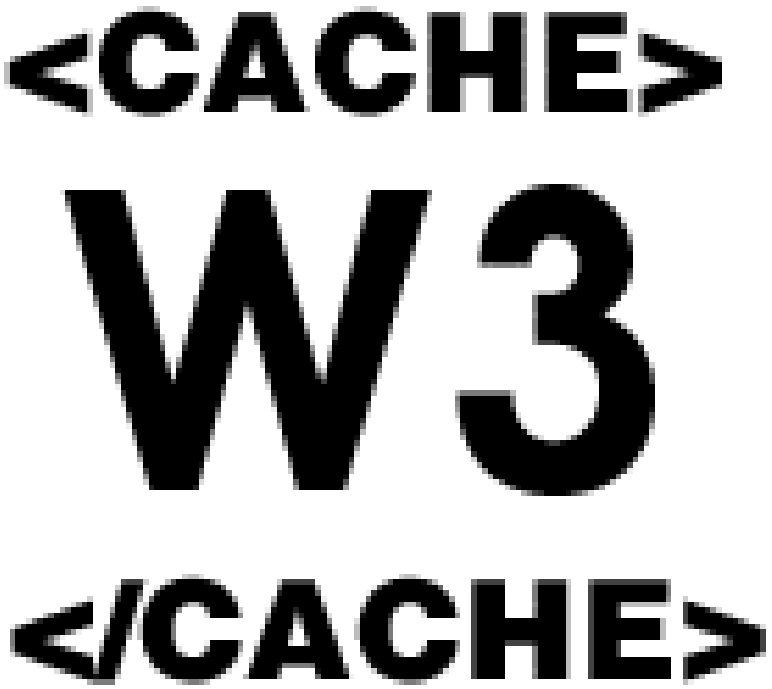
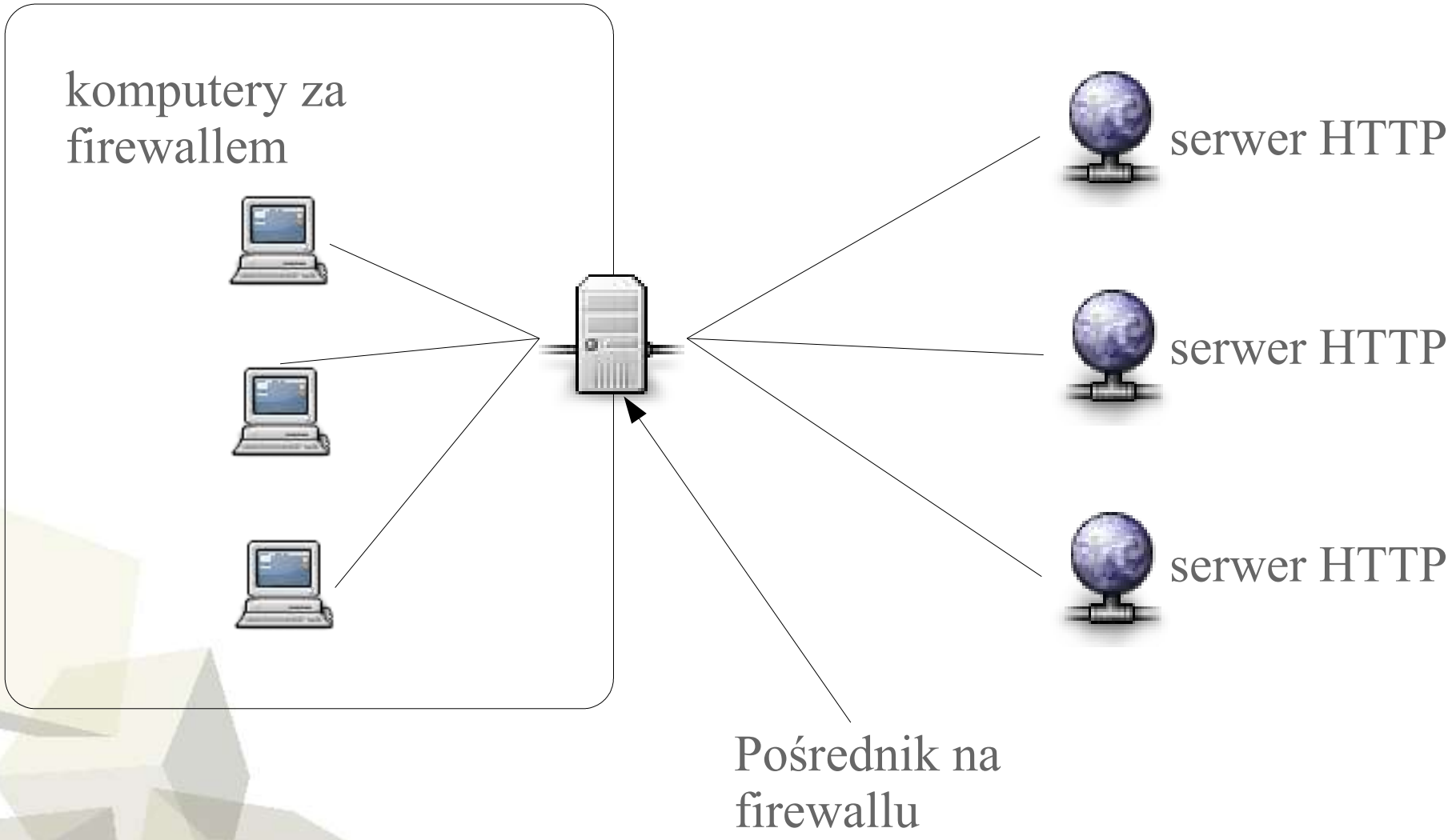
Keszowanie i systemy peer-to-peer

Paulina Kania i Łukasz Osipiuk



Pośrednik w instytucji



Pochodzenie pośrednika keszującego

- Serwery proxy w korporacjach
  - Jeden serwer dzielony przez wielu użytkowników
    - podobny zbiór pobieranych dokumentów
    - pobieranie tych samych dokumentów w krótkich okresach czasu

Dlaczego chcemy keszować?

- Ograniczamy ruch sieciowy
- Ograniczamy obciążenie serwera
- Zmniejszamy opóźnienia

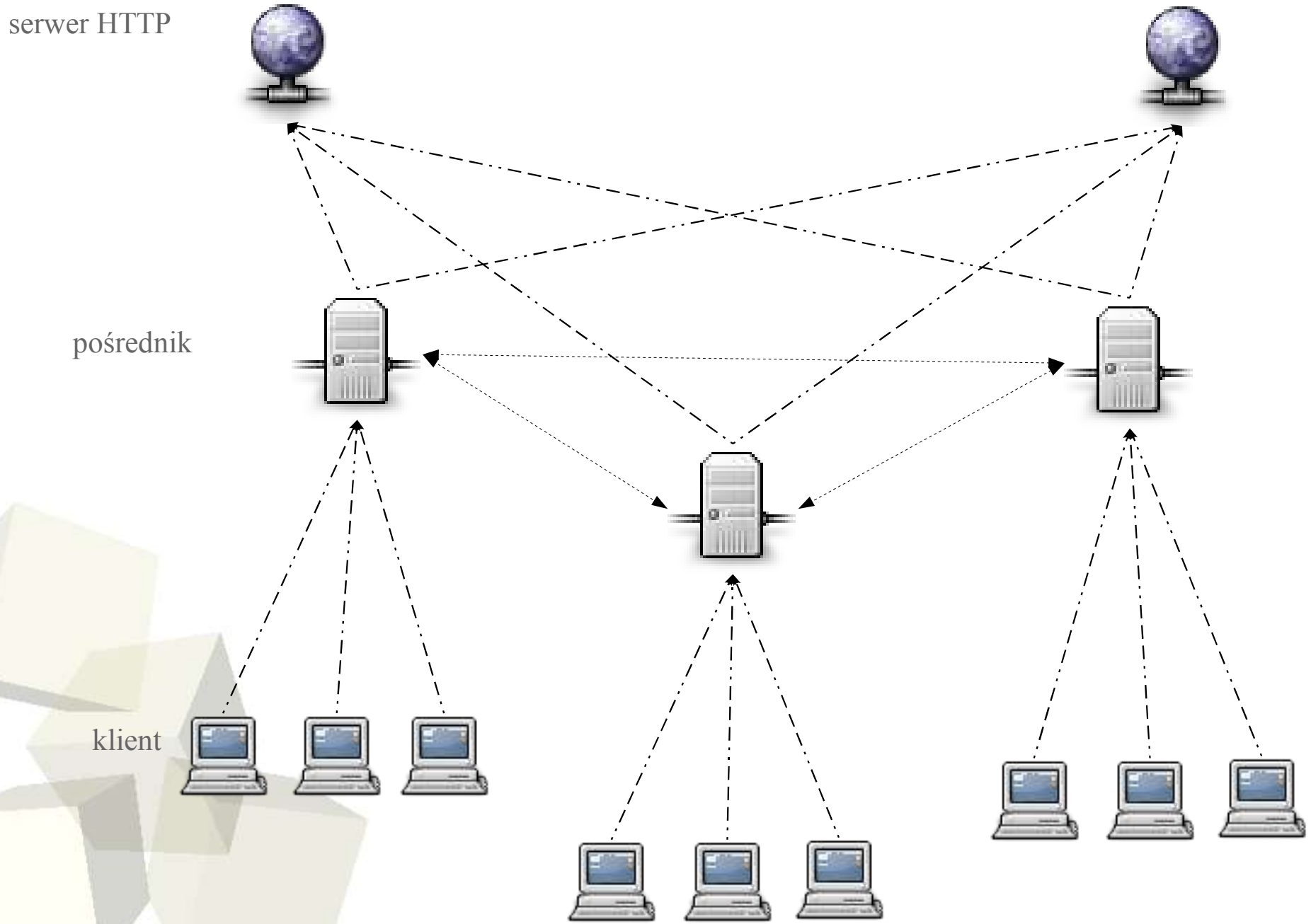
Dlaczego chcemy keszować?

- Możemy dostarczyć treść nawet jeśli serwer jest chwilowo niedostępny

Dlaczego nie chcemy keszować

- Nieaktualność danych
- Nietrafione zapytania
- Pojedynczy pośrednik – wąskie gardło
- Pojedynczy pośrednik – ryzyko awarii systemu
- Zaburzone statystyki odwołań do serwera

Ogólny model systemu pośredników



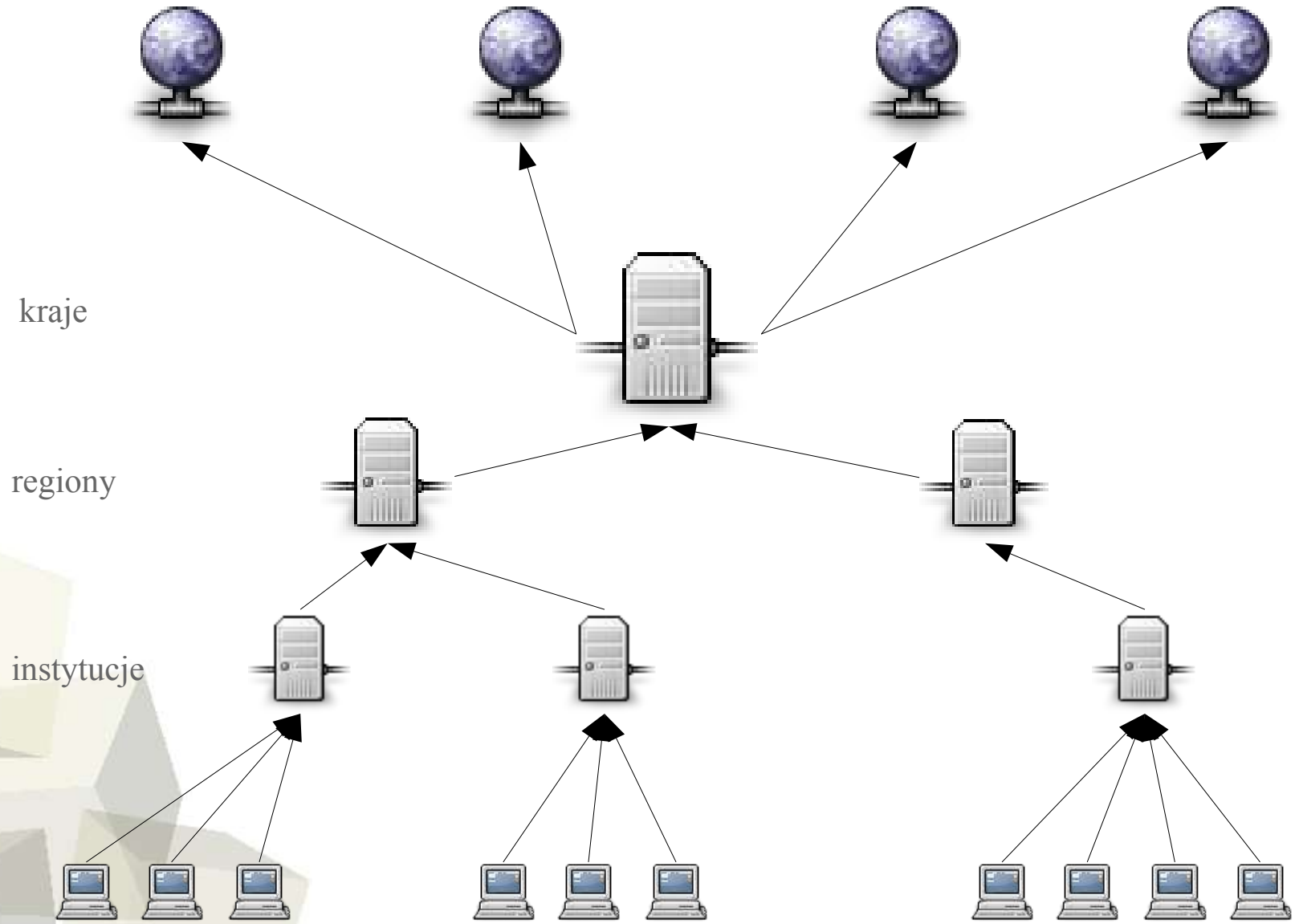
Pożądane cechy systemu pośredników

- Szybkość dostępu
- Niezawodność, dostępność
- Przezroczystość
- Skalowalność
- Wydajność

Pożądane cechy systemu pośredników

- Stabilność
- Adaptywność
- Rozkład ruchu
- Prostota

Model hierarchiczny

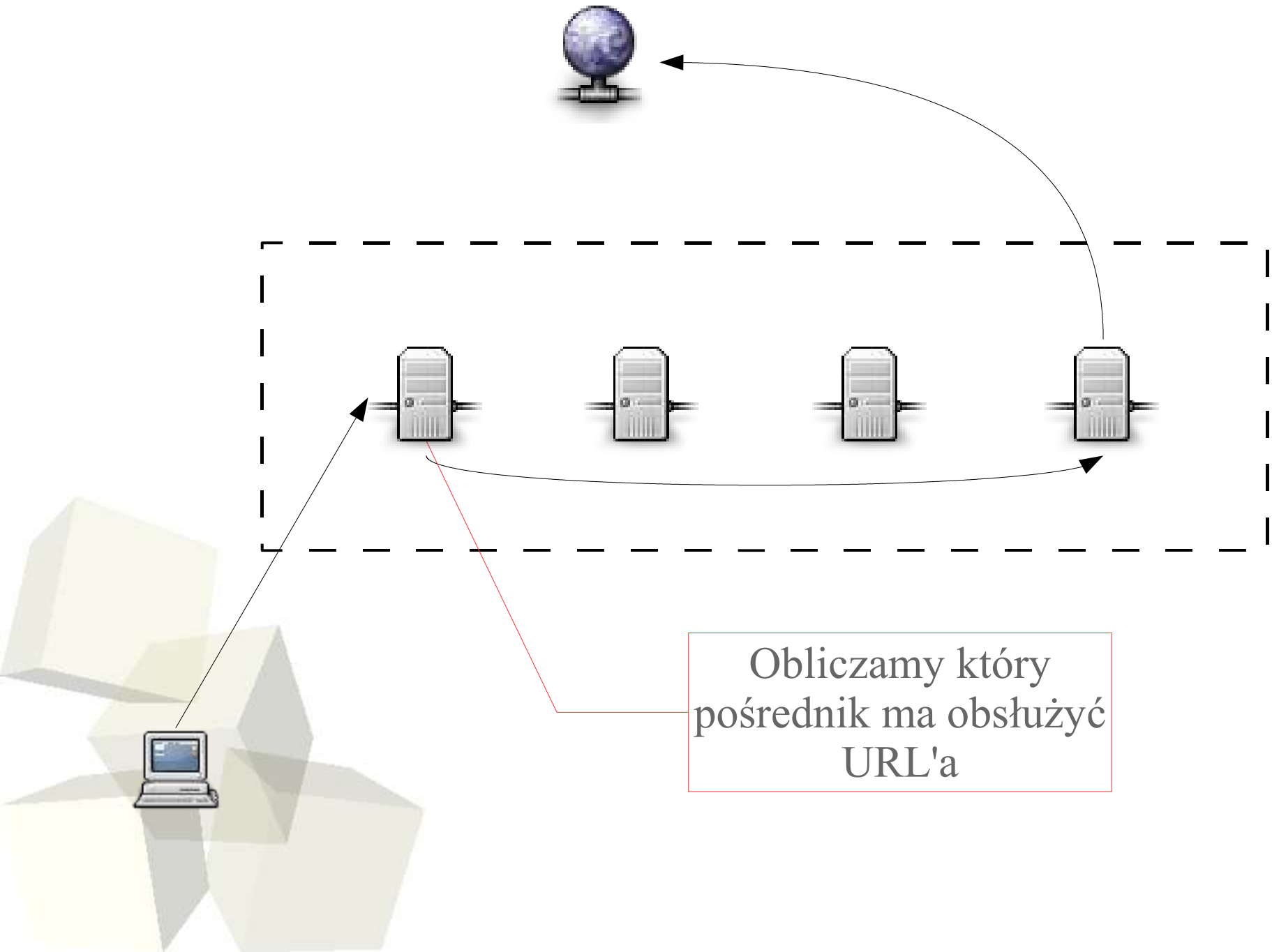


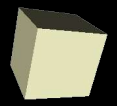
Wady modelu hierachicznego

- Konieczność instalowania pośredników w kluczowych miejscach sieci
- Dodatkowe opóźnienia
- Pośredniki wysoko w hierarchii mogą stanowić wąskie gardła
- Wiele kopii tego samego dokumentu

Model rozproszony

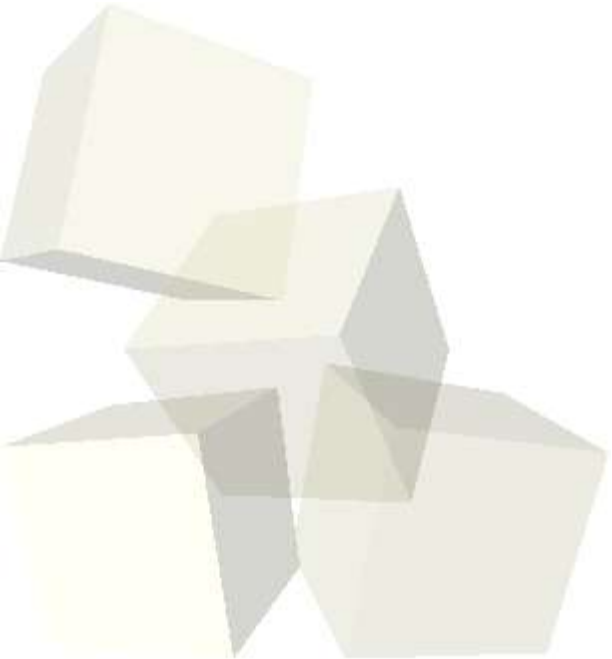
- Brak hierarchii wśród pośredników
- Każdy pośrednik może komunikować się z dowolnym innym pośrednikiem w systemie.
- Przykładowe systemy realizujące ten model:
  - CARP
  - IPC
  - CRISP


Obliczamy który pośrednik ma obsłużyć URL'a



Polityka wymiany keszowanej informacji

- Tradycyjne mechanizmy:
  - Least recently used (LRU)
  - Least frequently used (LFU)



aktualizacja informacji

- W przypadku keszowania pojawia się problem, czy pośrednik ma najświeższą wersję pliku
- Kesze muszą aktualizować przechowywane informacje aby serwować użytkownikom, tak świeże pliki, jak to tylko możliwe
- Protokół HTTP zawiera mechanizmy, które ułatwiają keszom dbanie o aktualność przechowywanej informacji

mechanizmy HTTP dotyczące keszowania

- Warunkowy GET
  - If-Modified-Since
  - If-Unmodified-Since
  - If-Match
  - If-None-Match
- Pragma: no-cache
- Last-modified
- Date

dlaczego P2P jest fajne

- Małe ograniczenia
- Systemy P2P są zdecentralizowane i rozproszone. Dzięki temu mogą być bardziej odporne na awarie i ataki.
  Mogą być stosowane do długich obliczeń i archiwizacji danych
- Systemy P2P pozwalają wykorzystać, nieużywane przez wielkość czasu:
  - dyski
  - czas procesora
  - łącza sieciowe

  użytkowników sieci.

Wyszukiwanie w sieci P2P

- Rozwiązanie z głównym serwerem (*Napster)*
  - Zapytania kieruje się do głównego serwera utrzymującego listę plików udostępnianych przez wszystkie węzły sieci
- Wady
  - Konieczność utrzymywania serwera
  - Mało skalowalne
  - Podatne na awarie (np. ataki ze strony RIAA)

Wyszukiwanie w sieci P2P

- Wyszukiwanie przez rozgłaszanie (*Gnutella*)
  - Każdy węzeł utrzymuje listę innych (dowolnych) węzłów sieci.
  - Zapytania kierowane są do wszystkich znanych węzłowi pytającemu węzłów.
  - Węzeł po otrzymaniu zapytania zwraca odpowiedź dotycząca jego zasobów i przekazuje zapytanie dalej do znanych sobie węzłów.
- Wady
  - Duża złożoność komunikacyjna $O(n^2)$
  - Mało skalowalne.

Wyszukiwanie w sieci P2P

- Cechy nowoczesnych systemów P2P:
  - Nie ma wyróżnionych węzłów
  - Każdy węzeł bierze udział tylko w małej części wyszukiwań
  - Każdy węzeł zna tylko mały zbiór innych węzłów
  - Sieć samoorganizuje się małym kosztem w miarę dołączania i odłączania węzłów.

DHT - model wyszukiwania

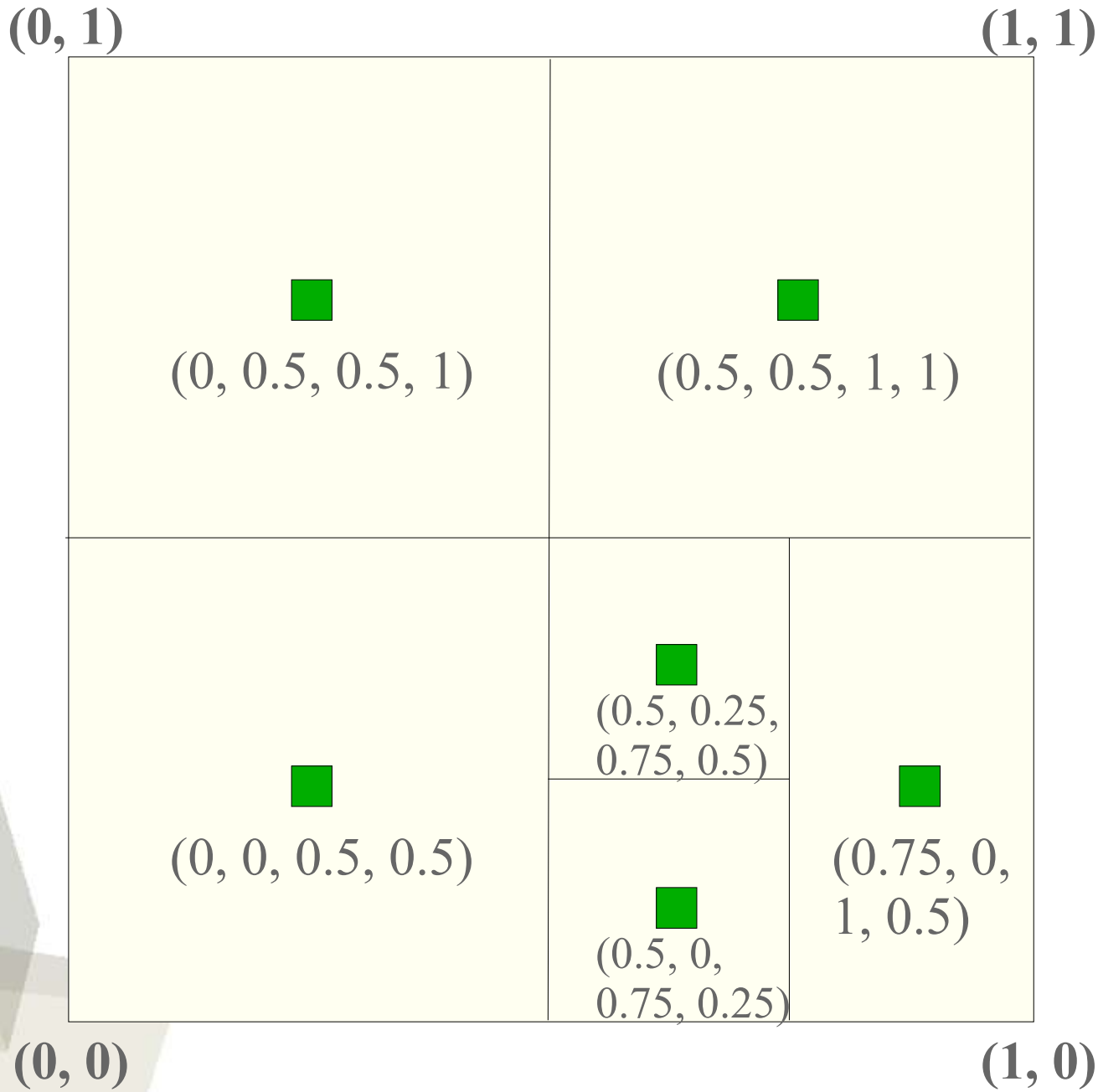
- Model Distributed Hash Table (DHT)
  - uniwersalny interfejs modelujący wyszukiwanie danych w sieci P2P
  - jedna operacja *lookup(key)*
  - operacja zwraca węzeł sieci
  - dane związane z kluczem *key* są umiejscawiane w węźle *lookup(key)*.

DHT - problemy

- Równomierne mapowanie kluczy do węzłów
- Przekierowywanie wyszukiwania klucza do odpowiedniego węzła
- Trzymanie tablic routingu

- CAN = Content-Addressable Network
- *d*-wymiarowy kartezjański układ współrzędnych
- Przestrzeń podzielona na hyper-prostokąty (strefy)

2-wymiarowy CAN z 6 węzłami

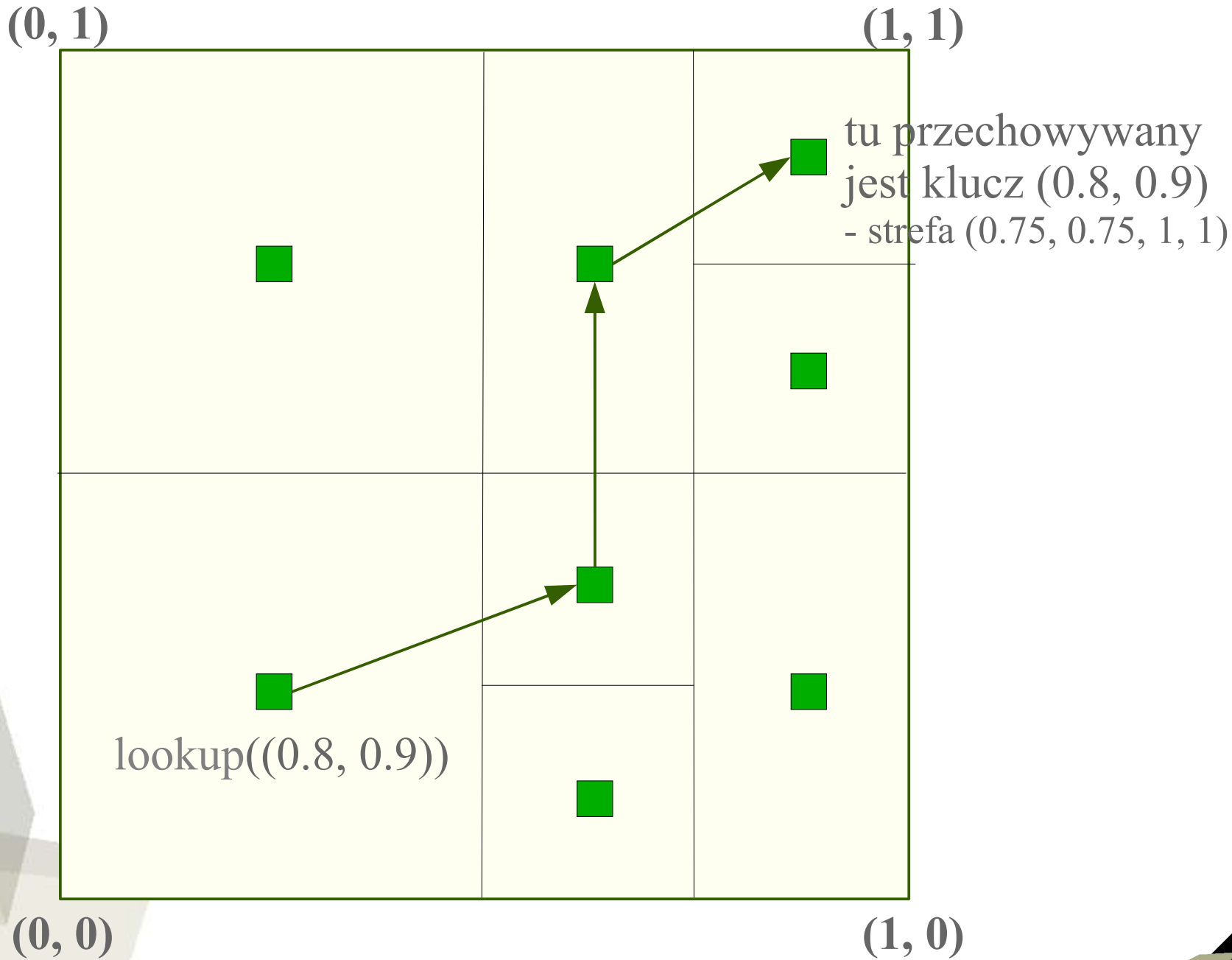


- Każdy węzeł odpowiada za jedną strefę i jest identyfikowany przez jej granice
- Każdy klucz mapowany jest do punktu w tej przestrzeni
- Każdy z węzłów trzyma listę sąsiadów – 2 węzły sąsiadują ze sobą jeśli dzielą tę samą *d-1*-wymiarową hyperpłaszczyznę

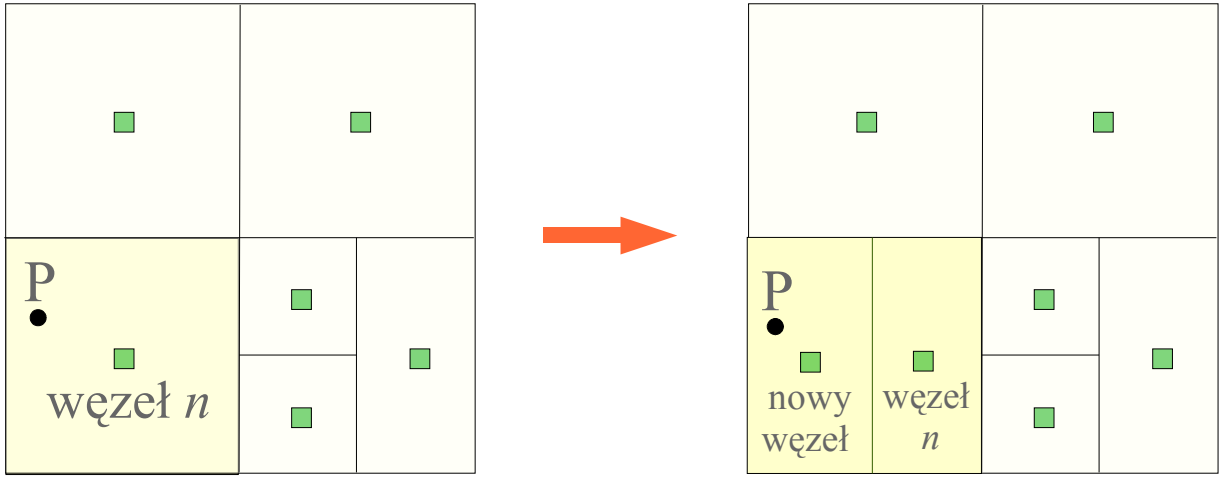
CAN - wyszukiwanie

- Wyszukiwanie klucza polega na przekazywaniu zapytania wzdłuż ścieżki przybliżającej linię prostą
- W trakcie zapytania, węzeł przekazuje je do sąsiada, który jest najbliżej węzła przechowującego klucz

CAN – wyszukiwanie klucza (0.8, 0.9)



CAN – dołącznie do sieci

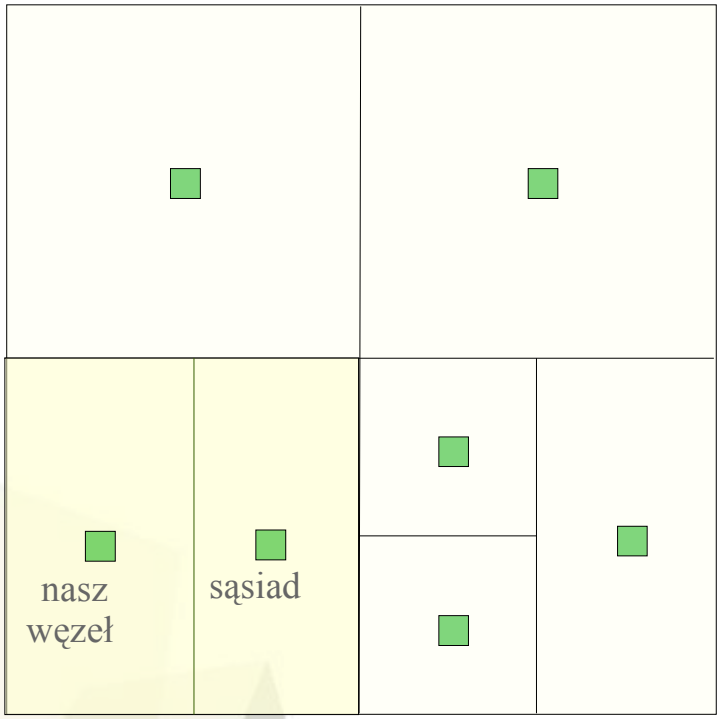


- Nowy węzeł wybiera losowy punkt P.
- Prosi węzeł będący już w sieci o znalezienie węzła *n*, którego strefa zawiera punkt P.
- *n* dzieli swoją strefę na pół i oddaje jedną część nowemu węzłowi.
- Po dołączeniu węzły uaktualniają listy sąsiadów.

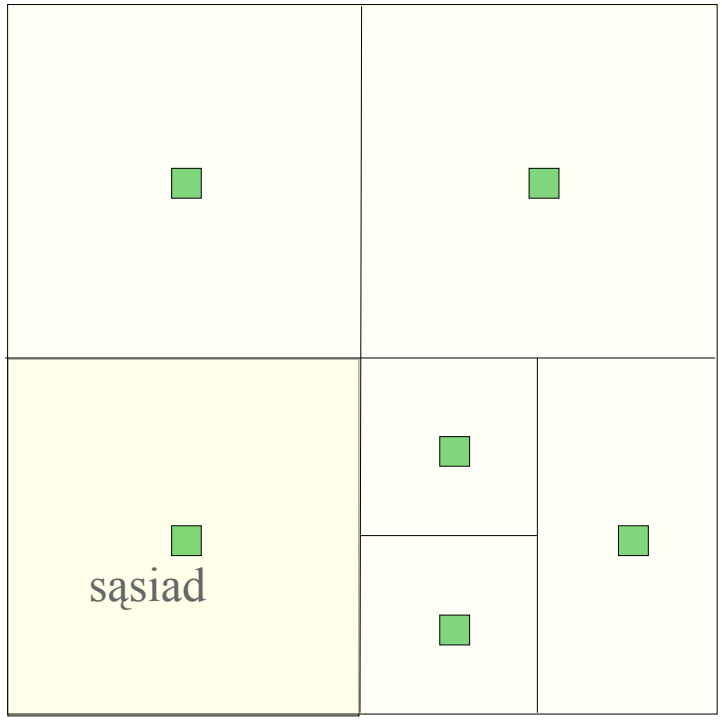
CAN – odłączanie od sieci

- Kiedy węzeł się odłącza, przekazuje swoją strefę jednemu z sąsiadów.
- Jeśli złączenie dwóch stref tworzy nową strefę, to są on łączone.
- W przeciwnym przypadku sąsiedni węzeł tymczasowo obsługuje obie strefy.
- Jeśli węzeł ulegnie awarii, jego strefę przejmuje sąsiad obsługujący najmniejszą strefę.

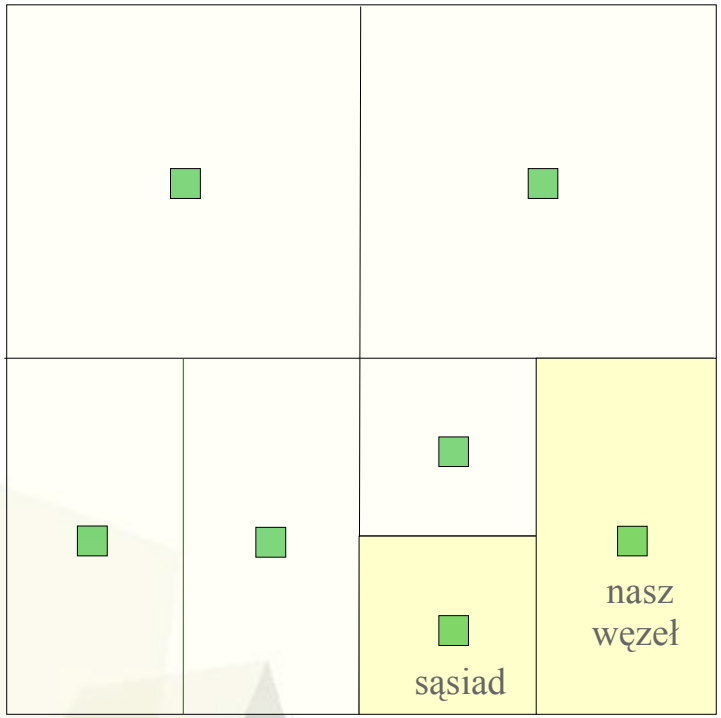
CAN – odłączanie od sieci



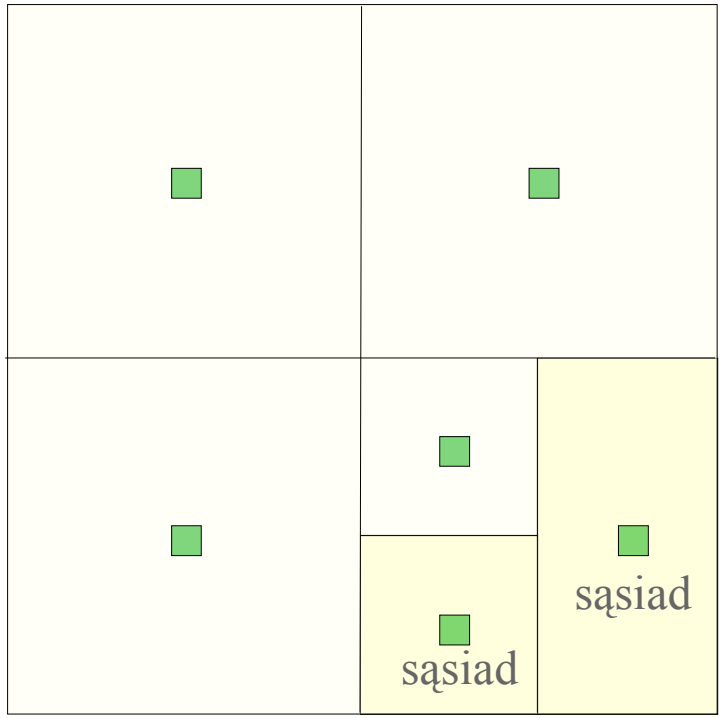
możemy
złączyć
strefy



CAN – odłączanie od sieci

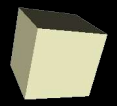


nie możemy
złączyć
stref



Sąsiad obsługuje dwie strefy

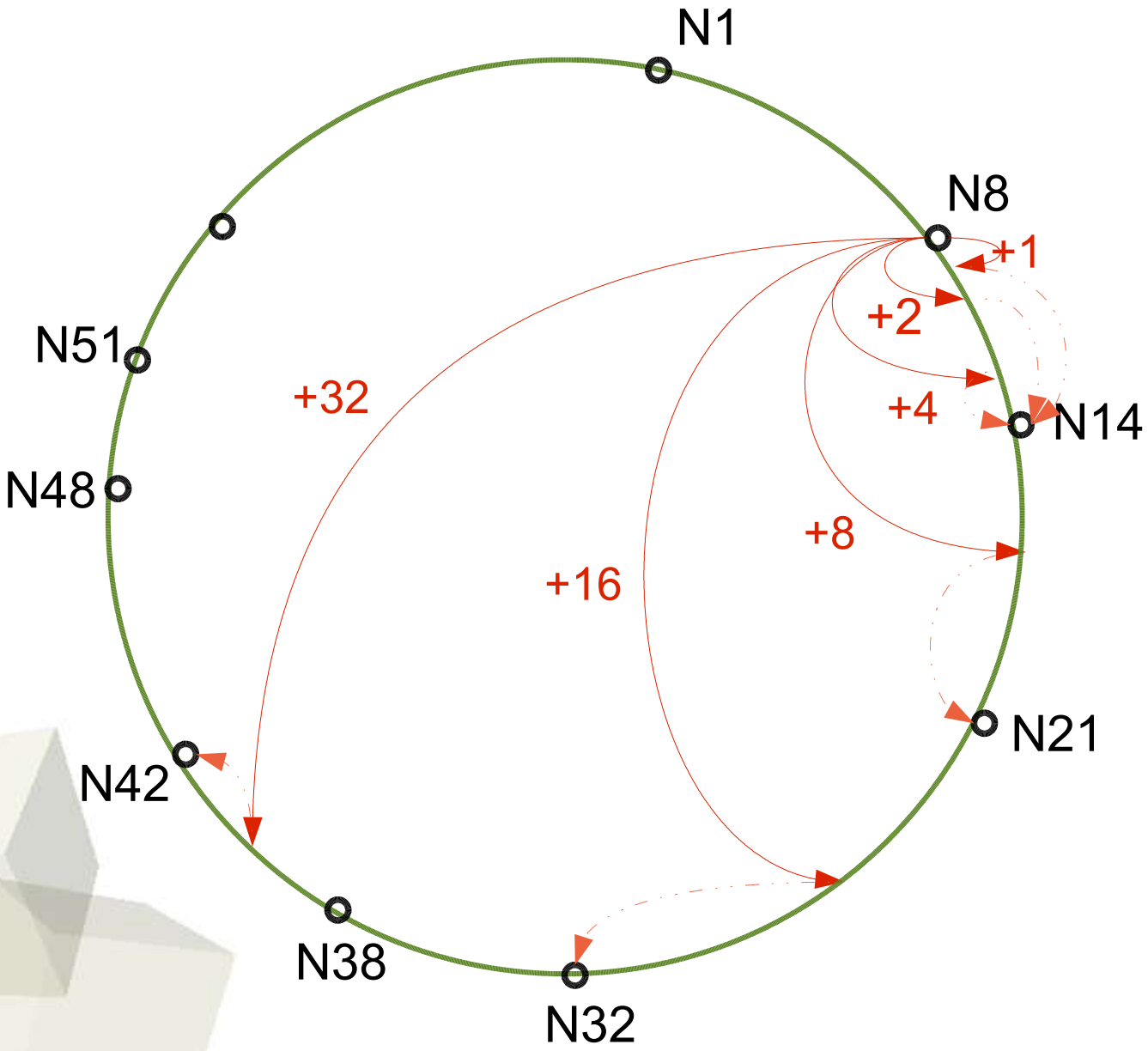
- Chord przyporządkowuje kluczom i węzłom identyfikatory z tej samej 1-wymiarowej przestrzeni
- Węzeł odpowiedzialny za klucz *k* nazywany jest *następnikiem*. Jest to najbliższy węzeł następujący po *k*.
- Przestrzeń identyfikatorów się zapętla.



Chord

- *n* – liczba węzłów
- *M* – wielkość przestrzeni identyfikatorów
- Każdy z węzłów przechowuje tablicę *log n* węzłów
- W tablicy tej znajdują się węzły odległe od danego węzła o:
  - *M*/2
  - *M*/4
  - ...
  - *1*

Chord – tablica węzłów

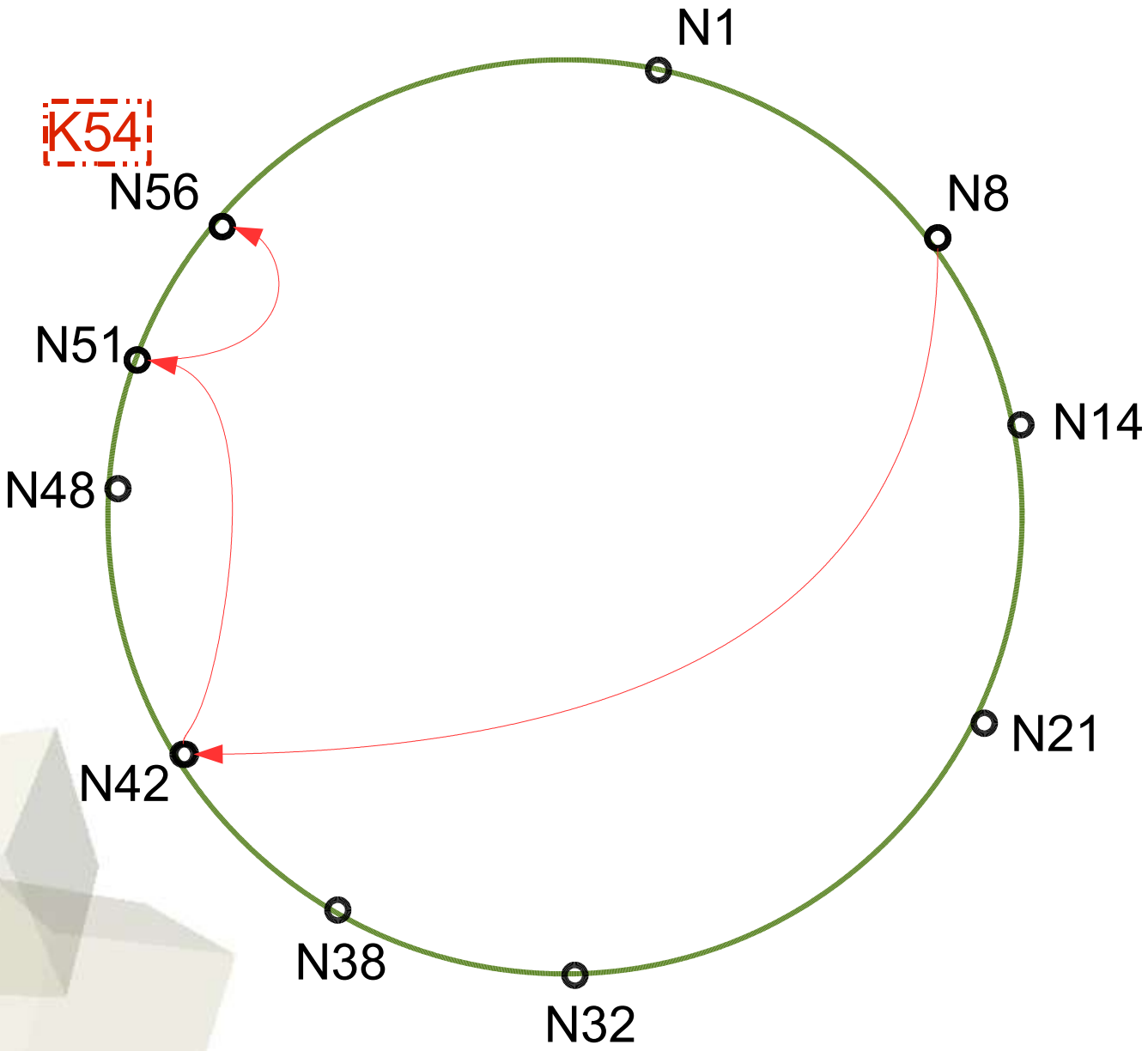


Chord - wyszukiwanie

- Węzeł wyszukuje w *tablicy węzłów*, węzeł z najwyższym identyfikatorem, mniejszym od *k*.
- Następnie przekierowuje zapytanie o klucz *k* do znalezionego węzła .
- Struktura potęg dwójki w *tablicy węzłów* zapewnia, że każdy etap wyszukiwania zmniejsza dystans co najmniej o połowę.
- W sumie wysyłamy *O(log n)* komunikatów.

Chord – wyszukiwanie klucza K54

przestrzeń identyfikatorów $M = 2^6 = 64$



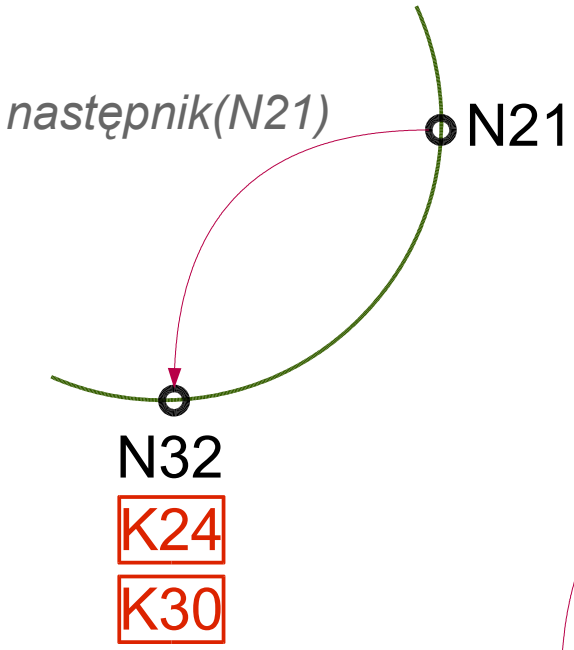
Chord – poprawność wyszukiwania

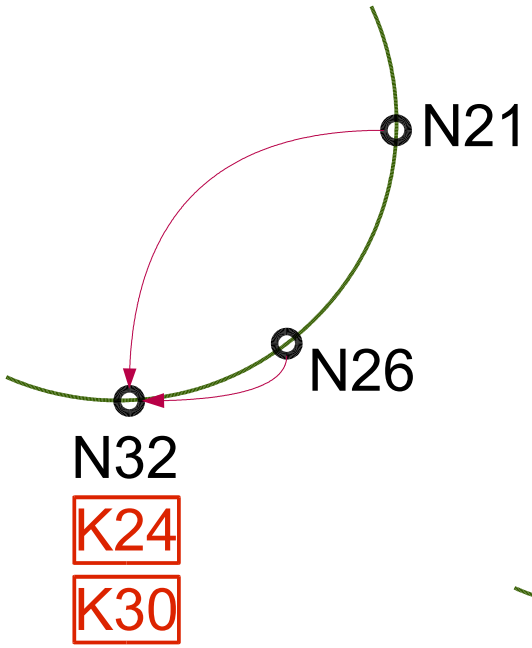
- Chord zapewnia poprawne wyszukiwanie nawet podczas awarii wielu węzłów w systemie.
- Wykorzystuje do tego *listę następników* – każdy węzeł trzyma listę *r* węzłów po nim następujących w przestrzeni nazw.
- Jako, że identyfikatory maszyn są wybrane losowo to szansa, że naraz ulegną awarii wszystkie węzły z *listy następników,* jest bardzo mała.

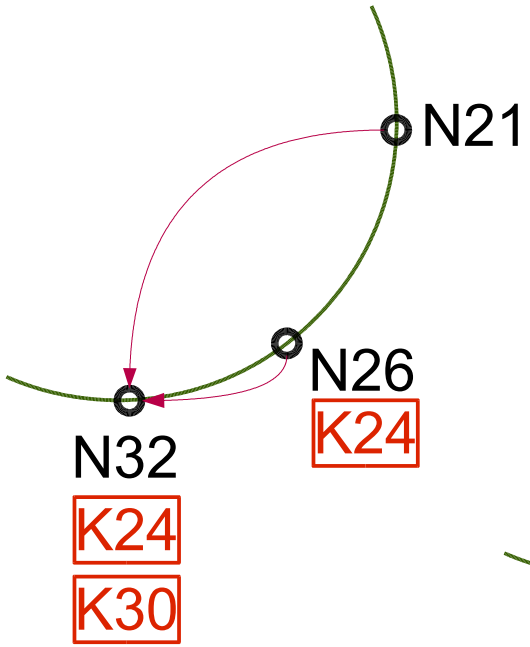
Chord – dodawanie węzła

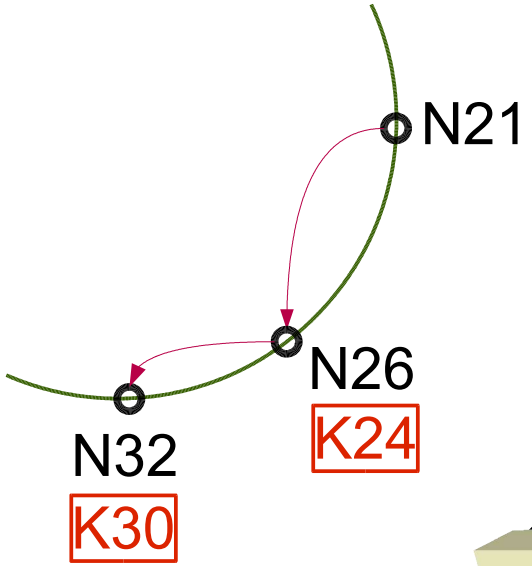
- Węzeł *n* prosi dowolny istniejący węzeł o odnalezienie jego ID.
- Nowy węzeł przejmuje od swojego *następnika* klucze, za które jest od teraz odpowiedzialny.
- Aby móc brać udział w wyszukiwaniu musimy zaktualizować *listę następników* dla nowego węzła i jego *poprzednika*.
- W tle uaktualniane są tablice węzłów dla węzłów systemu.

Chord – dodawanie węzła N26









Chord – usuwanie węzła

- Węzeł odchodzący przekazuje wszystkie klucze swojemu *następnikowi.*
- Węzeł przekazuje informację o tym, że odchodzi *następnikowi* i *poprzednikowi*. Dzięki temu *poprzednik* może zaktualizować *listę następników,* a *następnik* – swojego poprzednika.

Chord – koszt algorytmu

- *n – liczba węzłów w systemie*
- *log n* - o tylu węzłach wie każdy z węzłów
- *log n* - tyle komunikatów jest potrzebnych przy wyszukiwaniu
- $(log\ n)^2$ – tyle komunikatów jest potrzebnych przy przyłączeniu węzła

Nasza praca magisterska

- Połączenie keszowania z systemami P2P
- Rozproszony pośrednik realizowany jako sieć P2P
- Lokalny interfejs pośrednika HTTP w każdym węźle sieci
- Rozpraszanie informacji o miejscach przechowywania keszowanych plików podobnie jak w systemie Chord
- Rozpraszanie plików multimedialnych
- ...