MC714

Sistemas Distribuídos

1° semestre, 2017

Arquiteturas descentralizadas

- Arquiteturas multidivididas: conseqüência da divisão de aplicação em interface/processamento/dados.
- Em muitos ambientes, organização de aplicações cliente-servidor é feita em arquiteturas multidivididas: distribuição vertical.
  - Componentes logicamente diferentes em máquinas diferentes.
  - Relação com fragmentação vertical: tabelas de BD subdivididas em colunas e distribuídas.
  - Divisão lógica e física: cada máquina executa grupo específico de funções
- Distribuição horizontal
  - Servidor/cliente divididos em partes logicamente equivalentes.
  - Cada parte operando sobre seu próprio conjunto de dados.
  - Distribuição de carga.

Arquiteturas descentralizadas

- Peer-to-peer (P2P): distribuição horizontal.
- Não há servidor sempre ligado.
- Sistemas finais comunicam-se diretamente.
- *Peers* intermitentemente conectados e mudam de endereço.
- Ex: distribuição de arquivos (bitTorrent), streaming (KanKan), VoIP (Skype).

Cliente servidor versus P2P

- Quanto tempo para distribuir arquivo de tamanho F para N clientes: cliente-servidor versus P2P.
- Fig. 37.
- Cliente-servidor: envia sequencialmente N cópias.
  - Servidor:
    - tempo para enviar 1 cópia: F/u_s
    - tempo para enviar N cópias: NF/u_s
  - Cliente: cada cliente faz o download
    - d_{min} = taxa de download mínima entre os clientes.
    - Tempo de download do cliente mais lento: F/d_{min}

Aumenta linearmente em N

Tempo para distribuir F para N clientes usando cliente-servidor

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

Cliente servidor versus P2P

- P2P:
  - Servidor: upload de pelo menos uma cópia – tempo F/u_s
  - Cliente: cada cliente faz o download de uma cópia
    - Tempo de download do cliente mais lento: F/d_{min}
  - Clientes: download agregado de NF bits
    - Taxa máxima de upload: $u_s + \Sigma u_i$

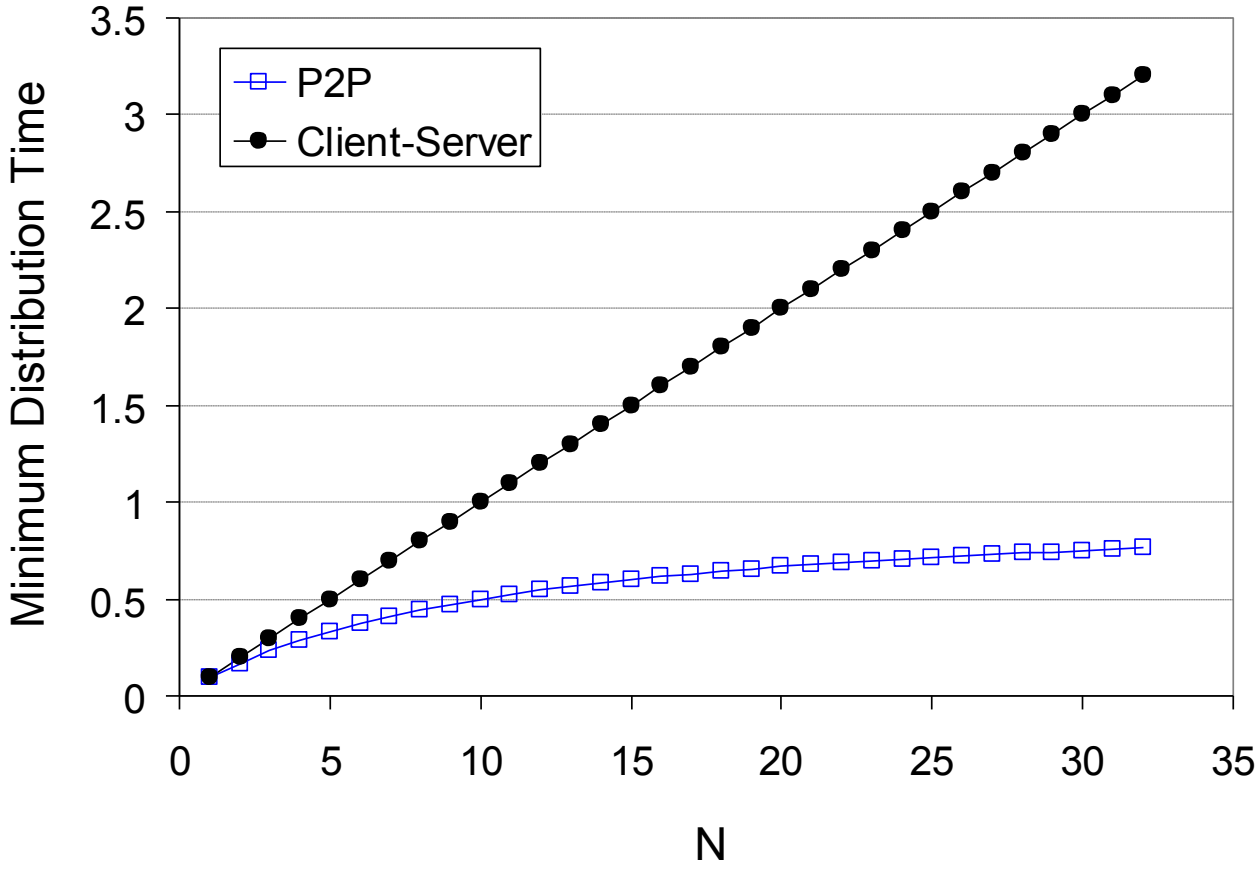
Tempo para distribuir F para N clientes usando P2P

$$D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

Aumentam linearmente em N

Cliente servidor versus P2P

- Upload cliente = u, F/u = 1 hora, $u_s = 10u$, $d_{min} \geq u_s$



Arquiteturas descentralizadas

- Peer-to-peer (P2P): distribuição horizontal.
- Processos que constituem o sistema são todos iguais.
  - Funções necessárias são executadas por todos.
  - Interação simétrica: cliente e servidor ao mesmo tempo.
- Como organizar os *peers*? Rede de sobreposição (overlay).
  - Nós são processos; enlaces são canais de comunicação lógicos.
  - Em geral, processo não pode se comunicar diretamente com outro processo arbitrário: deve obedecer overlay.
  - Redes de sobreposição: estruturadas e não estruturadas.

Arquiteturas descentralizadas

- Arquiteturas P2P estruturadas:
  - Rede de sobreposição é construída com a utilização de um procedimento determinístico.
  - Mais utilizado: tabela hash distribuída (distributed hash table – DHT).
  - Ex.: Chord, CAN, Pastry, Tapestry
- Arquiteturas P2P não estruturadas:
  - Algoritmos aleatorizados para construir a rede de sobreposição.
  - Idéia é que cada nó mantenha lista de vizinhos, mas que essa lista seja construída de modo que envolva alguma aleatorização.
  - Localização de item pode depender de inundação da rede.

Arquiteturas P2P estruturadas

- Rede de sobreposição construída com procedimento determinístico.
- Mais comum: Distributed Hash Table (DHT)
- Itens de dados recebem identificador (128, 160 bits...).
- Nós do sistema também recebem identificador, no mesmo espaço de identificadores.
- Ponto crucial: implementar um esquema eficiente e determinístico de mapeamento de chaves para identificadores de nós.
- Consulta a um item deve retornar o endereço do nó responsável pelo item.

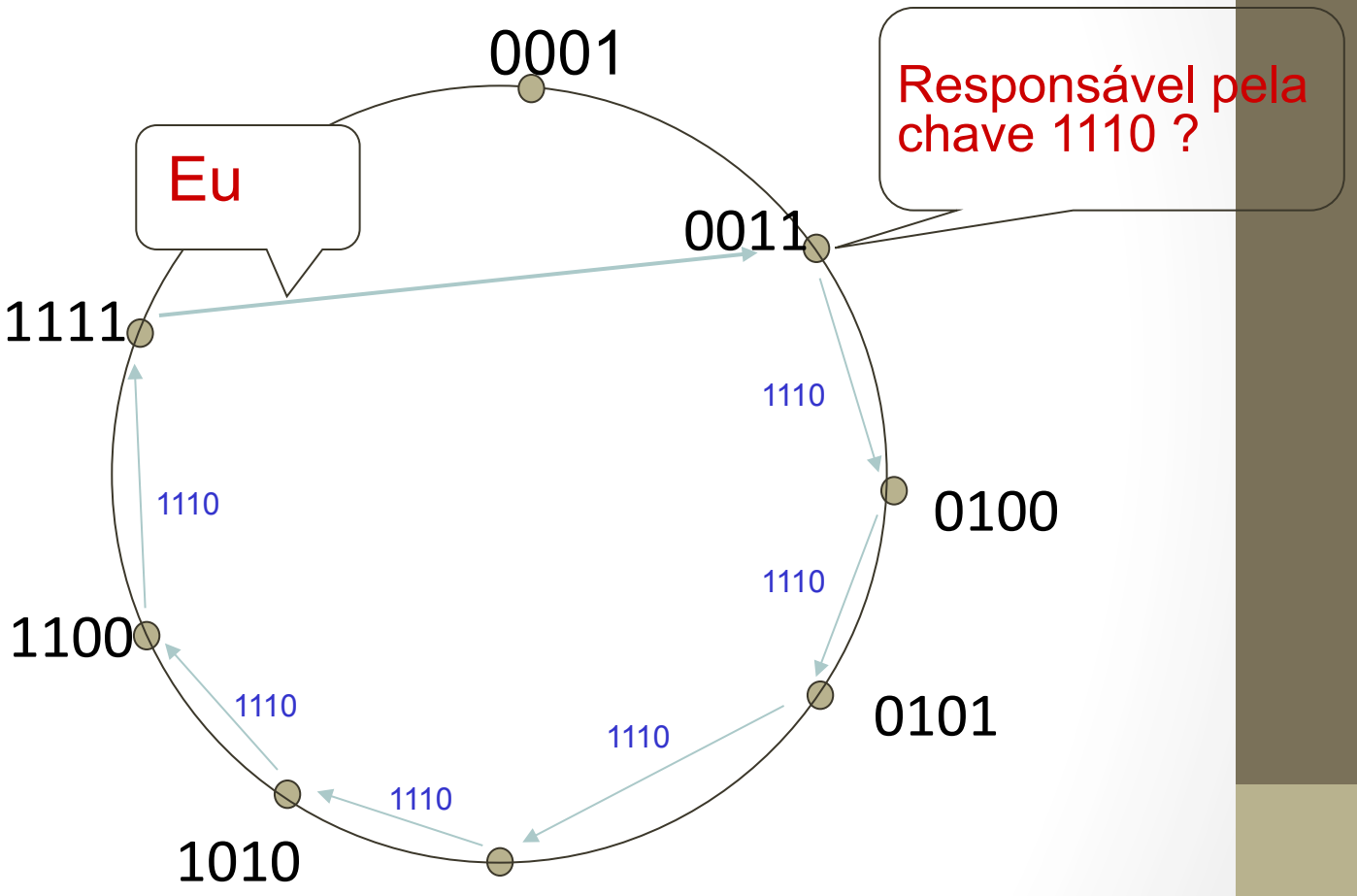
Arquiteturas P2P estruturadas

- Como atribuir chaves aos peers?
- Idéia básica:
  - Converter cada chave em um inteiro.
  - Atribuir inteiro a cada peer.
  - Colocar par (chave, valor) no peer mais próximo à chave.
- Atribuir identificador inteiro para cada peer no intervalo $[0,2^n-1]$ para algum n.
  - Cada identificador de nó tem n bits.
- Requer que cada chave esteja no mesmo intervalo.
- Para obter chave, usar função hash.
  - Ex.: chave = hash (“Pink Floyd – Dark Side of the Moon”).
  - Daí vem o nome de tabela hash distribuída.

Chord (Stoica et al., 2003)

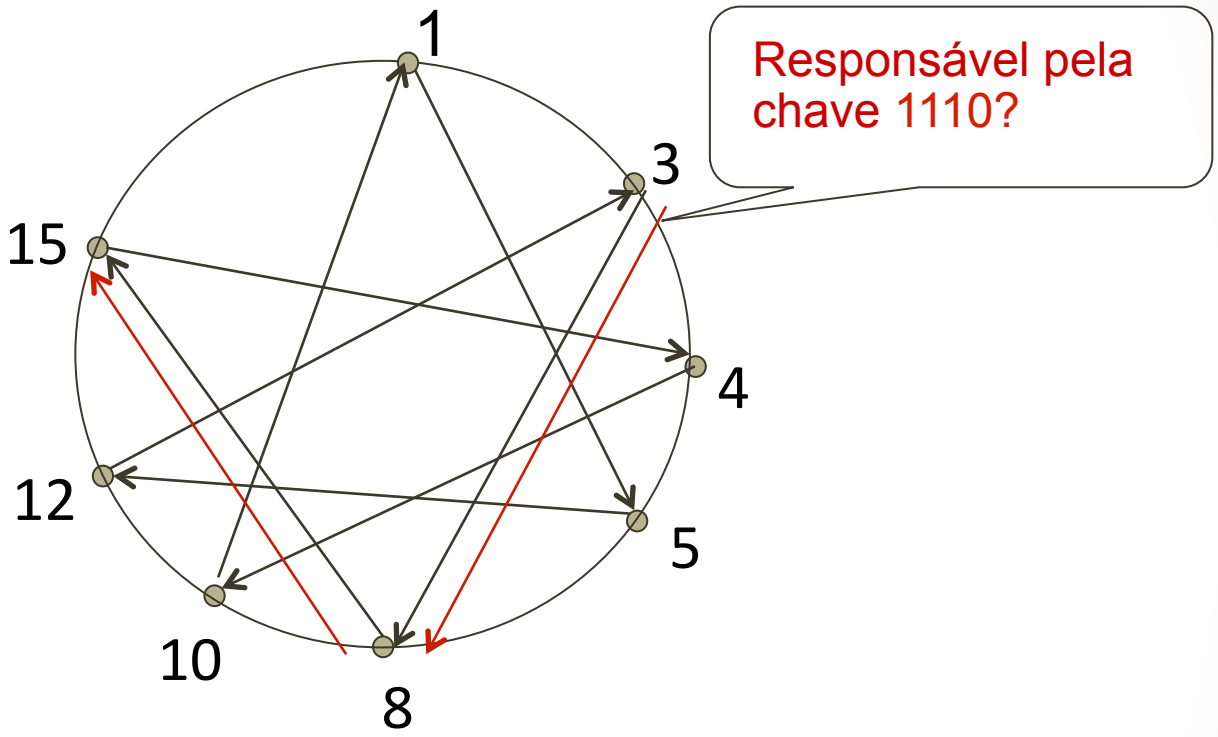
- Nós organizados logicamente em um anel - DHT circular.
- Regra: atribuir chave ao peer com ID mais próximo.
- Item de dado com chave *k* mapeado em nó com menor identificador *id* >= *k.*
  - Denominado *nó sucessor da chave k:* suc(k).
- Consulta: Lookup(k) deve retornar endereço de suc(k).
- Cada peer conhece sucessor e predecessor imediatos em uma rede de sobreposição.
- Fig. 38.
- Outras: CAN, Pastry, Tapestry, Kademlia, Ulysses, Koorde (grafos de DeBruijn)

Chord (Stoica et al., 2003)



- O(N) mensagens na média para resolver consulta (N=número de nós)

DHT Circular com atalhos



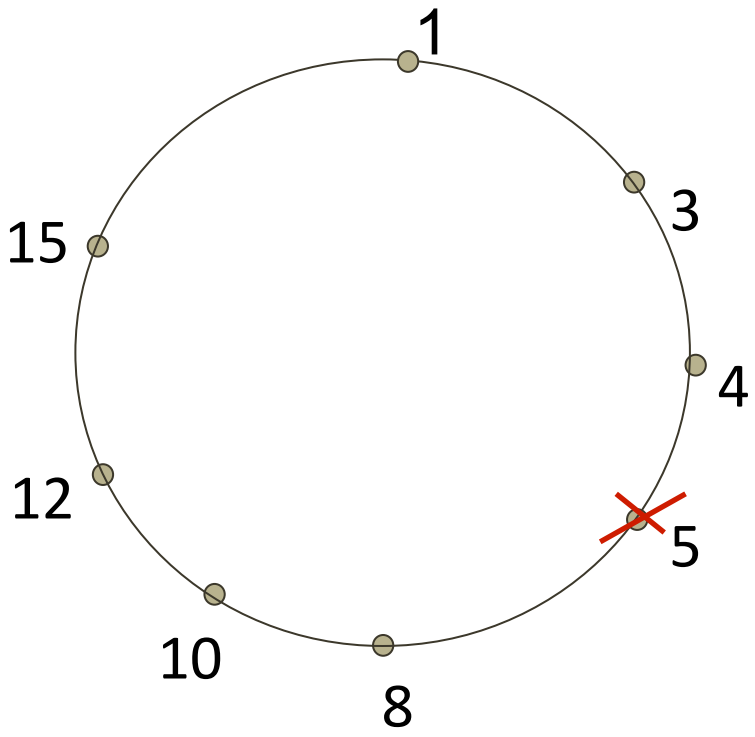
- Cada peer conhece sucessor, predecessor e atalhos.
- Redução de 6 para 2 mensagens.
- É possível desenhar atalhos para que existam O(log(N)) vizinhos, O(log(N)) mensagens em consultas.

Peer churn

- Peers entram e saem da rede (churn).
- Cada peer conhece seus dois sucessores.
- Cada peer “pinga” seus dois sucessores para verificar se continuam online.
- Se o sucessor imediato sai, realoca segundo sucessor como imediato.

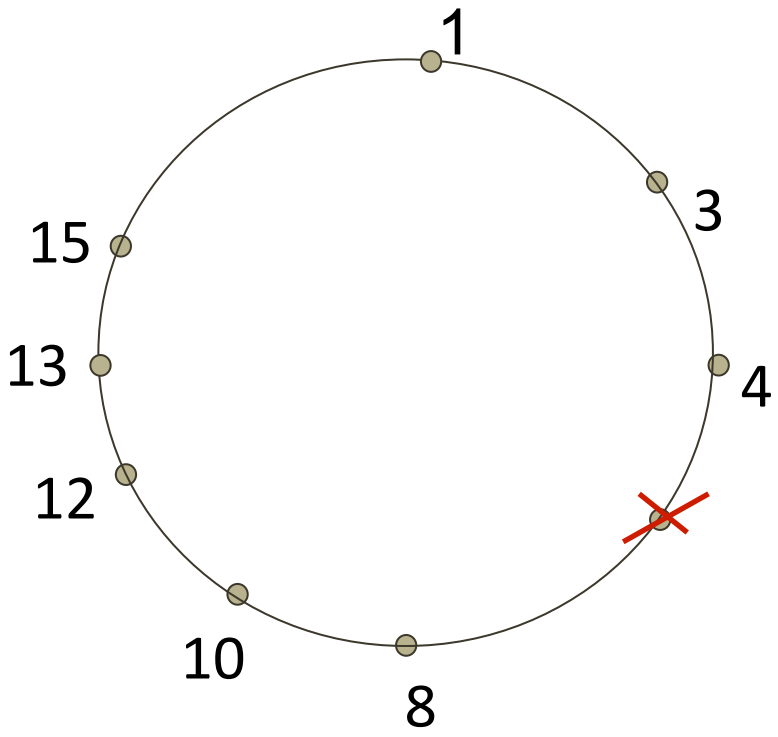
Peer churn

- Peer 5 sai.
- Peer 4 transforma 8 em seu sucessor imediato e pergunta ao 8 qual é seu sucessor.
- Peer 4 torna sucessor de 8 (10) seu segundo sucessor.



Peer churn

- Peer 13 quer entrar: gera um identificador (aleatório) *id.*
- Consulta algum nó qual ponto da rede deve entrar: quem será seu sucessor e predecessor.
- Transferência das responsabilidades de dados de 15 para 13.



CAN – Content Addressable Network

- Espaço de coordenadas cartesianas de *d* dimensões particionado entre os nós.
- Fig 48.
- Espaço bidimensional [0,1]x[0,1] dividido entre 6 nós.
- Cada nó tem uma região associada.
- Cada item de dados em CAN é atribuído um único ponto desse espaço, vinculando um nó responsável pelo dado.

CAN – Content Addressable Network

- Entrada de um nó *P* em CAN:
  - Escolhe ponto arbitrário no espaço de coordenadas;
  - Pesquisa o nó *Q* “dono” daquela região (utilizando roteamento baseado em posicionamento);
  - Nó *Q* subdivide sua região em duas metades e atribui metade a *P*;
- Nós monitoram seus vizinhos, responsáveis por regiões adjacentes.
- Na subdivisão, *P* sabe quem são seus vizinhos perguntado a *Q.*
- Itens de dados são transferidos de *Q* para *P.*
- Fig. 49

CAN – Content Addressable Network

- Saída de nó de CAN:
  - Saída do nó (0,6; 0,7).
  - Região é designada a um de seus vizinhos, por exemplo (0,9; 0,9).
  - Vizinho escolhido toma conta da região do nó que saiu.
  - Torna repartição menos simétrica: repartição do espaço inteiro por um processo em *background*.

Redes P2P não estruturadas

- Dependem, em grande parte, de algoritmos aleatorizados para construir overlay.
- Idéia: cada nó tem uma lista de vizinhos construída de modo (mais ou menos) aleatório.
- Itens podem ser colocados aleatoriamente nos nós – Balls and bins.
- Encontrar item = inundar a rede com consulta de busca.
- Rede parecida com grafo aleatório.
- Ex.: Gnutella, Freenet

Redes P2P não estruturadas

- Cada nó mantém uma lista de vizinhos vivos (visão parcial).
- Nós podem trocar regularmente entradas de suas visões parciais.
- Pode-se usar 2 threads, uma de “modo ativo” (push) e uma de “modo passivo” (pull).
  - Modo ativo: empurra entradas para peers vizinhos selecionados.
  - Modo passivo: aguarda nó enviar as entradas.
- Só ativo ou só passivo pode resultar em redes desconexas.
- É preciso também apagar entradas velhas.

Redes P2P não estruturadas

- *Pull* ou *push* isoladamente podem resultar em redes desconectadas.
  - Melhor que nós *troquem* entradas de suas visões parciais.

- Nó quer se juntar ao grupo: contata nó arbitrário, possivelmente de lista de nós bem conhecidos e com alta disponibilidade.
- Saída de nó, caso haja troca de visões parciais: nó sai sem informar qualquer nó. É removido das visões parciais dos seus vizinhos na próxima atualização.

Gerenciamento de topologia

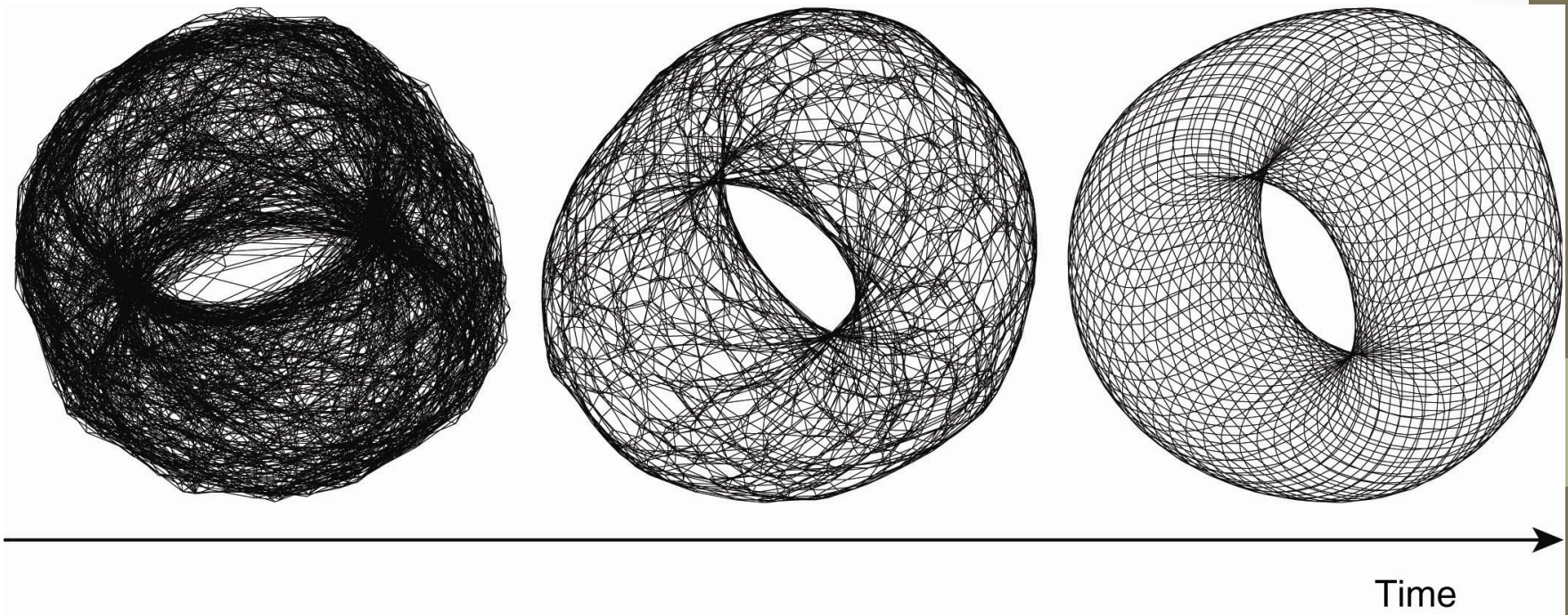
- Estruturado e não estruturado podem não ser estritamente independentes.
- Troca e seleção cuidadosa de entradas de visões parciais pode levar a topologias específicas.
- Adoção de abordagem de duas camadas.
- Fig 50.
  Camada inferior: p2p não estruturado com troca de visões parciais.
- Camada superior: seleção adicional de entradas para gerar topologia desejada.

Gerenciamento de topologia

- Por exemplo, camada superior: função de ordenação onde nós são ordenados de acordo com certo critério.
  - Ordenar conjunto de nós em ordem crescente de distância em relação a um determinado nó *P.*
  - Nó *P* gradativamente montará uma lista de seus vizinhos mais próximos, desde que a camada inferior continue enviando nós selecionados aleatoriamente.

Gerenciamento de topologia

- Jelasity e Babaoglu [2005].
- Grade lógica NxN, um nó em cada ponto.
- Lista de c vizinhos mais próximos por nó, onde distância entre nó (a1,a2) e (b1,b2) é d1+d2, com di = min (N-|ai-bi|, |ai – bi|).



Redes P2P não estruturadas

- Podem ser usadas funções de ordenação de diversas maneiras.
- Ex.: funções com captura de proximidade semântica de nós.
- Construção de redes semânticas de sobreposição.
  - Algoritmos de busca eficientes em P2P não estruturados.

Superpeers

- Busca em P2P não estruturado pode ser ineficiente e não escalável.
- Alternativa é usar superpares.
- Nós especiais que mantém um índice de itens de dados e/ou agem como intermediários.
- Outras situações convém abandonar simetria de sistemas P2P.
  - Rede colaborativa de entrega de conteúdo (Content Delivery Network – CDN) - armazenamento de páginas por nós para permitir acesso rápido a páginas próximas.
  - Nó que coleta informações sobre nós das proximidades permite seleção de quem tem recursos suficientes para armazenar conteúdo acessado.

Superpeers

- Superpares podem ser organizados em uma rede P2P → organização hierárquica.
- Fig. 39.
- Par comum conectado a superpar.
- Relação cliente-superpar fixa, em geral: conecta-se a um superpar e permanece até sair da rede.
  - Superpares: longa vida com alta disponibilidade.
- Associação fixa pode não ser melhor abordagem
  - Melhor cliente se ligar a um superpar com índices que sejam próximos ao seu interesse.
  - Garbacki et al. (2005) → nós associam-se preferencialmente a superpares que retornam um resultado de consulta para o nó.

Superpeers

- Novo problema: como selecionar nós para serem superpares.
- Estreita relação com eleição de líder em sistemas distribuídos.
- Exemplo: Skype/NAT