	32
5	67
6	67



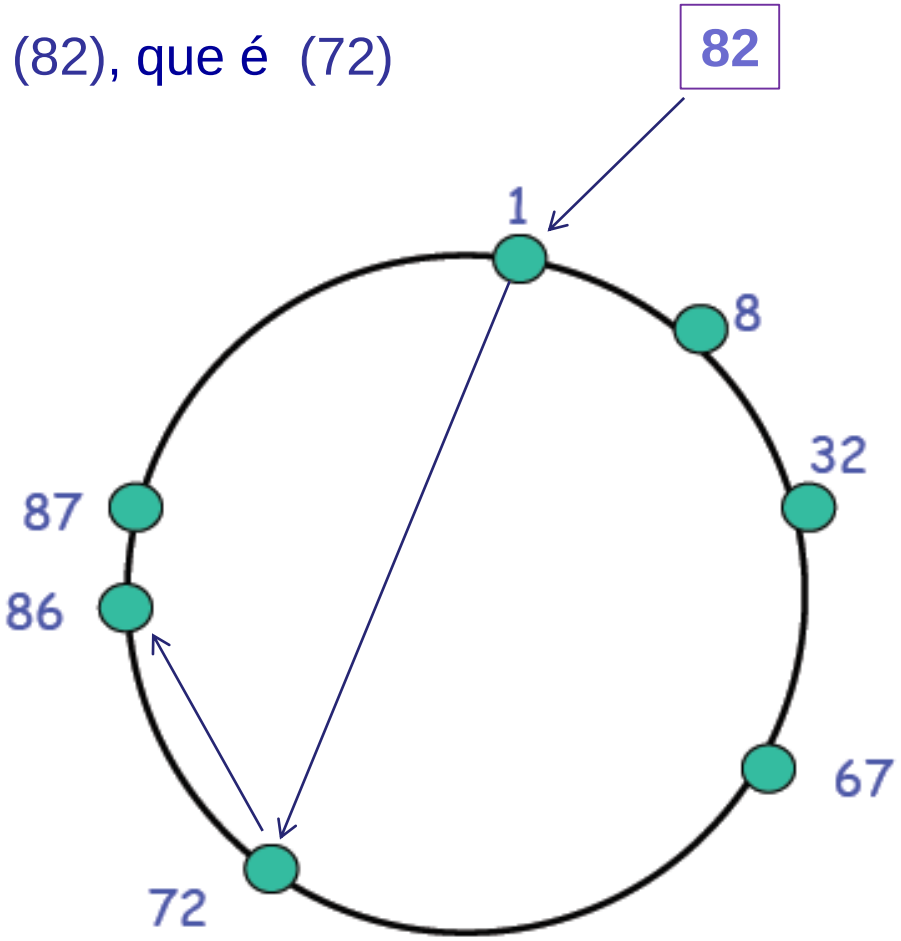
DHT: Chord

- Roteamento: qualquer nó pode ser alcançado a partir de qualquer outro nó em O (lg N)
  - N: número máximo de nós da rede



DHT: Chord

- Entrando na rede:
  - Nó (82) conhece nó (1), obtido fora de banda
  - (1) encontra predecessor de (82), que é (72)

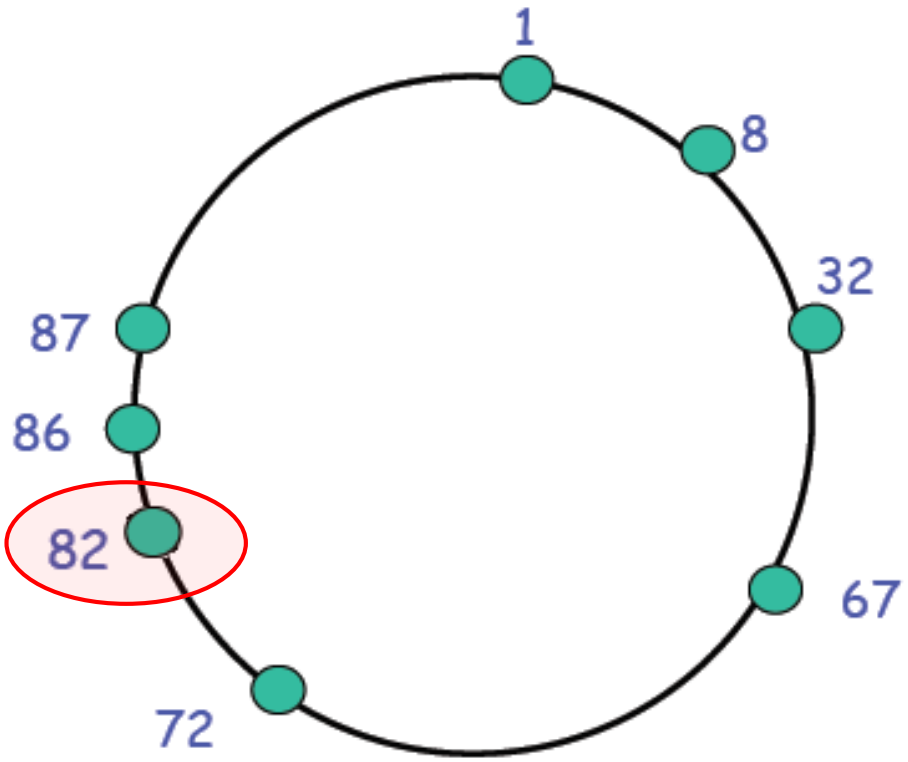


DHT: Chord

- Entrando na rede:
  - Nó (82) conhece nó (1), obtido fora de banda
  - (1) encontra predecessor de (82), que é (72)
  - (72) constrói finger table de (82)

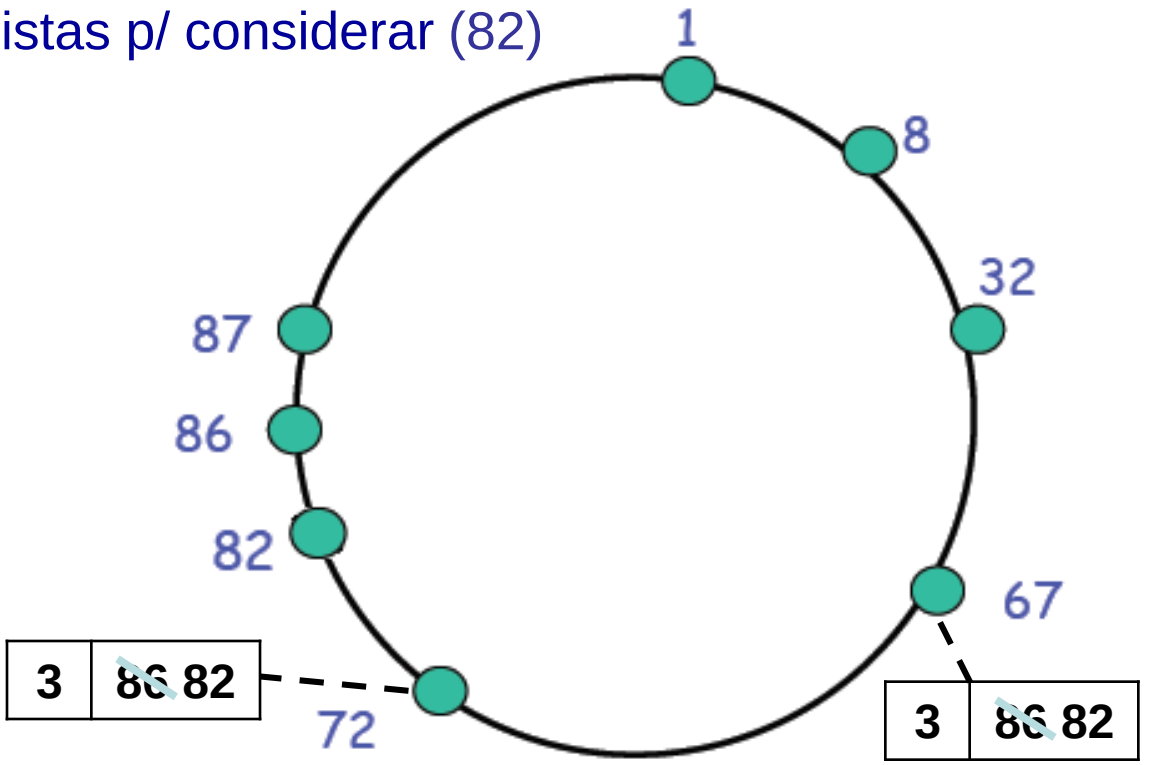
n = 82

i	$(n+2^i) \bmod N$	Finger table
0	68	72
1	69	72
2	71	72
3	75	86
4	83	86
5	99	1
6	3	8



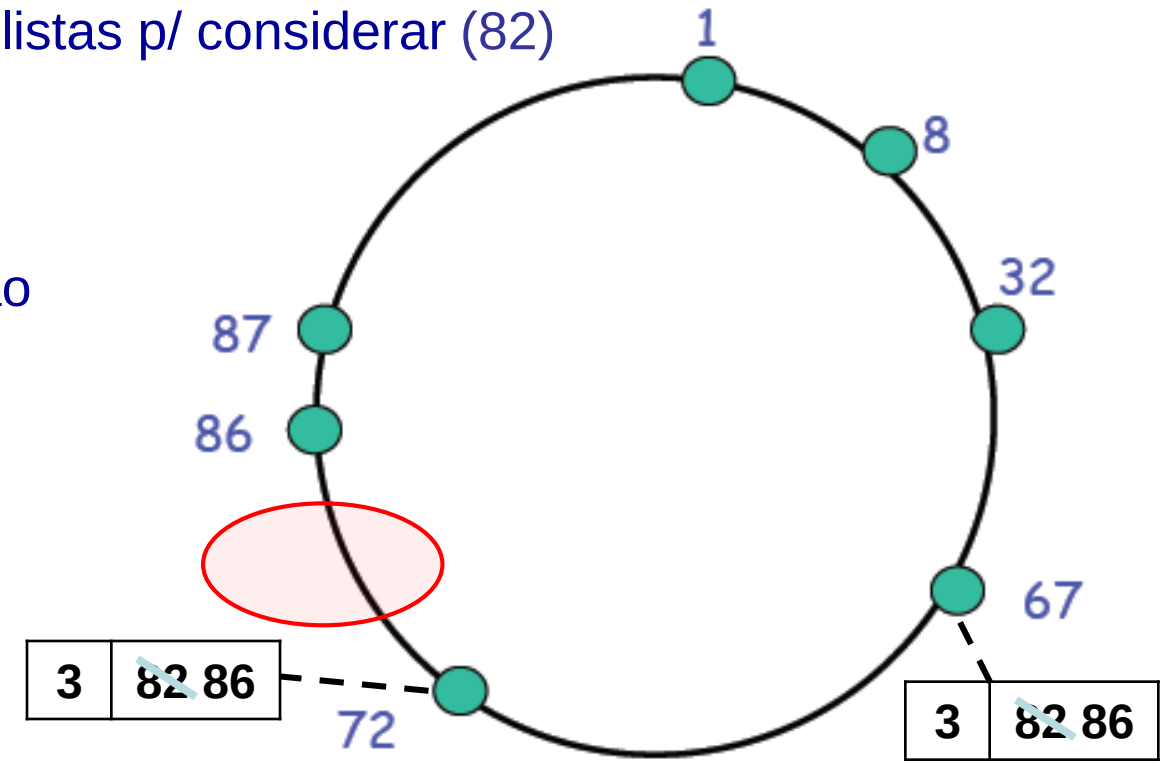
DHT: Chord

- Entrando na rede:
  - Nó (82) conhece nó (1), obtido fora de banda
  - (1) encontra predecessor de (82), que é (72)
  - (72) constrói finger table de (82)
  - Nós atualizam suas listas p/ considerar (82)
  - Atualização: $\log_2(N)$



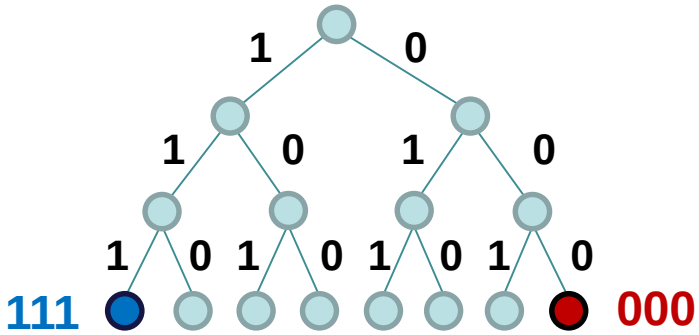
DHT: Chord

- Entrando na rede:
  - Nó (82) conhece nó (1), obtido fora de banda
  - (1) encontra predecessor de (82), que é (72)
  - (72) constrói finger table de (82)
  - Nós atualizam suas listas p/ considerar (82)
  - Atualização: $\log_2(N)$
- Saindo da rede:
  - Atualização após não receber resposta a mensagem de "keep-alive"



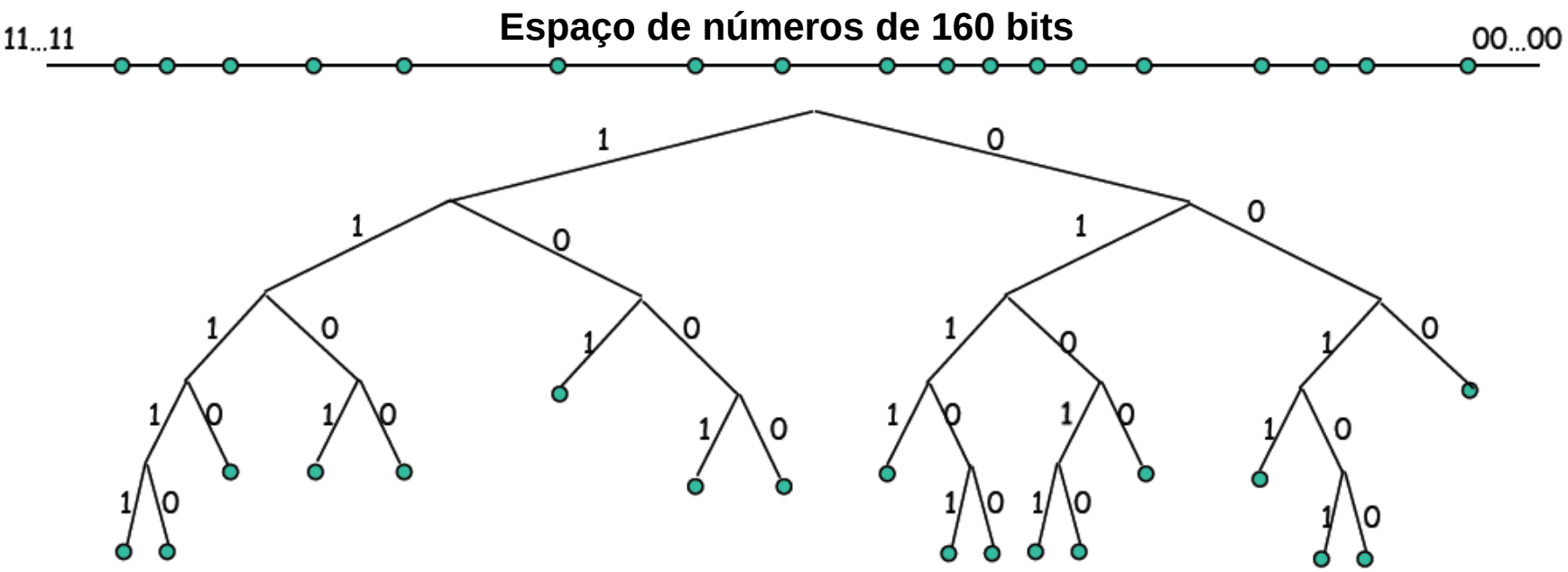
Kademlia

- Algoritmo de DHT criado em 2002
  - E ainda um dos mais usados (ex.: links magnéticos)
- Estrutura baseada em uma árvore binária
  - Organização dos nós e chaves dos arquivos



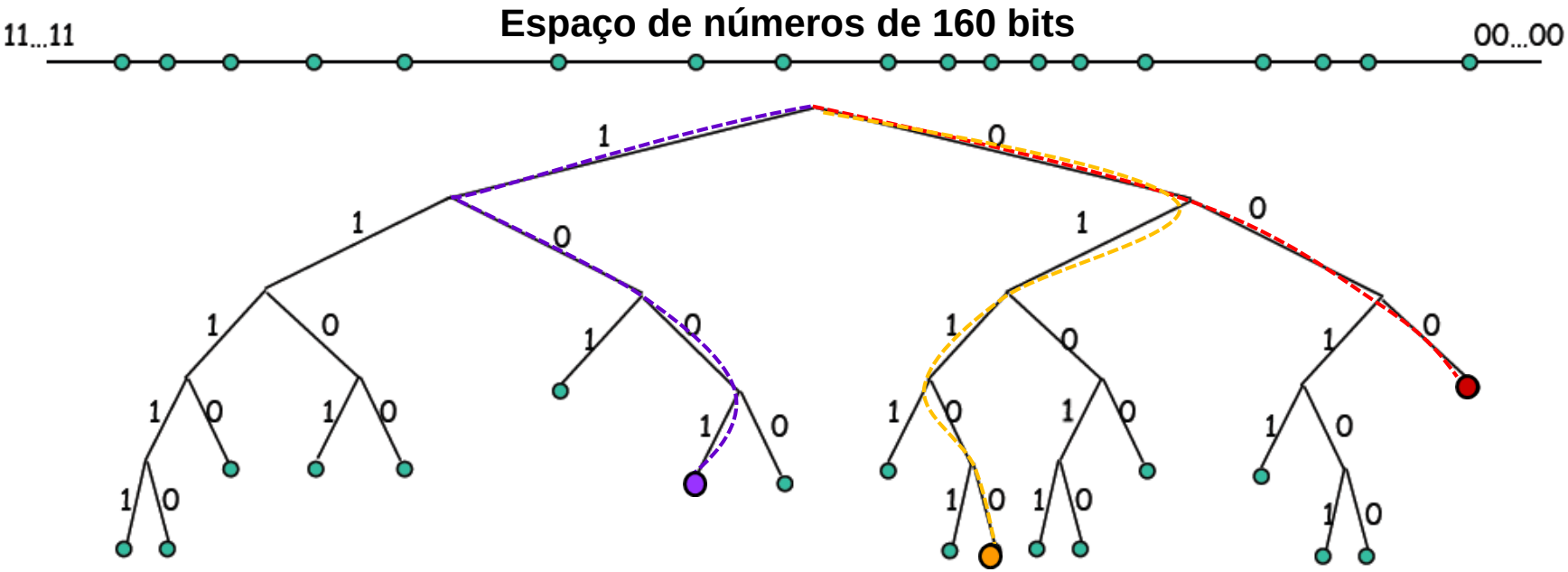
- S/Kademlia (2008): dificulta manipulação de IDs
  - ID calculado via hash de chave pública, e deve iniciar com #zeros arbitrário (“crypto puzzle”)

Kademlia



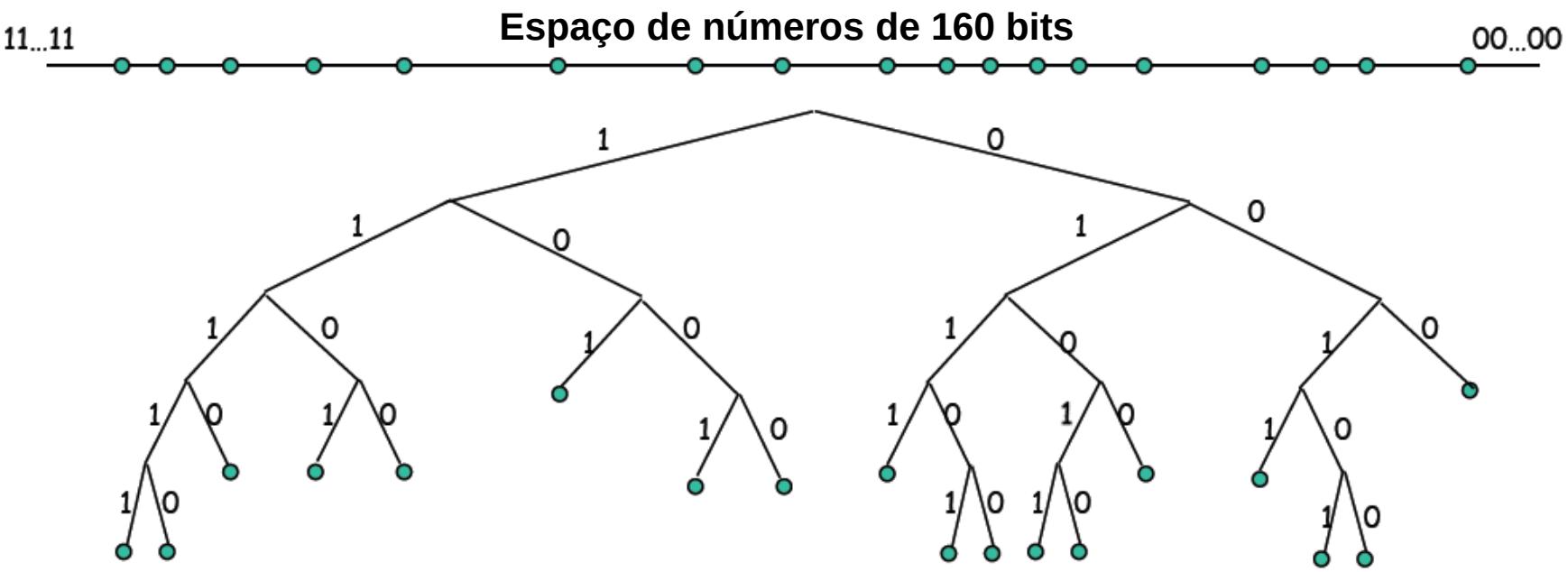
- Nós: folhas em uma árvore binária
  - Posição determinada pelo menor prefixo exclusivo de seu ID

Kademlia



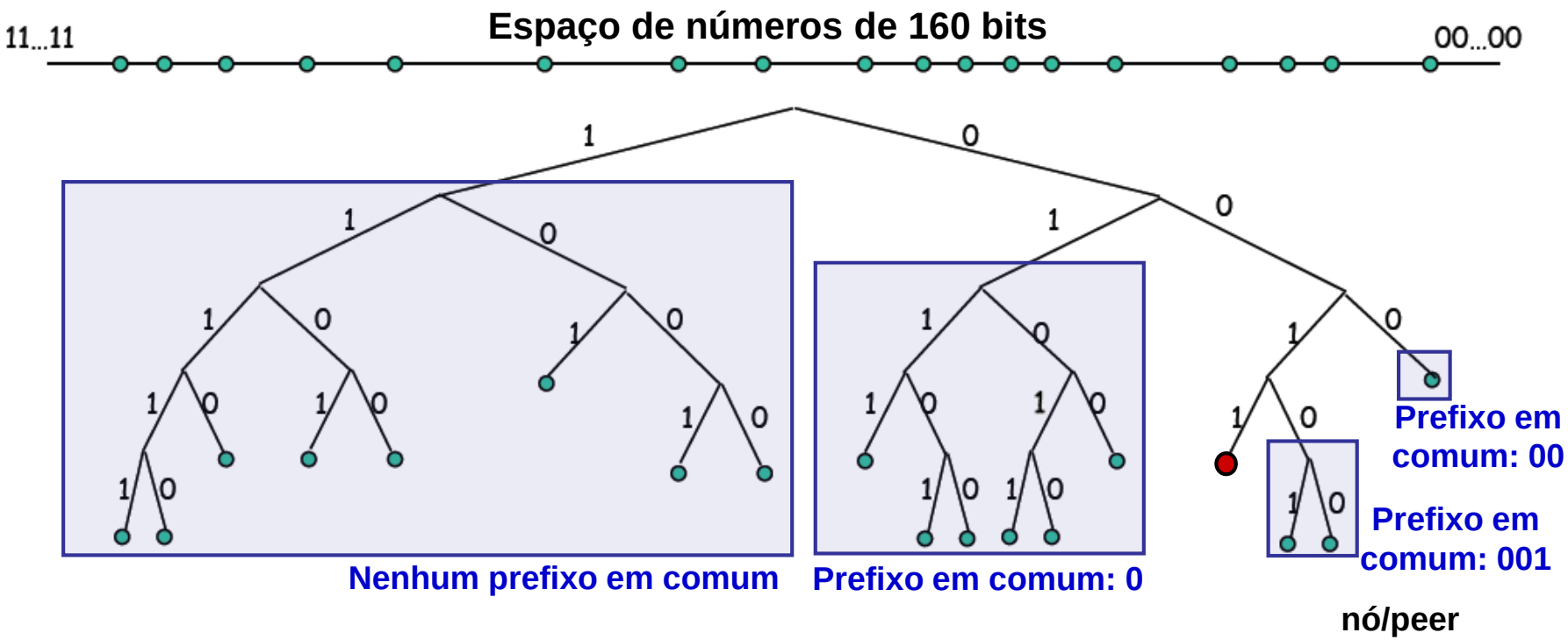
- Nós: folhas em uma árvore binária
  - Posição determinada pelo menor prefixo exclusivo de seu ID
  - Nó **000**…
  - Nó **1001**…
  - Nó **01100**…

Kademlia



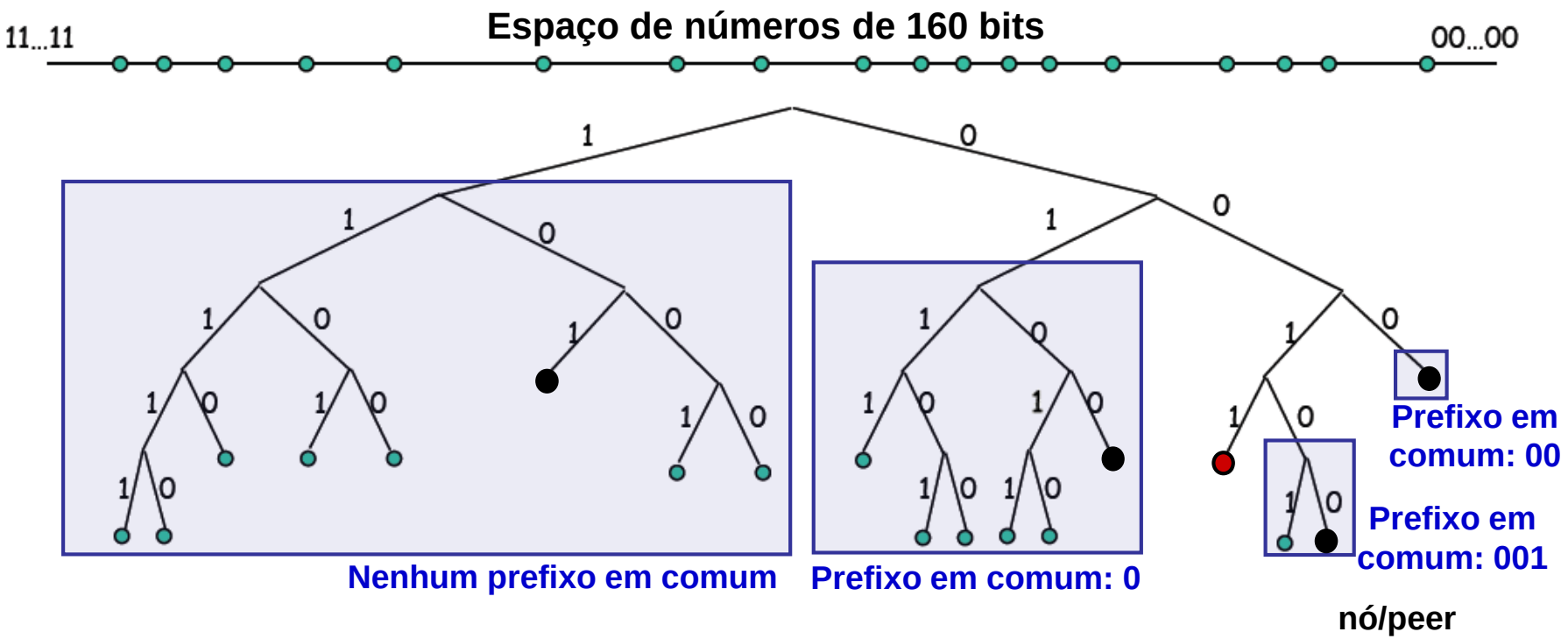
- Nós: folhas em uma árvore binária
  - Posição determinada pelo menor prefixo exclusivo de seu ID
  - Nó responsável por dados com IDs mais “próximos” a ele
  - Distância entre x e y: d(x,y) = x $\oplus$ y ➔ d(0101, 0011)=0110
  - Nós/IDs na mesma sub-árbore têm **maior prefixo em comum** ➔ estão mais próximos

Kademlia: distribuição de IDs



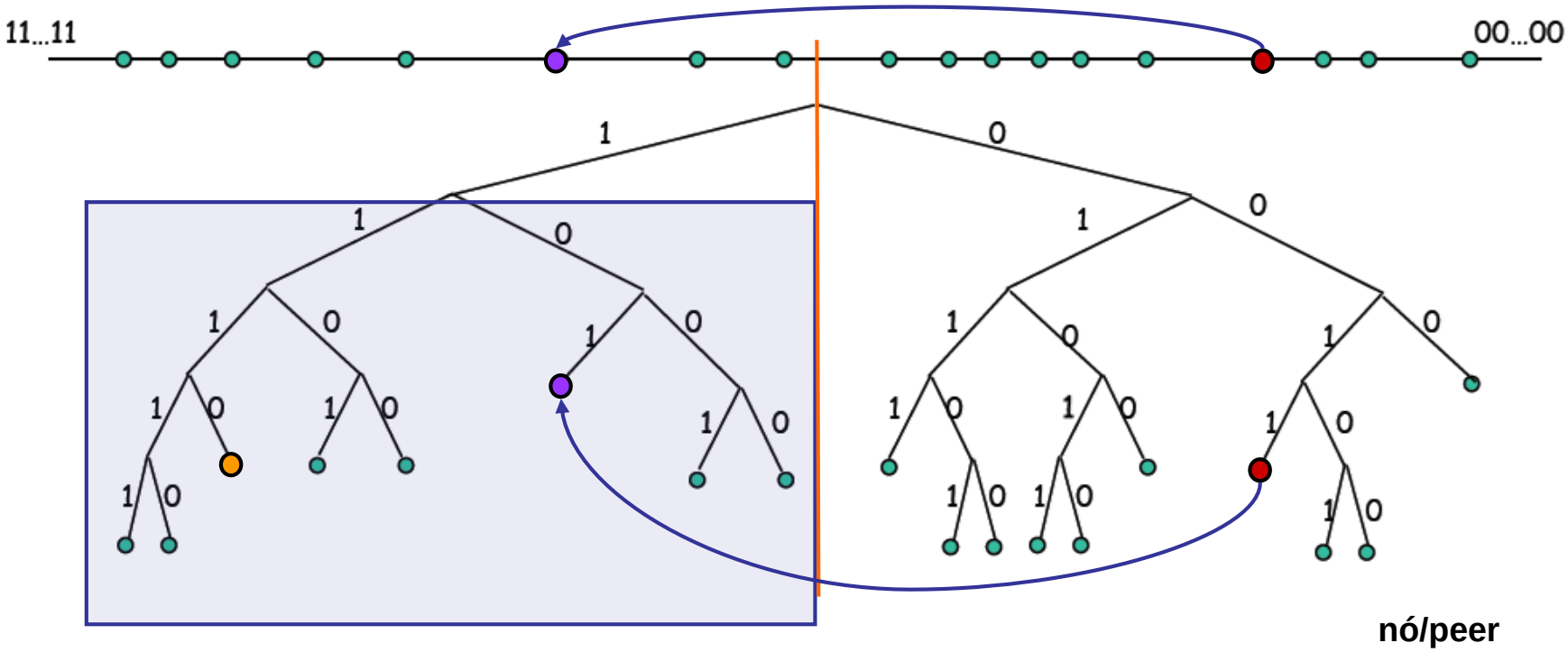
- Para um nó qualquer (ex.: nó 0011):
  - A árvore binária é dividida em sub-árvores de tamanho máximo que não contêm o nó

Kademlia: distribuição de IDs



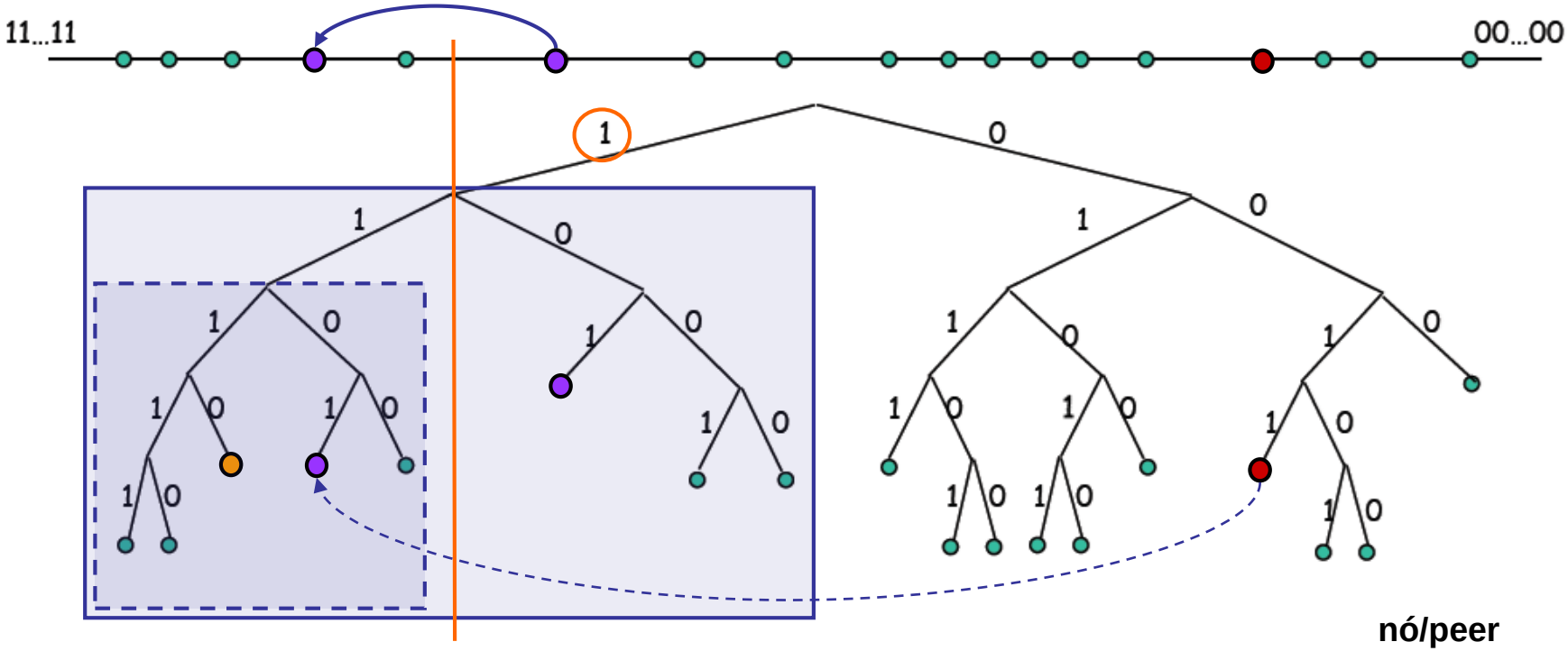
- Para um nó qualquer (ex.: nó 0011):
  - Um nó deve conhecer pelo menos um outro nó em cada uma destas sub-árvores (ex.: nó 0011 conhece nós em **preto**).
  - S/Kademlia: escolha de nós maximiza conectividade mesmo na presença de vários nós maliciosos

Kademlia: roteamento da busca



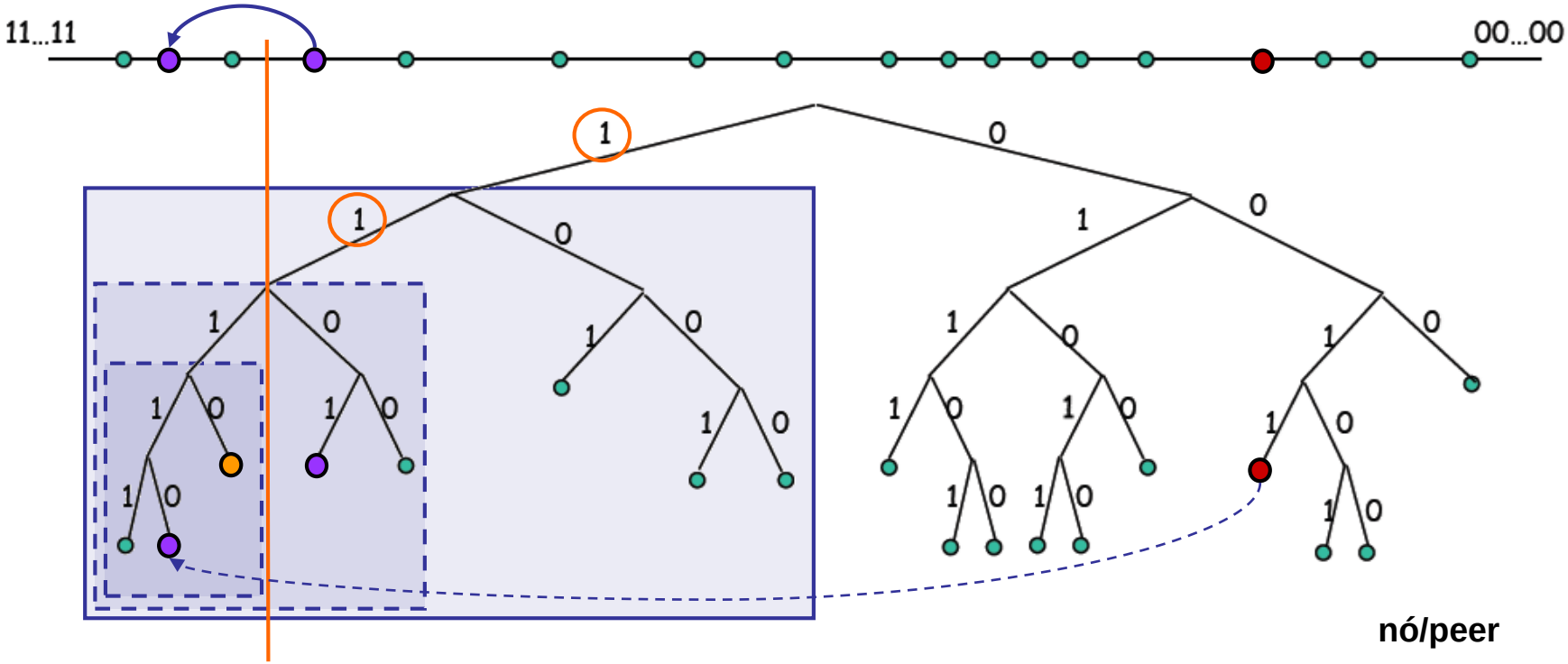
- Considere que o nó **0011100…** está procurando um dado cujo ID é **111010…**
  - Nenhum prefixo em comum: ele contacta nó na árvore correspondente (com prefixo 1: **1**01…) que roteia a busca

Kademlia: roteamento da busca



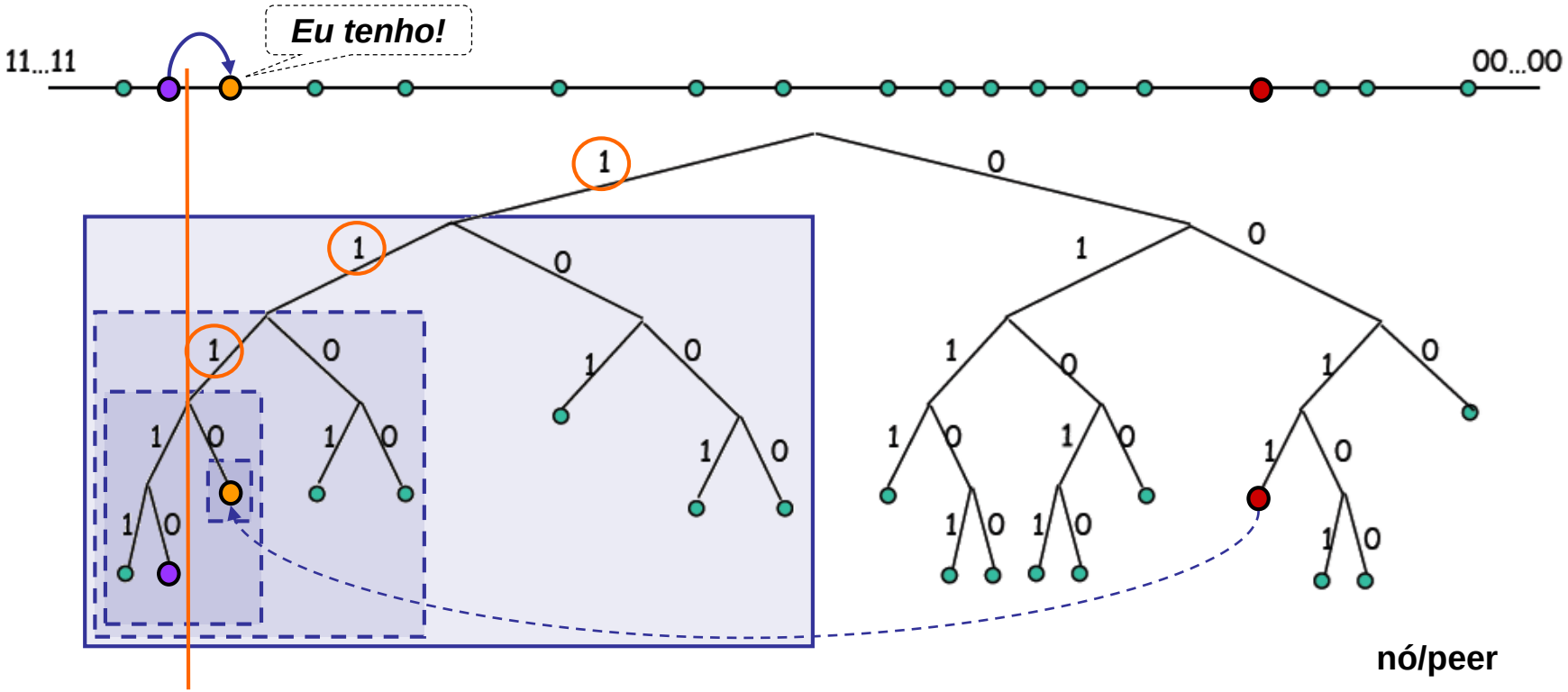
- Considere que o nó **0011100…** está procurando um dado cujo ID é **111010…**
  - Processo se repete até nó correto ser encontrado: nó **1101**…

Kademlia: roteamento da busca



- Considere que o nó **0011100…** está procurando um dado cujo ID é **111010…**
  - Processo se repete até nó correto ser encontrado: nó **11110**…

Kademlia: roteamento da busca



- Considere que o nó **0011100…** está procurando um dado cujo ID é **111010…**
  - Processo se repete até nó correto ser encontrado: nó **1110…**
  - Custo total da busca: **O(lg N)**

DHT: desafios

- **Consistência** após saída de nós
  - Manutenção de soft-state: mensagens de "**keep-alive**"
  - Equilíbrio entre consistência e overhead de comunicação
- **Desempenho**
  - Balanceamento de carga
  - Tempo de resposta na busca (otimizar #saltos).
    - Ex. (**replicação**): **a** ≥ 2 árvores Kademlia, e IDs de busca, por nó
- Reduzir **latência** no roteamento
  - Vizinhos lógicos podem estar fisicamente distantes na Internet
    - **Coral**: mecanismos de **localidade** (clusters por região e tamanho)
- **Inicialização**
  - Depende de **nó conhecido**: pode-se usar nós infraestruturados

Blockchain, Criptomoedas & Tecnologias Descentralizadas

Tecnologias descentralizadas:

Buscas e DHT

Prof. Dr. Marcos A. Simplicio Jr. – mjunior@larc.usp.br
Escola Politécnica, Universidade de São Paulo

Apoio financeiro: d24n.org <https://d24n.org>
ripple impact | university blockchain research initiative

USP

Referências

- Morris, R., Kaashoek, M. F., Karger, D., Balakrishnan, H., Stoica, I., Liben-Nowell, D., & Dabek, F. (2003). Chord: A scalable peer-to-peer look-up protocol for internet applications. *IEEE/ACM Transactions On Networking*, *11*(1), 17-32.
- I. Baumgart and S. Mies (2007) "S/Kademlia: A practicable approach towards secure key-based routing," *2007 International Conference on Parallel and Distributed Systems*, 2007, pp. 1-8, doi: 10.1109/ICPADS.2007.4447808.
- R. Boutaba (2008). "Peer-to-Peer networking: State of the art and research challenges". NOMS 2008 - 2008 IEEE Network Operations and Management Symposium. DOI: 10.1109/NOMS.2008.4575107
- R. Ahmed and R. Boutaba (2011). "A Survey of Distributed Search Techniques in Large Scale Distributed Systems," in *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 150-167, 2011, doi: 10.1109/SURV.2011.040410.00097.