Algoritmos e Sistemas Distribuídos

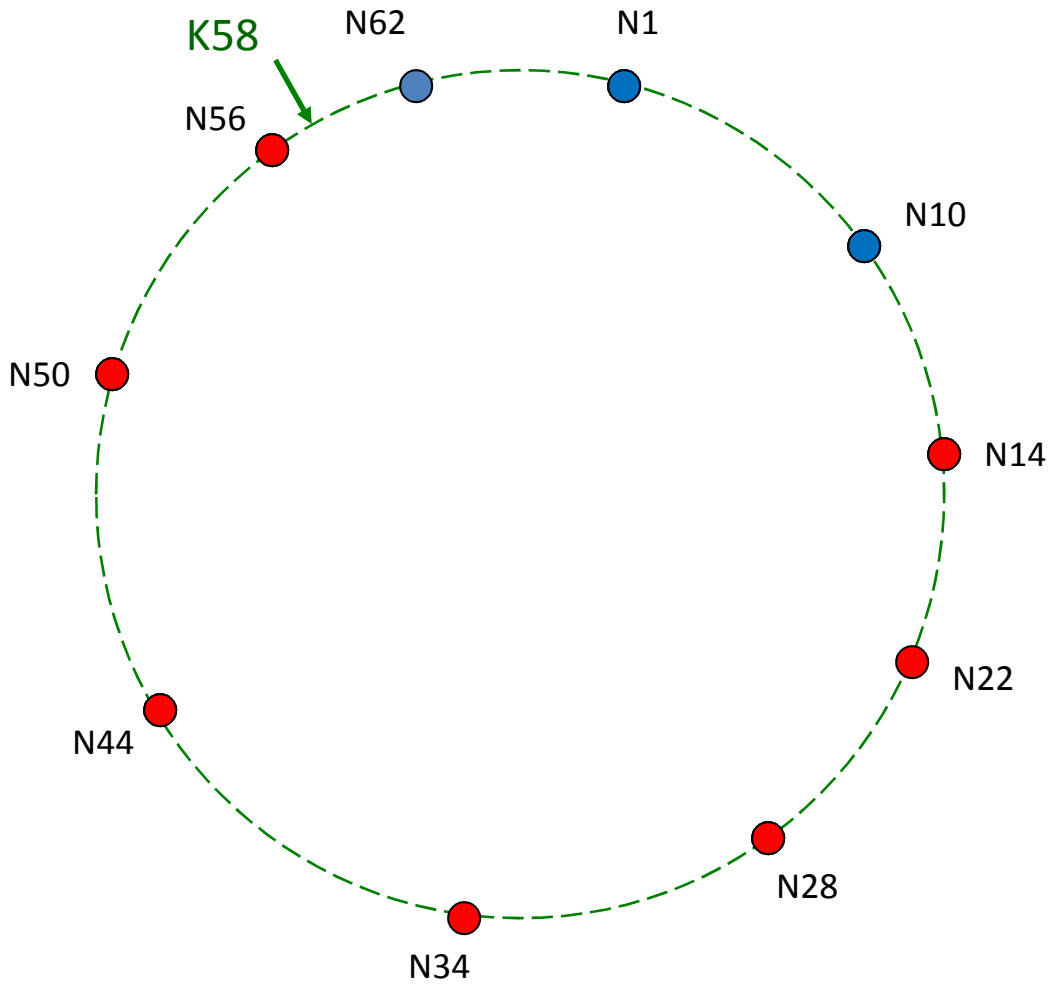
Aula teórica 10

Ano lectivo 2014/2015

Mestrado Integrado em Engenharia Informática

Última aula: Consistent hashing

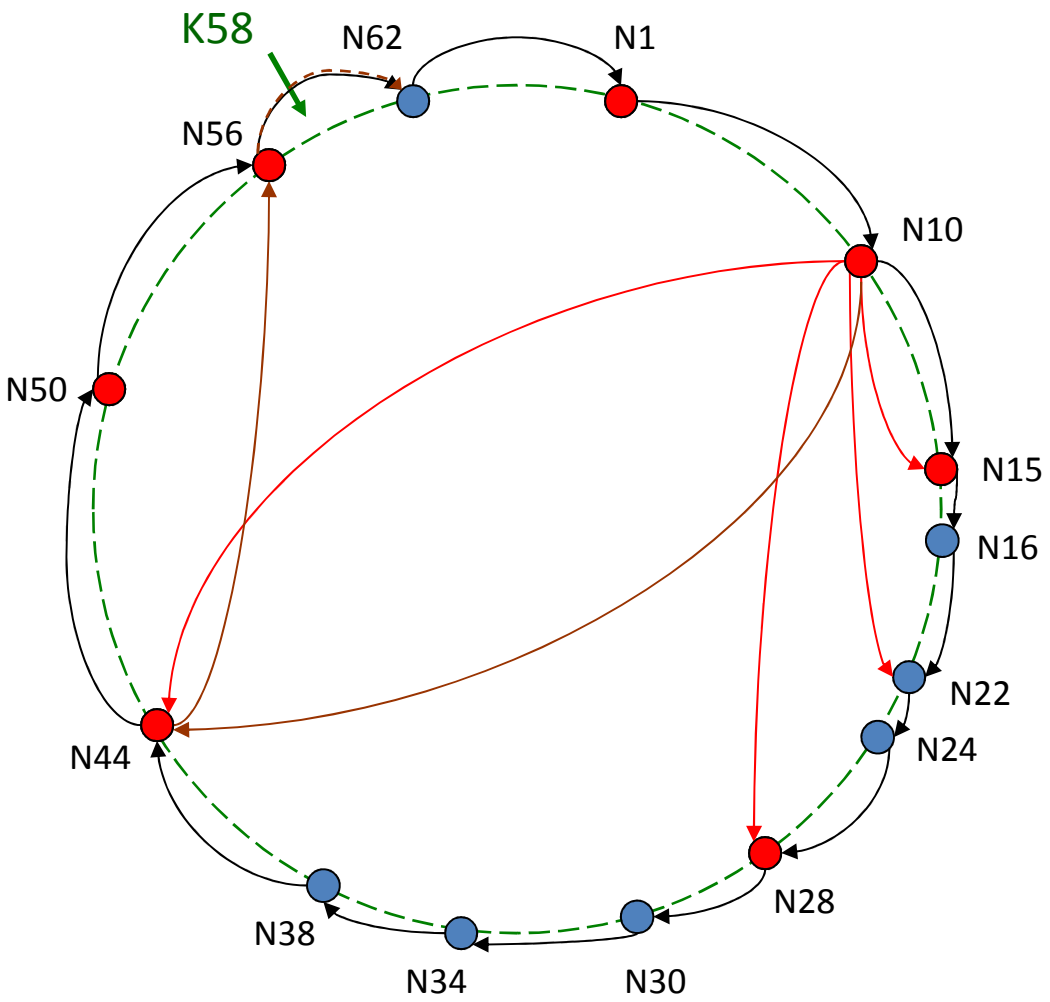
- ids estão dispostos num círculo módulo 2^m
- Chave com id k é replicada nas n réplicas que sucedem a k no círculo de ids
- Exemplo (n=3)



Chord

- **m = 6**
- **Qual a tabela de fingers do nó com id = 10?**

- **Como se pode encontrar o responsável pela chave 58?**



Entrada de novos nós

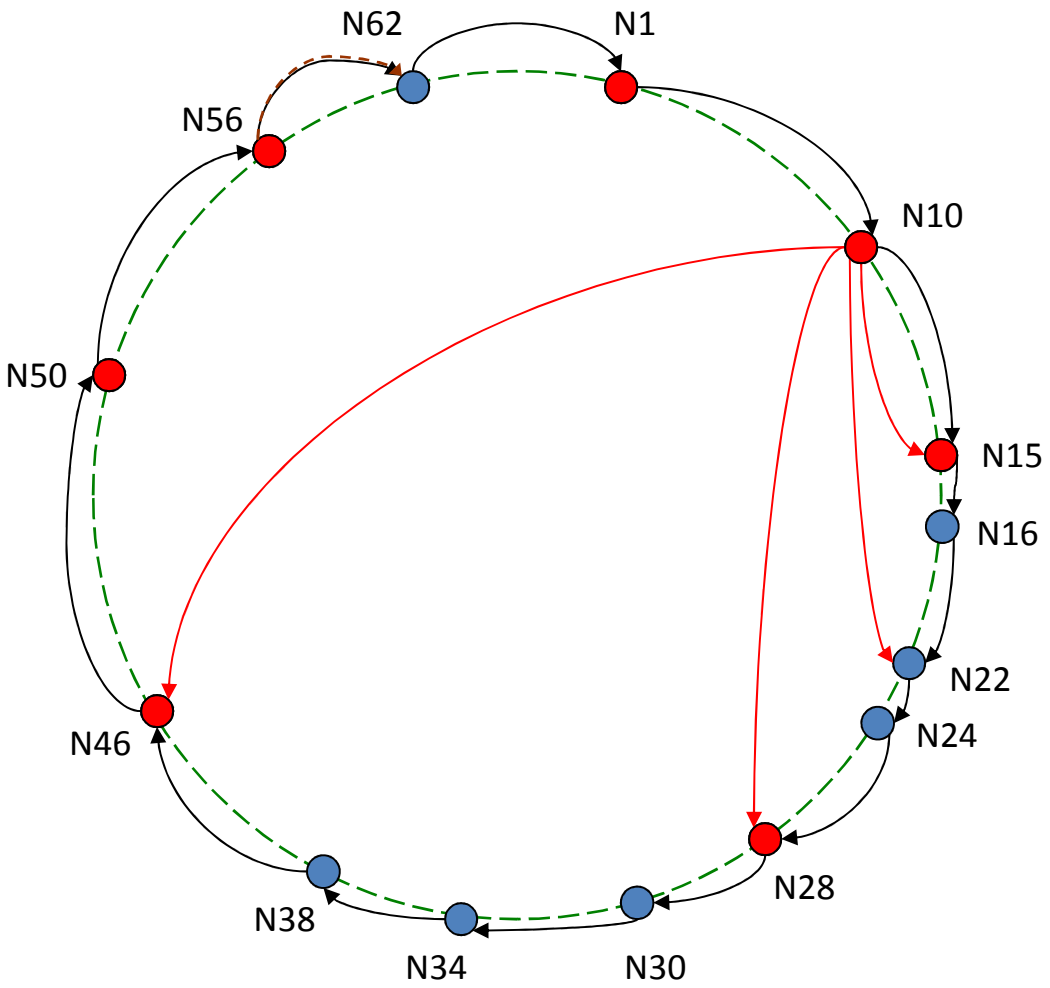
- Necessário garantir que todos os sucessores vêm a tomar os valores correctos após a entrada
  - Os fingers podem ser actualizados em background
- Para simplificar, estado de cada nó do Chord mantem um ponteiro para o antecessor
- Novo nó conhece a identidade um nó qualquer n'
- Passos necessários:
  1. Inicializar o sucessor do novo nó
  2. Todos os nós periodicamente verificam qual é o antecessor do sucessor → corrigir eventuais erros
  3. Todos os nós periodicamente actualizam os fingers

Pseudo-código da “estabilização”

```
n.join(n') {
    predecessor = nil;
    successor = n'.find_successor(n);
}
// periodically verify n's successor, and inform him of n
n.stabilize() {
    x = successor.predecessor;
    if ( x ∈ ]n,successor[ )
        successor = x;
    successor.notify(n);
}
// n' thinks it might be our predecessor.
n.notify(n') {
    if ( predecessor == nil or n' ∈ ]predecessor,n[ )
        predecessor = n';
}
// periodically refresh finger table entries.
n.fix_fingers() {
    i = random index > 1 into finger[];
    finger[i].node = find_successor(finger[i].start);
}
```

Exemplo

- **Novo nó com id = 43**
- **Qual a sequência de passos para actualizar sucessores e antecessores dos nós 38,43,46?**
- **Como é feita a actualização dos fingers do nó 10?**



Correcção do protocolo de estabilização

- Teorema: A partir de certo ponto após a última entrada de um novo nó, os ponteiros de sucessores estarão todos correctos
  - Com a possibilidade de existirem vários aneis disjuntos ou um ciclo com várias voltas ao espaço de identificadores
- Impacto de entradas nos fingers existentes é baixo:
  - Teorema: Numa rede estável com N nós, se N nós entrarem e estabilizarem os sucessores mas não os fingers, os lookups continuam com O(log N) passos

Tolerância a falhas

- Ao falhar um nó, o seu antecessor perde o ponteiro para o sucessor
- Solução para manter a correcção dos sucessores: manter uma lista de sucessores em cada nó
  - Nós são retirados desta lista quando falham
- Como lidar com um finger que falhou?

Eficácia da replicação

- Teorema: Com uma lista de sucessores de dimensão O(log N) numa rede inicialmente estável, se todos os nós falharem com probabilidade 0.5, então com elevada probabilidade o find_sucessor(k) encontra o sucessor da chave k, em O(log N) passos

Replicação dos dados

- Aplicações armazenam dados associados à chave k nos N sucessores de k
  - Tal como determinado pela lista de sucessores do antecessor de k
- Exemplo: DHT
  - get(key)
    - Lookup para encontrar N sucessores
    - Descarregar o valor da réplica mais próxima
  - put(key,value)
    - Lookup para encontrar N sucessores
    - Armazenar valor em todas as réplicas
- Necessário transferir dados à medida que a lista de sucessores altera

Coordenação em sistemas p2p

- Exemplo: nós que estão interessados num objecto querem saber quando o objecto é actualizado
  - E.g., lista de feeds RSS de interesse
  - E.g., conjunto de nós que subscrevem a uma "live stream" querem procurar um nó de quem receber a stream
- Chamados sistemas "publicador-subscritor"

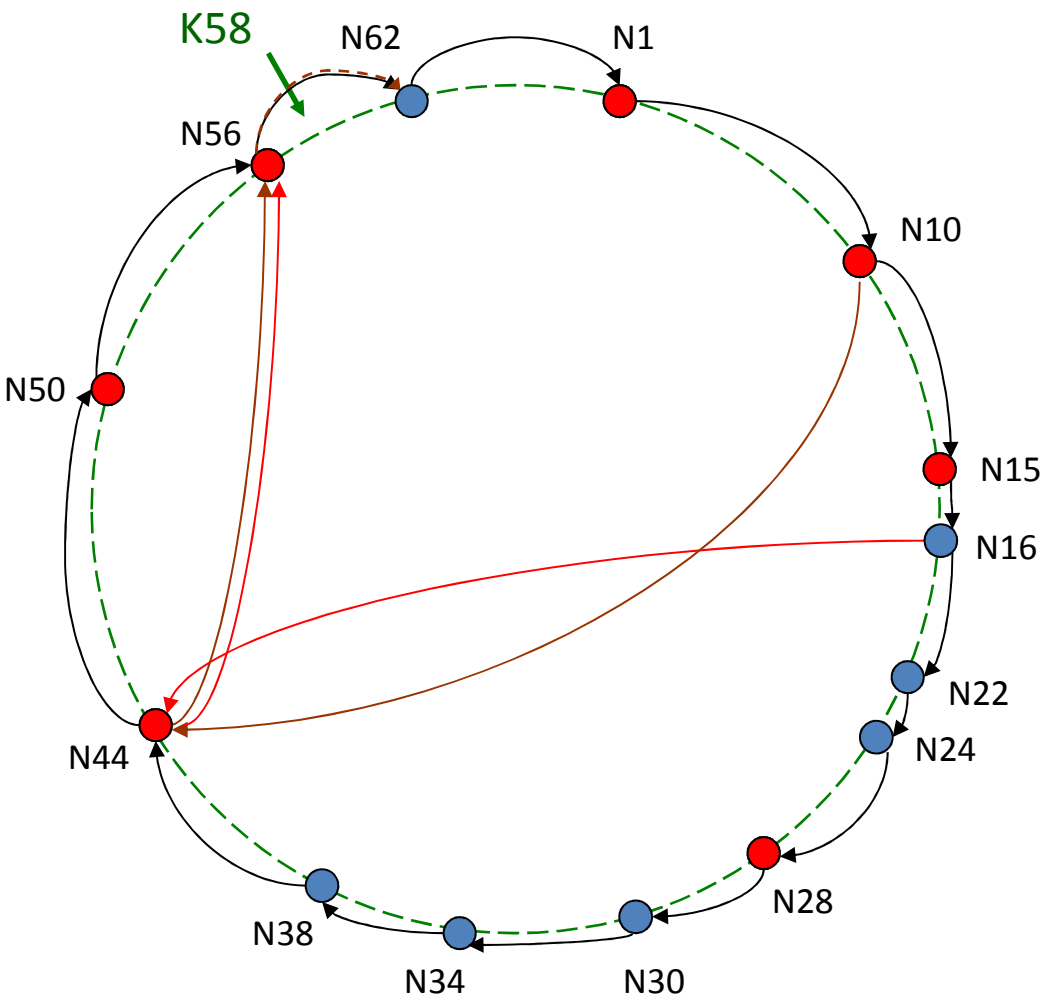
Solução: usar o lookup

- Associar chave (identificador) ao objecto, sucessor da chave é responsável por publicar as actualizações
- Problema: como fazer este sistema escalar para milhões de subscritores?
- Ideia: formar árvores para disseminar actualizações
  - (nós intermédios das árvores podem ou não ser subscritores)

Formar árvores com o lookup

- Cada nó que quer subscrever faz lookup para a chave associada ao objecto
- União dos caminhos inversos dos lookups forma uma árvore
  - Raíz é o sucessor da chave associada ao objecto

Exemplo



p2p streaming: limitações

- Supor que usávamos a árvore do slide anterior para fazer streaming de video; supor também que todos os membros estão a subscrever ao conteúdo; qual (umas das) desvantagens?
- Os nós que são folhas na árvore não contribuem para a distribuição do conteúdo
- Como resolver o problema?
- Fazer striping: construir N árvores e dividir o conteúdo pelas árvores

Desafios inerentes aos sistemas p2p

- Controlo da admissão de membros no sistema
- Integridade e disponibilidade dos dados
- Incentivos
- Auto-gestão
- Desconfiança dos ISPs