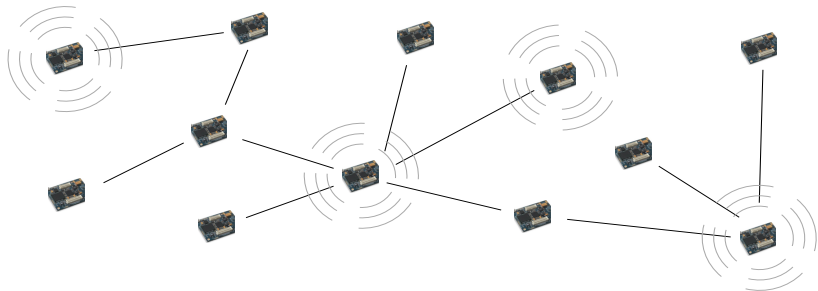


# Algorithmen für Ad-hoc- und Sensornetze

## VL 03 – Location Services

Fabian Fuchs | 29. Okt. 2015 (Version 1)

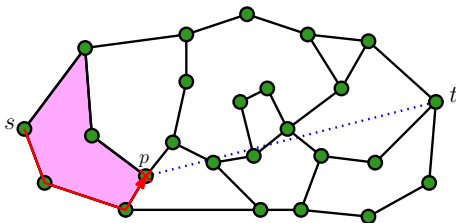
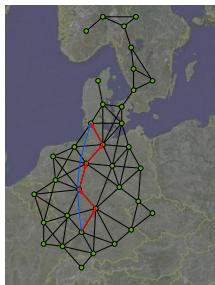
INSTITUT FÜR THEORETISCHE INFORMATIK - LEHRSTUHL FÜR ALGORITHMIK (PROF. WAGNER)



- Rückblick: Geographisches Routing
  - Denkanstoß: Anwendbarkeit in 3D
- Location Services
  - Definition und Anforderungen
  - Triviale Ansätze (Vor- und Nachteile)
- Grid Location System (GLS)
- MLS

Kennt jeder Knoten seine Position und die seiner Nachbarn, kann man Pakete zu *Zielkoordinaten* routen.

- Greedy: schnell und einfach, keine garantierte Auslieferung
- Facettenrouting (OAFR): Garantierte Auslieferung und Laufzeit
- Techniken können kombiniert werden (GOAFR)



Kennt jeder Knoten seine Position und die seiner Nachbarn, kann man Pakete zu *Zielkoordinaten* routen.

- Greedy: schnell und einfach, keine garantierte Auslieferung
  - Facettenrouting (OAFR): Garantierte Auslieferung und Laufzeit
  - Techniken können kombiniert werden (GOAFR)
- 
- Annahme: Position des Zielknotens bekannt
    - kein „Auffinden“ von bestimmten Knoten nötig
    - Knoten verändern ihre Position nicht

Was, wenn man gezielt bestimmte Knoten sucht? Was, wenn Knoten ihre Position ändern können?

- Wie finde ich zu einer Ziel-Knoten-ID eine aktuelle Position?

- Rückblick: Geographisches Routing
  - Denkanstoß: Anwendbarkeit in 3D
- Location Services
  - Definition und Anforderungen
  - Triviale Ansätze (Vor- und Nachteile)
- Grid Location System (GLS)
- MLS

## Definition

Ein *Location Service* ist eine Infrastruktur, die zu gegebener Knoten-ID eine aktuelle Geokoordinate liefert.

Ablauf:

- Sender schickt Anfrage mit Knoten-ID ab
- Location Service antwortet mit Geokoordinate
- Sender schickt Paket an Geokoordinate
  - wenn Sender eigene Position mitschickt, kann die Antwort direkt per Georouting kommen

## Alternative Sicht

Ein *Location Service* ist ein proaktives Routingprotokoll, das Pakete mit angegebener Zielknoten-ID an die aktuelle Geokoordinate des Zielknoten leitet.

Ablauf:

- Sender schickt Paket mit Ziel-ID ab
- Location Service leitet Paket an entsprechende Zielkoordinate
  - wenn Sender eigene Adresse mitschickt, kann die Antwort direkt per Georouting kommen

`publish`: Veröffentlichung von Knotenpositionen

- Knoten, die sich bewegen, müssen das mitteilen
- Positionsinformation enthält in der Regel Timeout
- *Wem teilt ein Knoten seine Position mit?*

`lookup`: Auflösen von IDs in Knotenpositionen

- Anfragen nach Knotenpositionen müssen umgesetzt werden. *An wen richtet man sie?*
- Pakete, die eine Ziel-ID enthalten, müssen zu Zielkoordinaten geleitet werden. *An wen schickt man diese Pakete?*

Besondere Form eines Rendezvous-Problems: Jeder muss seine Positionsinformation so hinterlassen, dass andere sie finden können.



## `publish` per Broadcast

Bei einem `publish` wird die neue Position eines Knotens an alle Knoten geschickt. Jeder Knoten kennt dann immer alle aktuellen Positionen.

- + triviale `lookup`-Operation
- jeder Knoten muss große Tabelle speichern
- Anzahl der Nachrichten pro `publish` immer in  $\Omega(n)!$

## Zentraler Server

Ein zentraler Knoten mit bekannter Position nimmt `publish`- und `lookup`-Nachrichten entgegen.

- + im Schnitt geringer Speicherplatzverbrauch
- ein Knoten mit hoher Last und großer Verantwortung
- ? Nachrichtenkomplexität in  $O(D)$  pro Operation, aber:
  - bei `publish` nicht von Positionsänderung abhängig
  - bei `lookup` nicht von Entfernung zum Ziel abhängig

## Faire Lastverteilung

Knoten teilen sich die Arbeit, kein Knoten erschöpft sich an dieser Aufgabe.

## Fehlertoleranz, kein „single point of failure“

Ausfall einzelner Knoten verursacht keinen Totalausfall.

## Verhältnismäßigkeit der Kommunikation

Anfragen zu nahen Knoten verursachen nur geringe Kommunikation.  
Idealerweise: Geringe Bewegungen erzeugen nur geringe Kommunikation.

## Skalierbarkeit

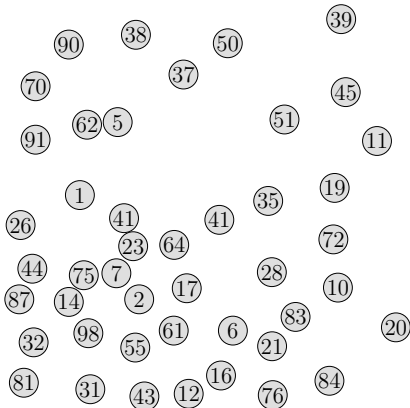
Kosten wachsen möglichst wenig in der Knotenzahl.

- zuverlässige lokale Kommunikation
- verhältnismäßig hohe Knotendichte
  - häufige Annahme:  $D \in O(\sqrt{n})$
- oft Gebiet bekannt
- eingeschränkte Mobilität
  - keine großen Bewegungen zwischen elementaren Operationen
- eindeutige, geordnete Knoten-IDs aus bekanntem Intervall
  - im Zweifel: IDs kollisionsfrei hashen

- Rückblick: Geographisches Routing
  - Denkanstoß: Anwendbarkeit in 3D
- Location Services
  - Definition und Anforderungen
  - Triviale Ansätze (Vor- und Nachteile)
- **Grid Location System (GLS)**
- MLS

# Grid Location System (GLS) — Idee

- Aufteilung des Gebietes durch  $M$  Gitter (Quadtree)
  - $G_0$  bis  $G_{M-1}$
  - $G_0$ : jeder sieht jeden
  - $G_i$  hat  $2^i$ -fache Kantenlänge von  $G_0$
  - $G_M$  hätte alle Knoten in einer Zelle



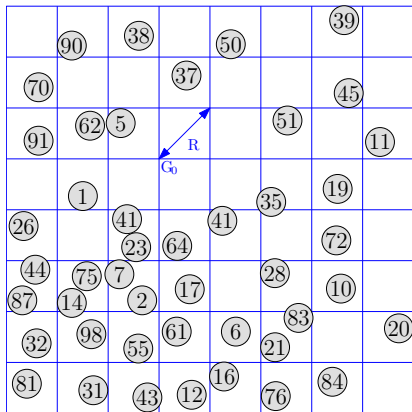
## Definition:

Eine Zelle auf Ebene  $G_i$  ist eine *Nachbarzelle von B*, wenn beide in  $G_{i+1}$  in derselben Zelle sind.

Jeder Knoten wählt einen Server pro Nachbarzelle in jedem  $G_i$ !

Das sind  $3M$  Server für jeden Knoten.

- Aufteilung des Gebietes durch  $M$  Gitter (Quadtree)
  - $G_0$  bis  $G_{M-1}$
  - $G_0$ : jeder sieht jeden
  - $G_i$  hat  $2^i$ -fache Kantenlänge von  $G_0$
  - $G_M$  hätte alle Knoten in einer Zelle



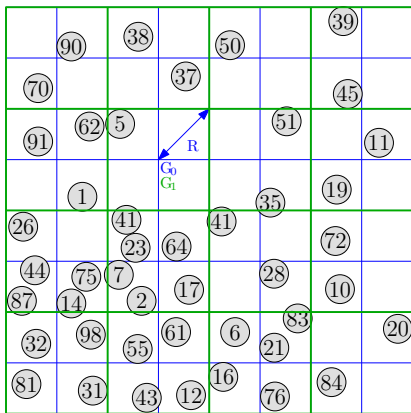
## Definition:

Eine Zelle auf Ebene  $G_i$  ist eine *Nachbarzelle von B*, wenn beide in  $G_{i+1}$  in derselben Zelle sind.

Jeder Knoten wählt einen Server pro Nachbarzelle in jedem  $G_i$ !

Das sind  $3M$  Server für jeden Knoten.

- Aufteilung des Gebietes durch  $M$  Gitter (Quadtree)
  - $G_0$  bis  $G_{M-1}$
  - $G_0$ : jeder sieht jeden
  - $G_i$  hat  $2^i$ -fache Kantenlänge von  $G_0$
  - $G_M$  hätte alle Knoten in einer Zelle



## Definition:

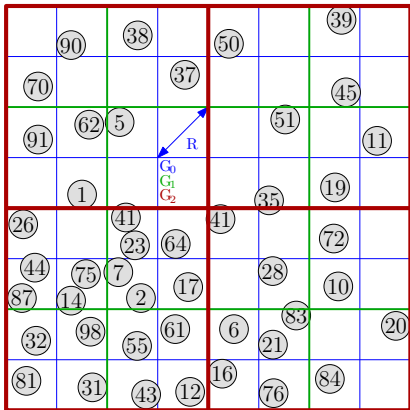
Eine Zelle auf Ebene  $G_i$  ist eine *Nachbarzelle von B*, wenn beide in  $G_{i+1}$  in derselben Zelle sind.

Jeder Knoten wählt einen Server pro Nachbarzelle in jedem  $G_i$ !

Das sind  $3M$  Server für jeden Knoten.



- Aufteilung des Gebietes durch  $M$  Gitter (Quadtree)
  - $G_0$  bis  $G_{M-1}$
  - $G_0$ : jeder sieht jeden
  - $G_i$  hat  $2^i$ -fache Kantenlänge von  $G_0$
  - $G_M$  hätte alle Knoten in einer Zelle



## Definition:

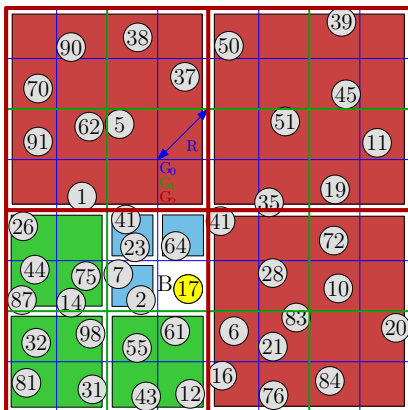
Eine Zelle auf Ebene  $G_i$  ist eine *Nachbarzelle von B*, wenn beide in  $G_{i+1}$  in derselben Zelle sind.

Jeder Knoten wählt einen Server pro Nachbarzelle in jedem  $G_i$ !

Das sind  $3M$  Server für jeden Knoten.

# Grid Location System (GLS) — Idee

- Aufteilung des Gebietes durch  $M$  Gitter (Quadtree)
  - $G_0$  bis  $G_{M-1}$
  - $G_0$ : jeder sieht jeden
  - $G_i$  hat  $2^i$ -fache Kantenlänge von  $G_0$
  - $G_M$  hätte alle Knoten in einer Zelle



## Definition:

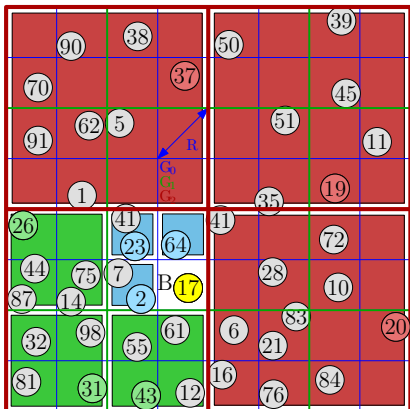
Eine Zelle auf Ebene  $G_i$  ist eine *Nachbarzelle* von  $B$ , wenn beide in  $G_{i+1}$  in derselben Zelle sind.

Jeder Knoten wählt einen Server pro Nachbarzelle in jedem  $G_i$ !

Das sind  $3M$  Server für jeden Knoten.

# Grid Location System (GLS) — Idee

- Aufteilung des Gebietes durch  $M$  Gitter (Quadtree)
  - $G_0$  bis  $G_{M-1}$
  - $G_0$ : jeder sieht jeden
  - $G_i$  hat  $2^i$ -fache Kantenlänge von  $G_0$
  - $G_M$  hätte alle Knoten in einer Zelle



## Definition:

Eine Zelle auf Ebene  $G_i$  ist eine *Nachbarzelle* von  $B$ , wenn beide in  $G_{i+1}$  in derselben Zelle sind.

Jeder Knoten wählt einen Server pro Nachbarzelle in jedem  $G_i$ !

Das sind  $3M$  Server für jeden Knoten.

## Grundidee:

- Knoten  $A$ , der Position eines Knotens  $B$  sucht, muss einen der Server von  $B$  finden
    - ⇒  $A$ s Suche nach Server von  $B$  und  $B$ s Server-Auswahl müssen ähnlich ablaufen
  - Gemeinsames Wissen von  $A$  und  $B$ :
    - Gitter-Einteilung
    - ID von  $B$
- ⇒ Strategie:  $B$  wählt Knoten als Location Server, deren ID ähnlich zur ID von  $B$  ist

## Grundidee:

- Knoten  $A$ , der Position eines Knotens  $B$  sucht, muss einen der Server von  $B$  finden
    - ⇒  $A$ s Suche nach Server von  $B$  und  $B$ s Server-Auswahl müssen ähnlich ablaufen
  - Gemeinsames Wissen von  $A$  und  $B$ :
    - Gitter-Einteilung
    - ID von  $B$
- ⇒ Strategie:  $B$  wählt Knoten als Location Server, deren ID ähnlich zur ID von  $B$  ist

## Grundidee:

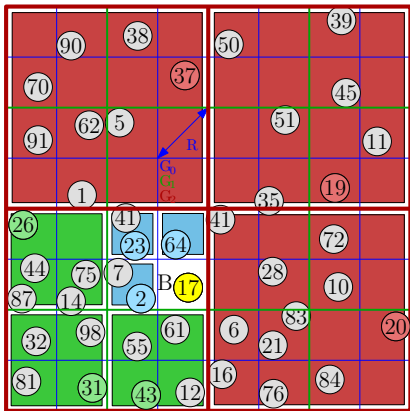
- Knoten  $A$ , der Position eines Knotens  $B$  sucht, muss einen der Server von  $B$  finden
    - ⇒ As Suche nach Server von  $B$  und  $B$ s Server-Auswahl müssen ähnlich ablaufen
  - Gemeinsames Wissen von  $A$  und  $B$ :
    - Gitter-Einteilung
    - ID von  $B$
- ⇒ Strategie:  $B$  wählt Knoten als Location Server, deren ID ähnlich zur ID von  $B$  ist

## Serverauswahl

In einer Nachbarzelle des Knoten  $B$  ist immer der Knoten  $X$  Server für  $B$ , der

$ID_X - ID_B \pmod{ID_{\max}}$   
minimiert.

- etwa gleichmäßige Verteilung der Arbeit
- jeder Knoten ist Server für  $\Theta(\log n)$  Knoten



Garantien nur bei gleichverteilten Knoten und zufälligen IDs!

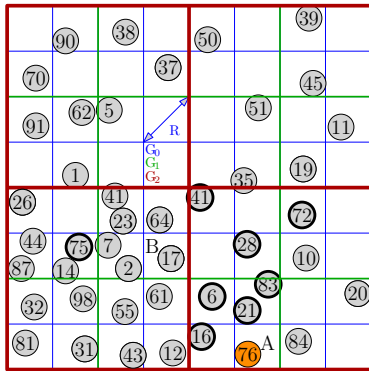
Welche Knoten ist A Server?

Frage für jeden Knoten  $B$  in Nachbarzelle von  $A$  (im Gitter  $G_k$ ):

- Gibt es in  $A$ s Zelle einen Knoten  $X$  mit  $ID_A > ID_X \geq ID_B$ ?<sup>1</sup>
- Falls nicht, ist  $A$  Server von  $B$

Beispiel:

- In Gitter  $G_0$ : Ja, da  $A$  einziger Knoten in  $A$ s Zelle
- In Gitter  $G_1$ : Für Knoten  $B$  mit
- In Gitter  $G_2$ : Für Knoten  $B$  mit



<sup>1</sup> Lies: Von  $ID_B$  an aufsteigend im Restklassenring  $\mathbb{Z}/ID_{\max}$  kommt  $ID_X$  vor  $ID_A$ .

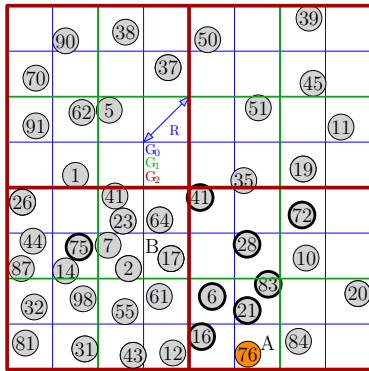


Welche Knoten ist A Server?

Frage für jeden Knoten  $B$  in Nachbarzelle von  $A$  (im Gitter  $G_k$ ):

- Gibt es in  $A$ s Zelle einen Knoten  $X$  mit  $ID_A > ID_X \geq ID_B$ ?<sup>1</sup>
- Falls nicht, ist  $A$  Server von  $B$

Beispiel:



- In Gitter  $G_0$ : Ja, da  $A$  einziger Knoten in  $A$ s Zelle
- In Gitter  $G_1$ : Für Knoten  $B$  mit  $76 > ID_B \geq 21$
- In Gitter  $G_2$ : Für Knoten  $B$  mit  $76 > ID_B \geq 72$

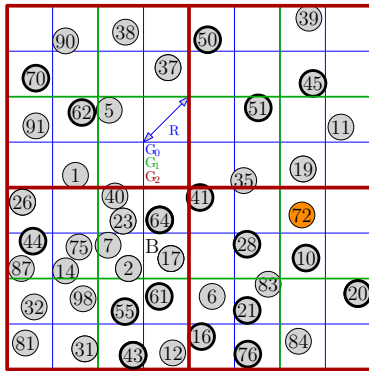
<sup>1</sup> Lies: Von  $ID_B$  an aufsteigend im Restklassenring  $\mathbb{Z}/ID_{\max}$  kommt  $ID_X$  vor  $ID_A$ .

Welche Knoten ist A Server?

Frage für jeden Knoten  $B$  in Nachbarzelle von  $A$  (im Gitter  $G_k$ ):

- Gibt es in  $A$ s Zelle einen Knoten  $X$  mit  $ID_A > ID_X \geq ID_B$ ?<sup>1</sup>
- Falls nicht, ist  $A$  Server von  $B$

Beispiel:



- In Gitter  $G_0$ : Ja, da  $A$  einziger Knoten in  $A$ s Zelle
- In Gitter  $G_1$ : Für Knoten  $B$  mit  $72 > ID_B \geq 10$
- In Gitter  $G_2$ : Für Knoten  $B$  mit  $72 > ID_B \geq 41$

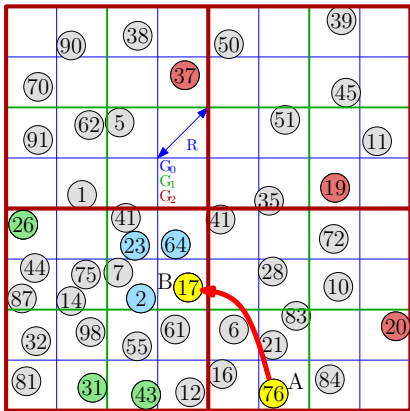
<sup>1</sup> Lies: Von  $ID_B$  an aufsteigend im Restklassenring  $\mathbb{Z}/ID_{\max}$  kommt  $ID_X$  vor  $ID_A$ .

# Lookup (Versuch 1)

Tun wir für einen Moment so, als kenne jeder Knoten die Positionen der Knoten, für die er Server ist!

Wie könnte ein `lookup` funktionieren?

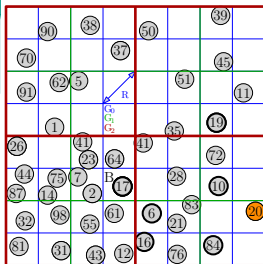
- Knoten A sucht Knoten B.



## Satz

Seien  $A, B$  beliebige Knoten.  $A$  ist Server für einen Knoten  $X$  mit  $ID_A > ID_X \geq ID_B \pmod{ID_{\max}}$ .

- sind  $A$  und  $B$  in benachbarten  $G_i$ -Zellen,
  - liegt kein  $X$  mit  $ID_A > ID_X \geq ID_B$  in  $A$ s Zelle  $\Rightarrow A$  ist Server für  $B$ .
  - liegt ein größtes  $X$  mit  $ID_A > ID_X \geq ID_B$  in  $A$ s Zelle  $\Rightarrow A$  ist Server für  $X$ .

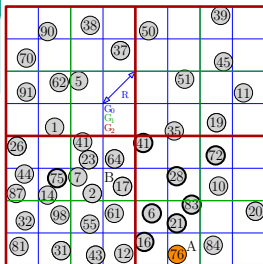


Weiterleitung an irgendeine „besseren“ Knoten ist zwar korrekt,  
kann aber beliebig lange Wege erzeugen!

## Satz

Seien  $A, B$  beliebige Knoten.  $A$  ist Server für einen Knoten  $X$  mit  $ID_A > ID_X \geq ID_B \pmod{ID_{\max}}$ .

- sind  $A$  und  $B$  in benachbarten  $G_i$ -Zellen,
  - liegt kein  $X$  mit  $ID_A > ID_X \geq ID_B$  in  $A$ s Zelle  $\Rightarrow A$  ist Server für  $B$ .
  - liegt ein größtes  $X$  mit  $ID_A > ID_X \geq ID_B$  in  $A$ s Zelle  $\Rightarrow A$  ist Server für  $X$ .



Weiterleitung an irgendeine „besseren“ Knoten ist zwar korrekt, kann aber beliebig lange Wege erzeugen!

## Definition

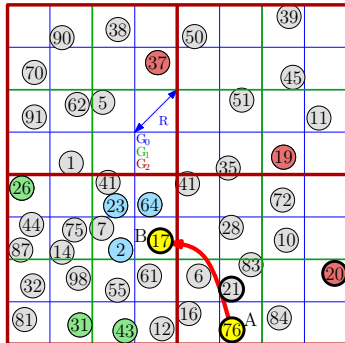
Seien  $A, B$  beliebige Knoten. Dann sei  $A_i^B$  der Knoten in der  $G_i$ -Zelle von  $A$ , der  $ID_{A_i^B} - ID_B$  minimiert.

Beispiel:

- $A_0^B$ : 76
- $A_1^B$ : 21
- $A_2^B$ : 20

## Beobachtung

Seien  $A, B$  beliebige Knoten in derselben Zelle in  $G_k$ . Dann ist  $A_k^B = B$ .



## Definition

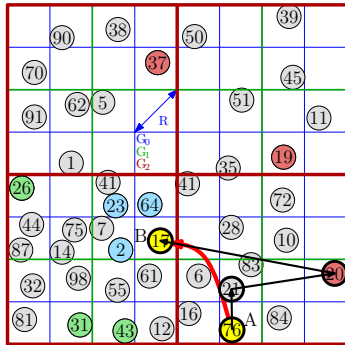
Seien  $A, B$  beliebige Knoten. Dann sei  $A_i^B$  der Knoten in der  $G_i$ -Zelle von  $A$ , der  $ID_{A_i^B} - ID_B$  minimiert.

## Lemma

Seien  $A, B$  beliebige Knoten.  $B$  kennt  $B_0^A$  und jedes  $B_i^A$  ist Server von  $B_{i+1}^A$ .

- $A$  kennt komplette  $G_0$ -Zelle.
  - $ID_{A_i^B}$  ist in  $G_i$ -Zelle von  $A$  am wenigsten größer als  $ID_B$
- ⇒  $A_i^B$  ist Server für alle Knoten in Nachbarzellen zwischen  $B$  und  $A_i^B$ , auch  $A_{i+1}^B$ !

Hier: 21 ist Server für 20



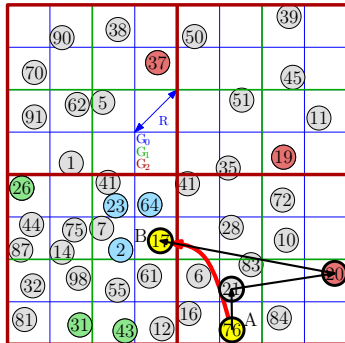
## Definition

Seien  $A, B$  beliebige Knoten. Dann sei  $A_i^B$  der Knoten in der  $G_i$ -Zelle von  $A$ , der  $ID_{A_i^B} - ID_B$  minimiert.

## GLS-lookup

Auf Suche nach Knoten  $B$  schickt  $A$  die Anfrage an  $A_0^B, A_1^B, \dots, B$ .

- Anfragen?
- Weglänge?
  - $A$  zu  $A_0^A$ : maximal  $R$ .
  - $A_i^B$  zu  $A_{i+1}^B$ : maximal  $2^{i+1} R$ .
  - insgesamt Summe der „Luftlinien“  $\sum_{i=0}^k 2^i R \in O(2^i)$
  - hängt vom Gitter ab!





# Einschub: Zielkoordinaten ohne Knoten

## Idee

Man kann per Georouting auch Zieladressen angeben, an denen gar kein Knoten liegt! Was passiert dann z. B. bei GOAFR?

- das Paket
  - umrundet die Facette, die die Zielcoordinate einschließt
  - lernt alle Knoten der Facette kennen
  - im Gabriel Graph: sieht dichtesten Knoten zur Zielcoordinate

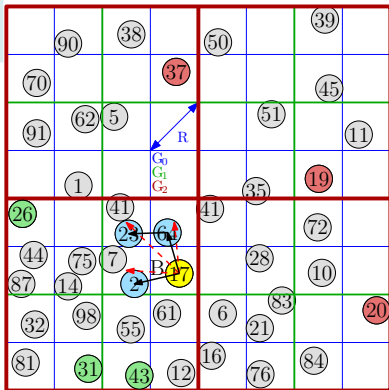
## Lemma

*Wird ein Paket zur Mitte einer Gitterzelle  $C$  geroutet, in der ein Knoten liegt, dann erreicht das Paket mindestens einen Nachbarn eines solchen Knotens.*

(gilt zumindest in  $1/\sqrt{2}$ -Quasi-UDGs, ohne Beweis)

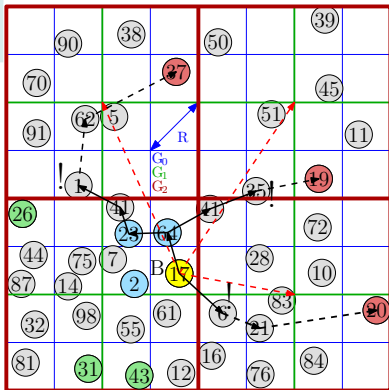
Woher weiß  $B$ , wer seine Server sind?  
Wie findet er sie?

- Ebene für Ebene!
- $G_0$ -Zellen-Server: Schicke Paket an Zentrum der Zellen
  - in nichtleerer Zelle kommt das Paket bei einem Knoten der Zelle an
  - der kennt alle Knoten der Zelle und benachrichtigt den zuständigen Knoten



Woher weiß  $B$ , wer seine Server sind?  
Wie findet er sie?

- Ebene für Ebene!
- wenn  $G_i$ -Zellen-Server bekannt sind
- route künstliches Lookup zu irgendeinem Knoten  $A$  in entsprechende  $G_{i+1}$ -Zellen.
- $A$  führt beschränktes GLS-lookup zu  $A_{i+1}^B$  durch
- $A_{i+1}^B$  wird dann Server für  $B$  (Position aus Paket)

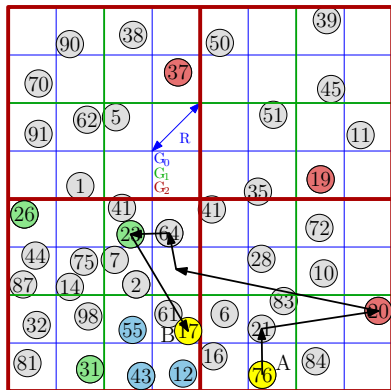


Bisher: Jede `publish`-Operation benachrichtigt alle Server!

## Lazy Publishing

Bei Bewegung innerhalb einer  $G_i$ -Zelle benachrichtige die Server in den  $G_i$  bis  $G_M$ -Zellen nicht.

- Es reicht, wenn entfernte Server an die alte Position weiterleiten



- ✓ Faire Lastverteilung
  - bei vernünftiger Verteilung der Knoten
- ✓ Fehlertoleranz
  - Ausfall einzelner Knoten betrifft nur wenig Information
- Verhältnismäßigkeit der Kommunikation
  - Kommunikationskosten abhängig von kleinster Gitterebene in der beide Knoten in gleicher Zelle liegen
  - dasselbe gilt bei Bewegungen über Gittergrenzen
- ? Skalierbarkeit
  - Anzahl Gitterebenen sicher in  $O(\log n)$

- Rückblick: Geographisches Routing
  - Denkanstoß: Anwendbarkeit in 3D
- Location Services
  - Definition und Anforderungen
  - Triviale Ansätze (Vor- und Nachteile)
- Grid Location System (GLS)
- **MLS**

GLS nutzt Georouting an *virtuelle* Knotenpositionen, um irgendeinen Knoten in einem Gebiet zu finden. Man kann auch an virtuellen Knotenpositionen Informationen speichern!

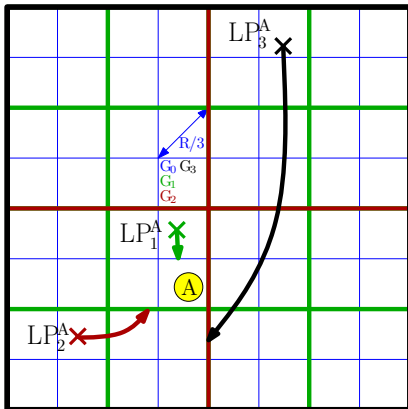
- a) dichtester Knoten ist verantwortlich
  - bewegen sich Knoten von Position weg, übergeben sie die Verantwortung
  - setzt voraus, dass Koordinaten nie „verwaisen“, dass z. B. immer ein Knoten im Abstand  $R_{\min}/3$  von jeder genutzten Position ist.
  
- b) dichtester Knoten auf umgebender Facette ist verantwortlich
  - Information wird bei allen Knoten der Facette regelmäßig aufgefrischt
  - Knoten, die sich wegbewegen, sind dann irgendwann nicht mehr beteiligt

## Gitter & Geohash

- $G_0, \dots, G_M$ , Faktor 1/3 enger
- Hashfunktion  $g$  berechnet zu jeder Gitterzelle  $C$  und Knoten-ID individuelle, **eindeutige Position**  
 $g(C, (ID)) \in C$

## publish

Für jedes  $i$  hinterlege in *eigener*  $G_i$ -Zelle  $C_i^A$  an Position  $LP_i^A = g(C_i^A, ID_A, i)$  nur  $C_{i-1}^A$ .



⇒ Position  $LP_i^A$  speichert in welcher Zelle  $C_{i-1}^A$   $A$  gesucht werden sollte!

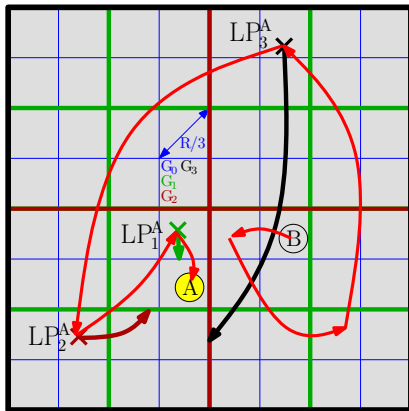


## lookup

In welchen Gitterzellen sollte man Pointer suchen?

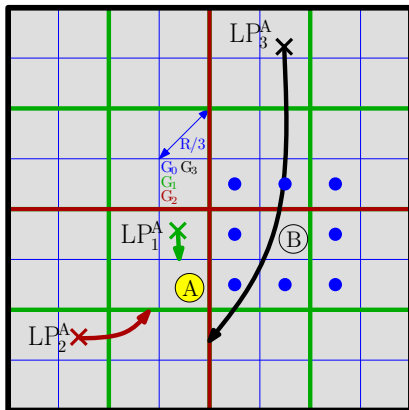
- in umgebenden Zellen?
- dann gibt es wieder weite Wege zu nahen Knoten!

Wie kann man das verhindern?



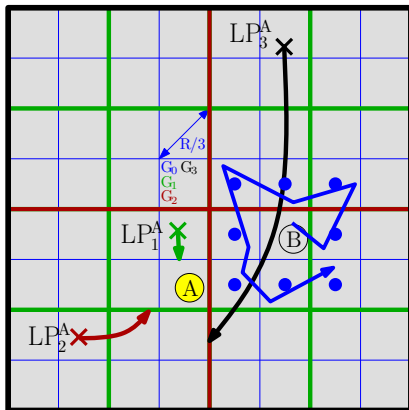
## MLS-lookup

- durchsuche umliegende Zellen spiralförmig
  - je 8 Zellen in  $G_0, G_1, \dots$
  - in jeder Zelle  $C$  liegt  $g(C, ID_A)$  an irgendeiner (bekannten) Position



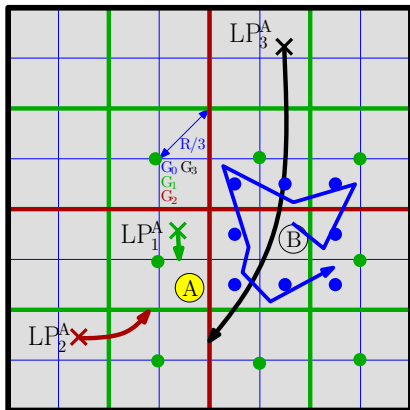
## MLS-lookup

- durchsuche umliegende Zellen spiralförmig
  - je 8 Zellen in  $G_0, G_1, \dots$
  - in jeder Zelle  $C$  liegt  $g(C, ID_A)$  an irgendeiner (bekannten) Position



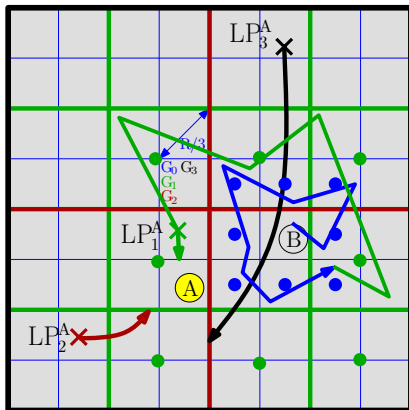
## MLS-lookup

- durchsuche umliegende Zellen spiralförmig
  - je 8 Zellen in  $G_0, G_1, \dots$
  - in jeder Zelle  $C$  liegt  $g(C, ID_A)$  an irgendeiner (bekannten) Position



## MLS-lookup

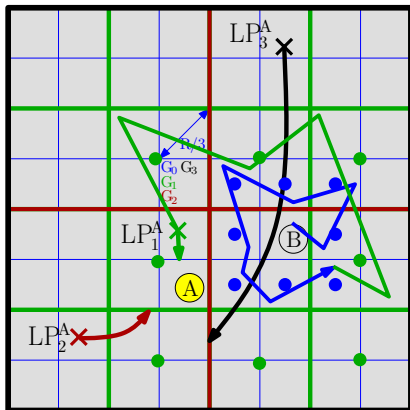
- durchsuche umliegende Zellen spiralförmig
  - je 8 Zellen in  $G_0, G_1, \dots$
  - in jeder Zelle  $C$  liegt  $g(C, ID_A)$  an irgendeiner (bekannten) Position



## Lemma

Sei  $C$  eine  $G_i$ -Zelle und  $ID$  eine beliebige ID. Jeder Punkt in  $C$  oder einer angrenzenden  $G_i$ -Zelle hat zu  $g(C, ID)$  maximal den Abstand  $2^i \cdot R$ .

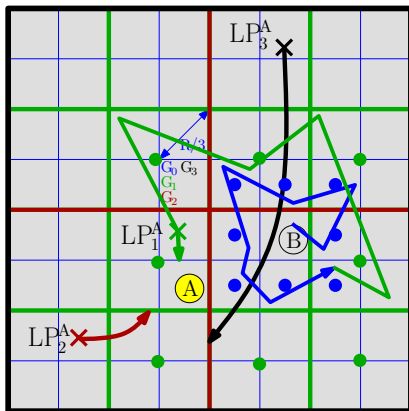
- Folgt aus Größe der Gitterzellen!



## Lemma

Sei  $B$  ein beliebiger Knoten. Die Summe der zu routenden Teilstrecken, bis ein `lookup` alle umliegenden  $G_j$ -Zellen abgesucht hat, ist in  $O(2^i \cdot R)$ .

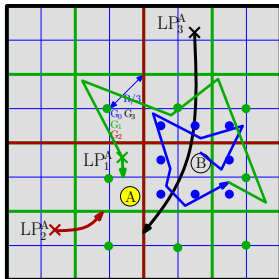
- die Suche zerfällt in  $G_j$ -Phasen,  $j \leq i$
- in jeder Phase Suche in 8 Zellen, jeweils aus derselben oder angrenzender Zelle
- $\sum_{j=0}^i 8 \cdot 2^j R < 16 \cdot 2^i R$ .



## Satz

Startet ein Knoten  $B$  ein `lookup` nach einem Knoten  $A$  in Abstand  $d$ , dann ist die Summe der zu routenden Teilstrecken in  $O(d)$ .

- Sei  $G_i$  das kleinste Gitter, in dem  $A$  und  $B$  in angrenzenden Zellen liegen
- beim Durchsuchen aller benachbarten  $G_i$ -Zellen wird  $LP_i^A$  gefunden
- Teilstrecken bis dahin unter  $2^{i+4}R$
- $A$  und  $B$  sind nicht in benachbarten  $G_{i-1}$ -Zellen  
 $\Rightarrow$  Abstand mindestens  $d > 2^{i-1}R/6$
- Teilstrecken in  $O(d)$

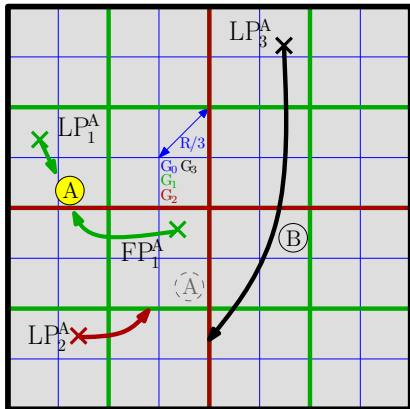




## Lazy Publishing bei MLS

$LP_i^A$ -Pointer wird erst geändert, wenn sie nicht einmal in angrenzende  $G_{i-1}$ -Zelle zeigt. Alter  $LP_{i-1}$  wird zu Forwarding-Pointer!

- Auch an groben Gittergrenzen wird Oszillation unkritisch! (ohne Beweis)
- noch mehr Pointer, um Racing Conditions zu lösen (nicht hier)
  - beweisbare Korrektheit bei langsamen Bewegungen



- Location Services: Proaktives Georouting mit IDs
  - Anforderungen: Lastverteilung, Robustheit, Verhältnismäßigkeit
- GLS: Hierarchische Server
  - individuelle Server für jeden Knoten
  - Stärke: Gute Aufgabenverteilung
  - Schwäche: Bewegungen/Routen über Gittergrenzen unverhältnismäßig teuer
- MLS: *Serverpositionen* statt Knoten
  - Stärke: beweisbare Schranken in Bewegung/Entfernung
  - Schwäche: Setzt sehr hohe Knotendichte voraus

- 1 J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, R. Morris: *A scalable location service for geographic ad hoc routing*. In: MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, 2000.
- 2 R. Flury, R. Wattenhofer: *MLS: an efficient location service for mobile ad hoc networks*. In: MobiHoc '06: Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2006.

# Anhang: Erweitertes Beispiel für GLS

	70 72,76,81 82,84,87	1,5,6,10,12 14,37,62,70 90,91			19,35,37,45 50,51,82	
	<b>A: 90</b>	38			39	
1,5,16 63,90,91	70		16 17 19,21 23,26,28,31	19,35,39,45 51,82	39,41,43	
			32,35			
	70		<b>37</b>	50		45
1,62,70,90	1,5,16,37,39 41,43,45,50 51,55,61,91	1,2,16,37,62 70,90,91			35,39,45,50	19,35,39,45 50,51,55,61 62,63,70,72 76,81
	91	62	5		51	11
	62,91,98					
	1				19,20,21,23 26,28,31,32 51,82	1,2,5,6,10,12 14,16,17,82 84,87,90,91 98
					35	<b>19</b>
14,17,19,20 21,23,26,87		2,17,23,63	2,17,23,26 31,32,43,55 61,62	28,31,32,35 37,39		10,20,21,28 41,43,45,50 51,55,61,62 63,70
	<b>26</b>	<b>23</b>	<b>63</b>	41		<b>72</b>
14,23,31,32 43,55,61,63 81,82,84	2,12,26,87 98	1,17,23,63,81 87,98	2,12,14,16 23,63		6,10,20,21 23,26,41,72 76,84	6,72,76,84
	87	14	<b>2</b>	<b>B: 17</b>		28
						10
31,81,98	31,32,81,87 90,91	12,43,45,50 51,61	12,43,55	1,2,5,21,76 84,87,90,91 98	6,10,20,76	6,10,12,14 16,17,19,84
	32	98	55	61	6	21
						<b>20</b>
31,32,43,55 61,63,70,72 76,98	2,12,14,17 23,26,28,32 81,98	12,14,17,23 26,31,32,35 37,39,41,55 61	2,5,6,10,43 55,61,63,81 87,98		6,21,28,41 72	20,21,28,41 72,76,81,82
	81	<b>31</b>	<b>43</b>	12	<b>A: 76</b>	84

(Entnommen aus "J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, R. Morris: A scalable location service for geographic ad hoc routing").