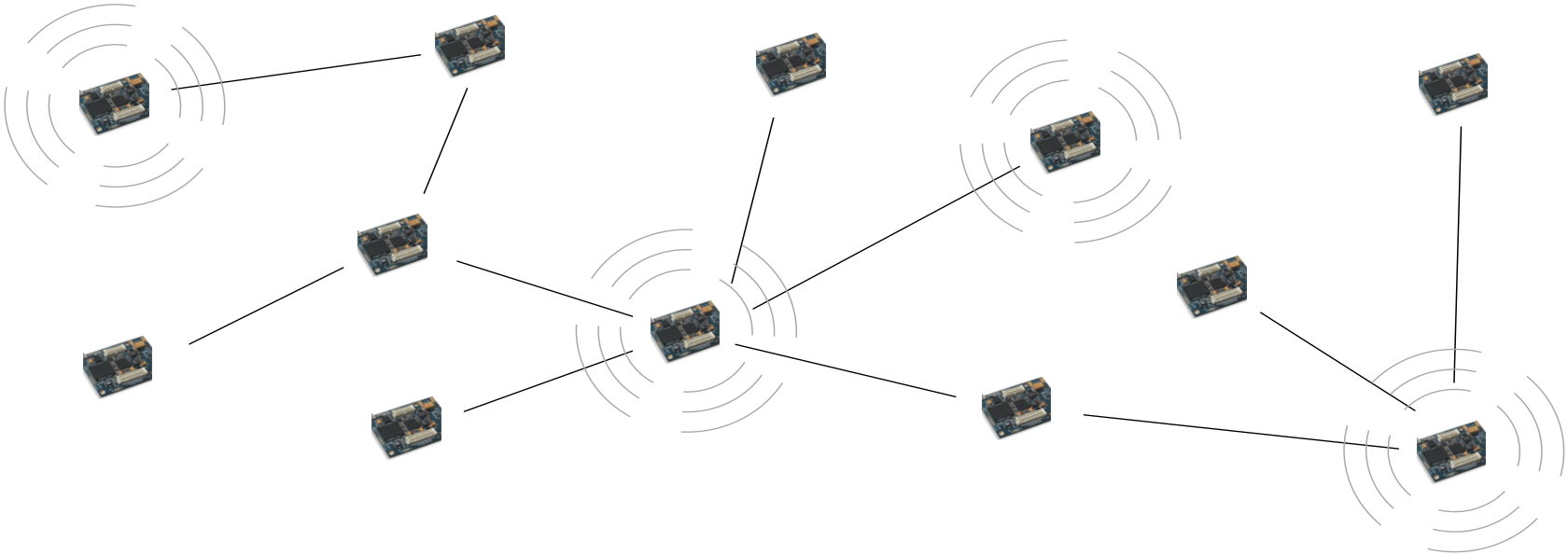
Karlsruher Institut für Technologie

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)



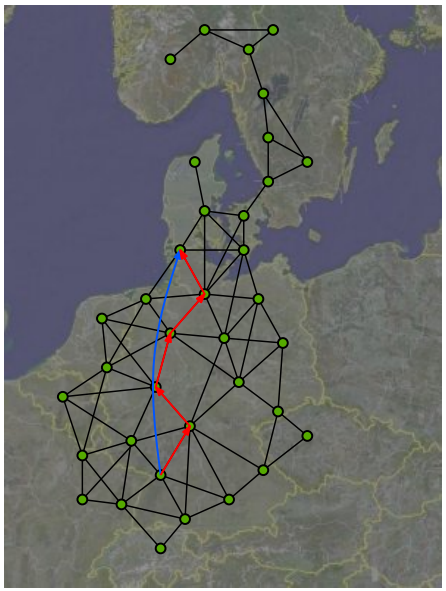
Überblick

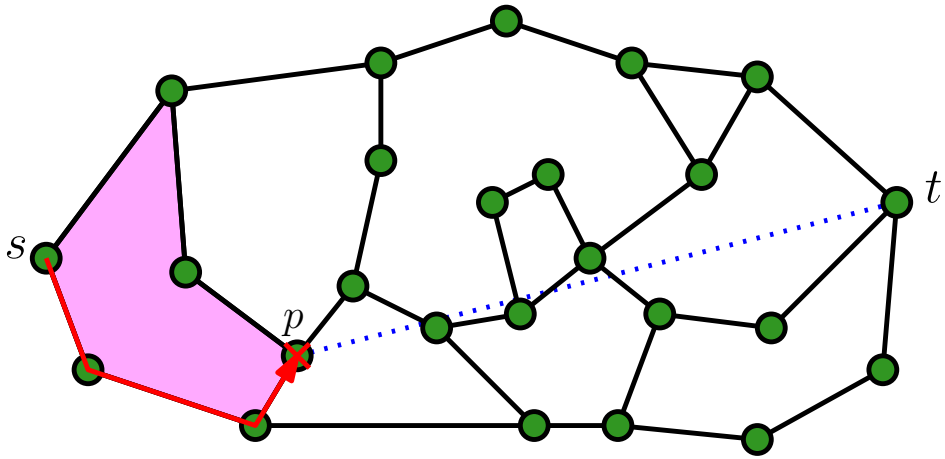
- Rückblick: Geographisches Routing
  - Denkanstoß: Anwendbarkeit in 3D
- Location Services
  - Definition und Anforderungen
  - Triviale Ansätze (Vor- und Nachteile)
- Grid Location System (GLS)
- MLS

Rückblick: Georouting

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)





Rückblick: Georouting

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?

Überblick

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

Location Service (LS)

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

LS – alternative Sichtweise

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

Was muss ein LS können?

`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.

Einfache Lösung I: Broadcasts

`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 $\Omega(n)$!

Einfache Lösung II: Zentraler Server

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

Anforderungen an Location Service

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.

„Typische“ Annahmen für LS-Design

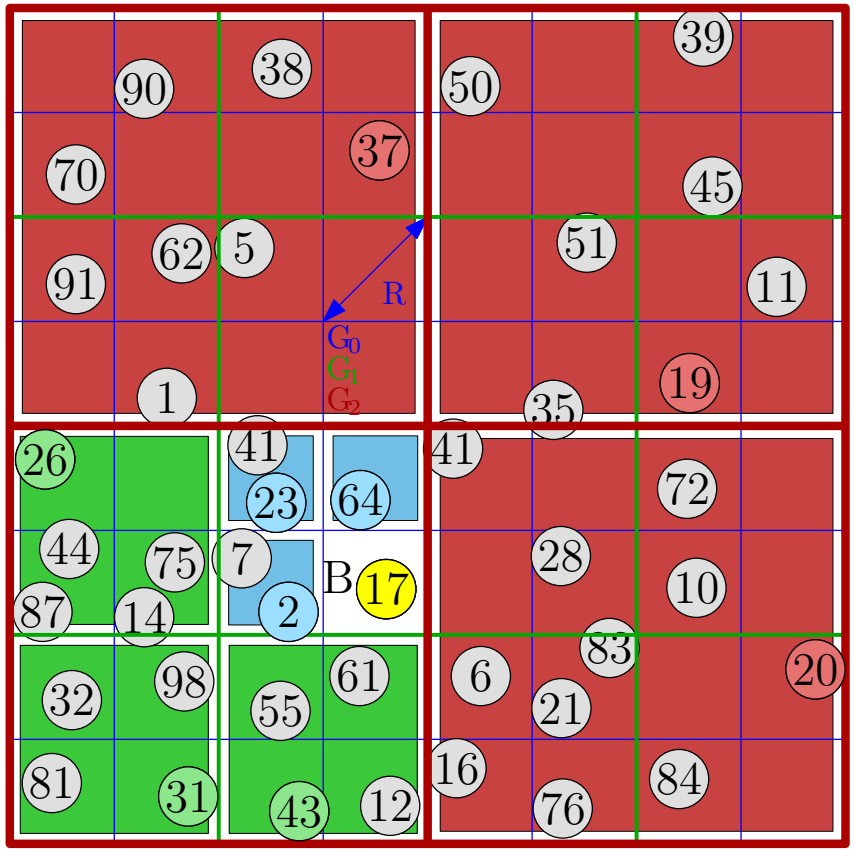
- zuverlässige lokale Kommunikation
- verhältnismäßig hohe Knotendichte
  - häufige Annahme: $D \in O(\sqrt{n})$
- oft Gebiet bekannt
- eingeschränkte Mobilität
  - keine großen Bewegungen zwischen elementaren Operationen
- eindeutige, geordnete Knoten-IDs aus bekanntem Intervall
  - im Zweifel: IDs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

Überblick

- 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.

Auswahl der Server

Grundidee:

- Knoten A, der Position eines Knotens B sucht, muss einen der Server von B finden

  $\Rightarrow$ As Suche nach Server von B und Bs Server-Auswahl müssen ähnlich ablaufen

- Gemeinsames Wissen von A und B:
  - Gitter-Einteilung
  - ID von B

$\Rightarrow$ Strategie: B wählt Knoten als Location Server, deren ID ähnlich zur ID von B ist

Auswahl der Server

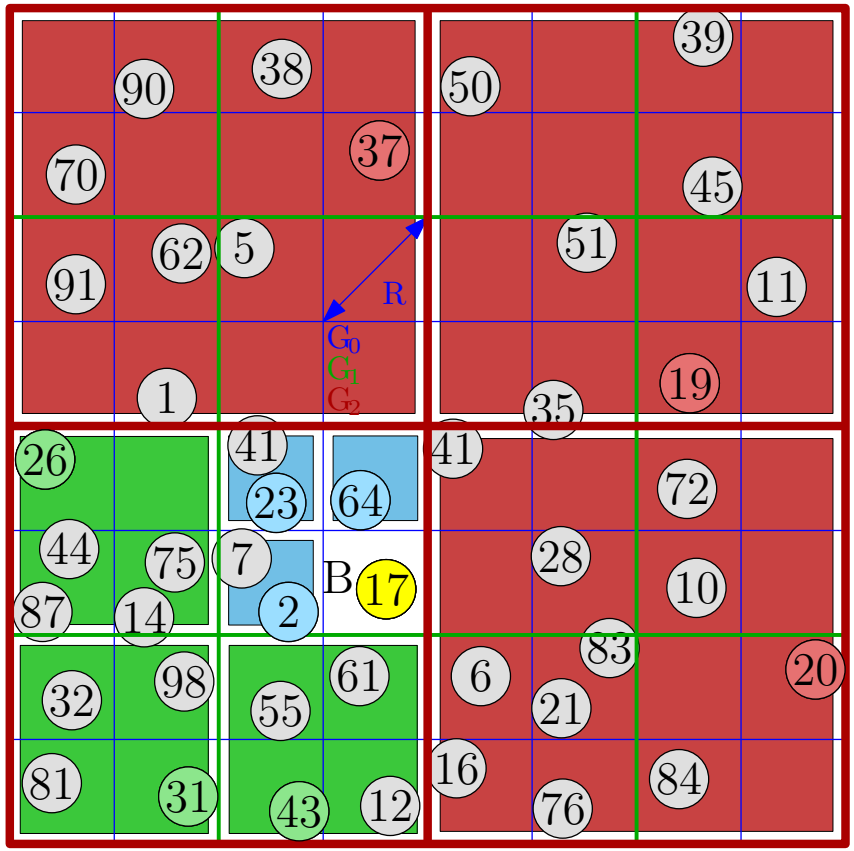
Serverauswahl

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

$$\texttt{ID}_X - \texttt{ID}_B \mod \texttt{ID}_{\max}$$

minimiert.

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



Garantien nur bei gleichverteilten Knoten und zufälligen IDs!

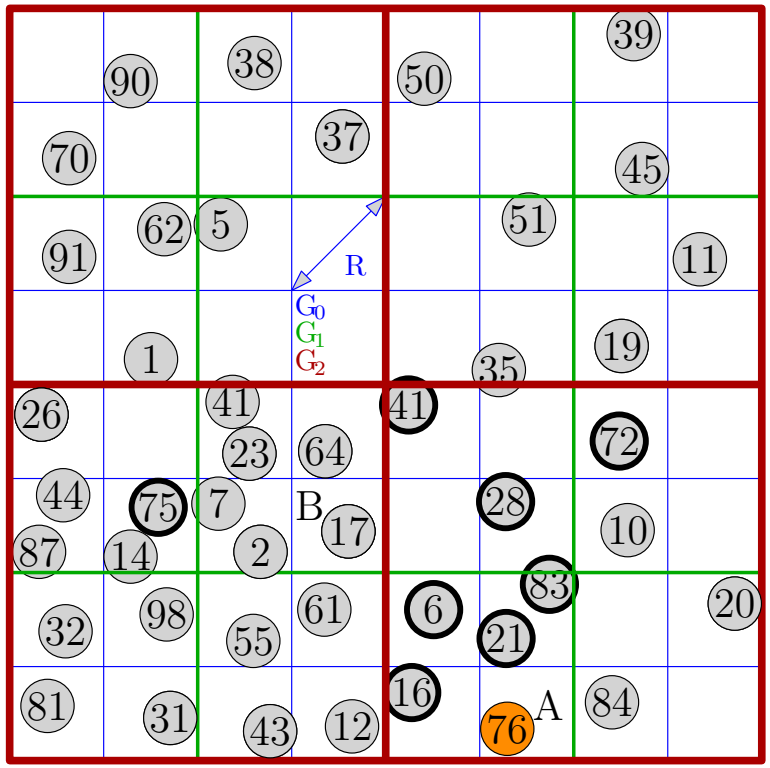
Auswahl der Server

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 $\texttt{ID}_A > \texttt{ID}_X \geq \texttt{ID}_B$?[1]
- 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 $76 > \texttt{ID}_B \geq 21$
- In Gitter G_2: Für Knoten *B* mit $76 > \texttt{ID}_B \geq 72$

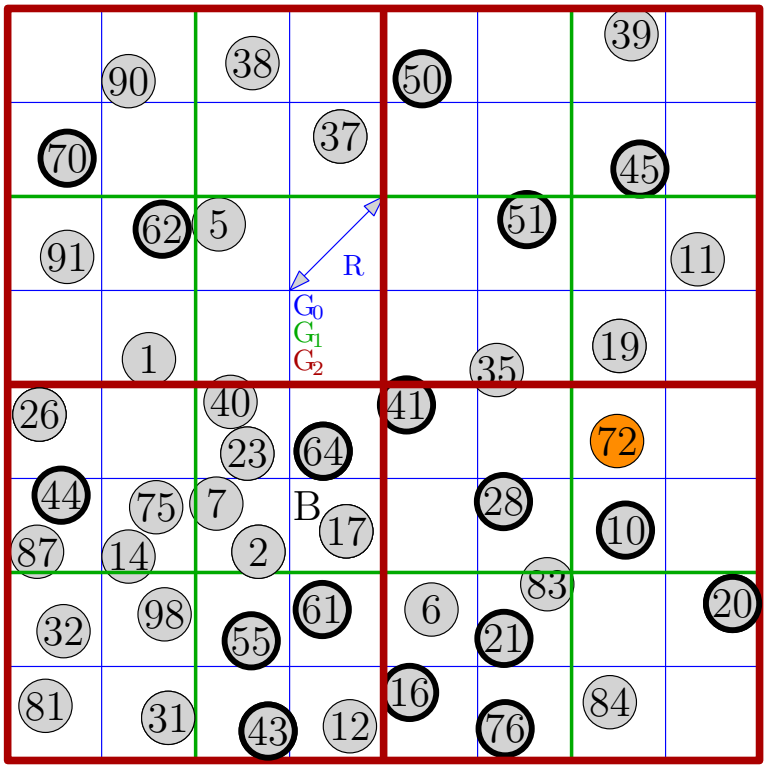
[1] Lies: Von $\texttt{ID}_B$ an aufsteigend im Restklassenring $\mathbb{Z}/\texttt{ID}_{\max}$ kommt $\texttt{ID}_X$ vor $\texttt{ID}_A$.

Auswahl der Server

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 $\texttt{ID}_A > \texttt{ID}_X \geq \texttt{ID}_B$?[1]
- Falls nicht, ist *A* Server von *B*



- In Gitter G_0: Ja, da *A* einziger Knoten in *A*s Zelle
- In Gitter G_1: Für Knoten *B* mit $72 > \texttt{ID}_B \geq 10$
- In Gitter G_2: Für Knoten *B* mit $72 > \texttt{ID}_B \geq 41$

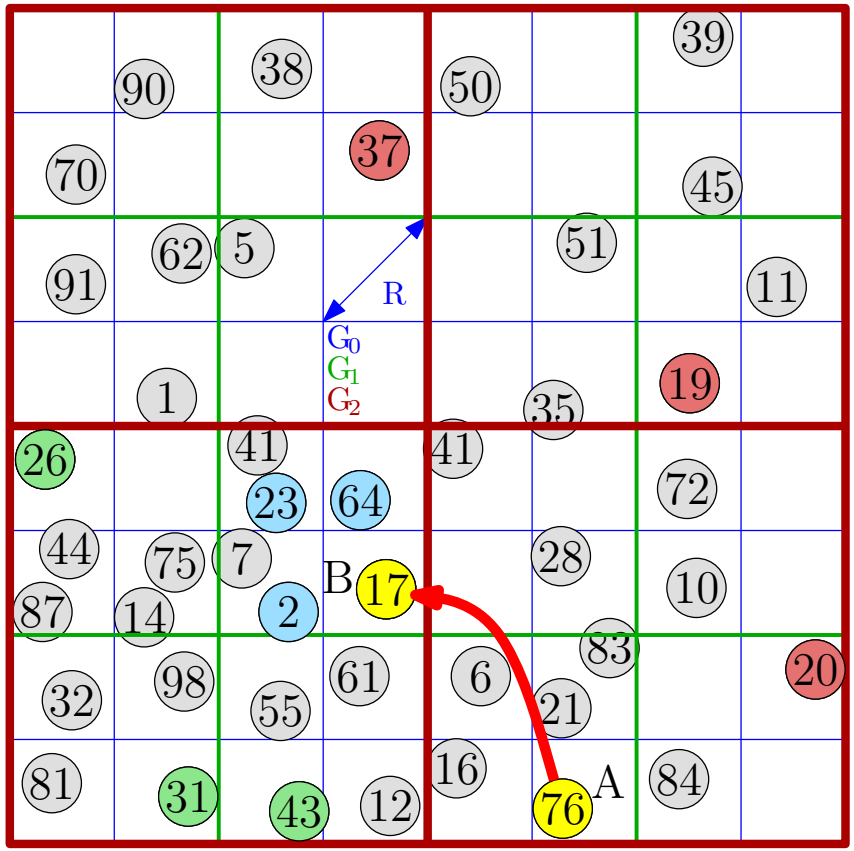
[1] Lies: Von $\texttt{ID}_B$ an aufsteigend im Restklassenring $\mathbb{Z}/\texttt{ID}_{\max}$ kommt $\texttt{ID}_X$ vor $\texttt{ID}_A$.

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?

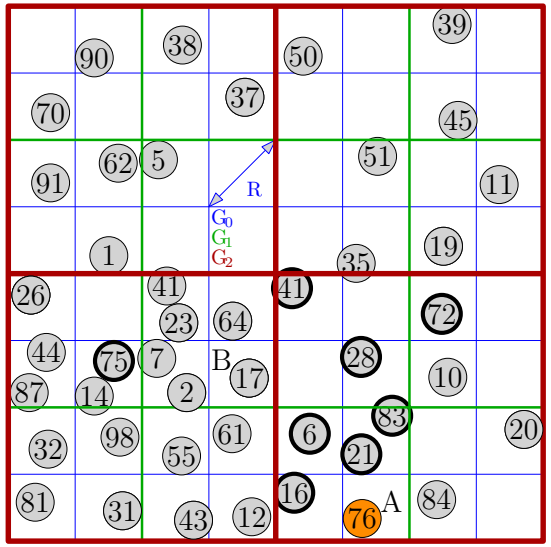
- Knoten *A* sucht Knoten *B*.



Lookup (Versuch 1)

Satz

Seien A, B beliebige Knoten. A ist Server für einen Knoten X mit $\texttt{ID}_A > \texttt{ID}_X \geq \texttt{ID}_B \mod \texttt{ID}_{\max}$.



- sind A und B in benachbarten G_i-Zellen,
  - liegt kein X mit $\texttt{ID}_A > \texttt{ID}_X \geq \texttt{ID}_B$ in As Zelle $\Rightarrow$ A ist Server für B.
  - liegt ein größtes X mit $\texttt{ID}_A > \texttt{ID}_X \geq \texttt{ID}_B$ in As Zelle $\Rightarrow$ A ist Server für X.

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

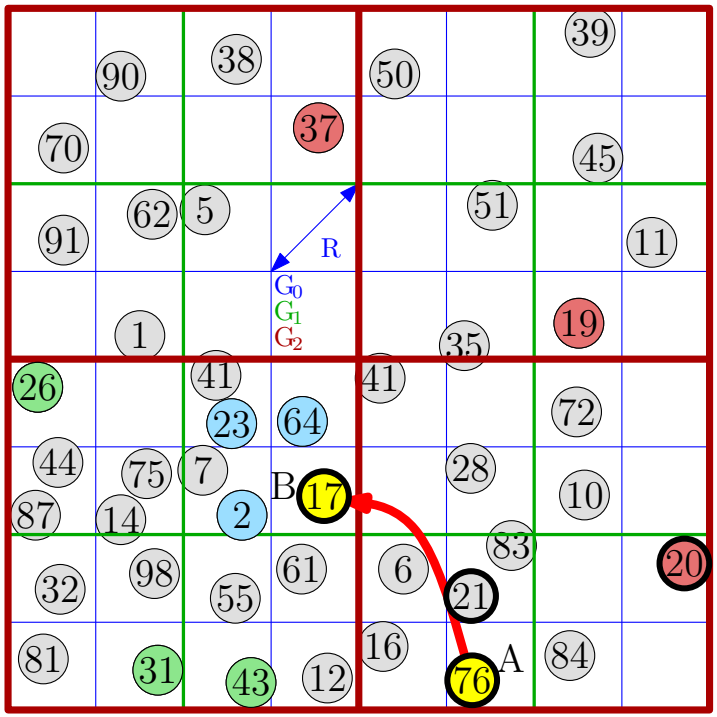
GLS-lookup

Definition

Seien A, B beliebige Knoten. Dann sei A_i^B der Knoten in der G_i-Zelle von A, der $\texttt{ID}_{A_i^B} - \texttt{ID}_B$ 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 $A_k^B = B$.

GLS-lookup

Definition

Seien A, B beliebige Knoten. Dann sei A_i^B der Knoten in der G_i-Zelle von A, der $\texttt{ID}_{A_i^B} - \texttt{ID}_B$ minimiert.

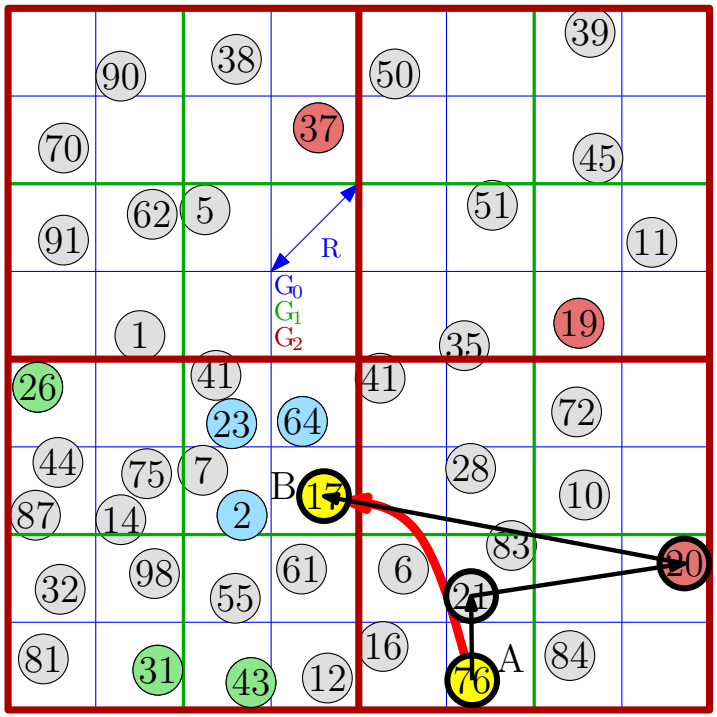
Lemma

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.
- $\texttt{ID}_{A_i^B}$ ist in G_i-Zelle von A am wenigsten größer als $\texttt{ID}_B$

$\Rightarrow$ 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



GLS-lookup

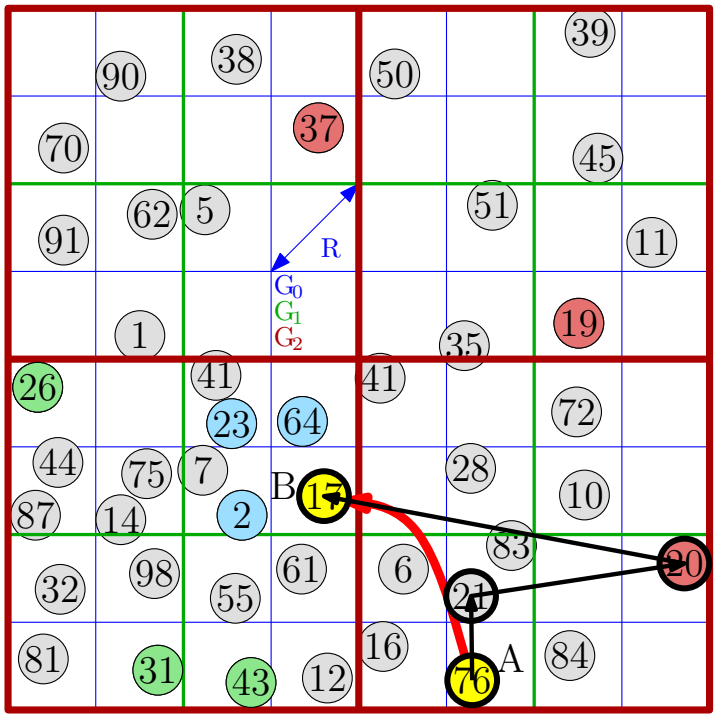
Definition

Seien A, B beliebige Knoten. Dann sei A_i^B der Knoten in der G_i-Zelle von A, der $\texttt{ID}_{A_i^B} - \texttt{ID}_B$ minimiert.

GLS-lookup

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

- 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 Zielkoordinate einschließt
  - lernt alle Knoten der Facette kennen
  - im Gabriel Graph: sieht dichtesten Knoten zur Zielkoordinate

Lemma

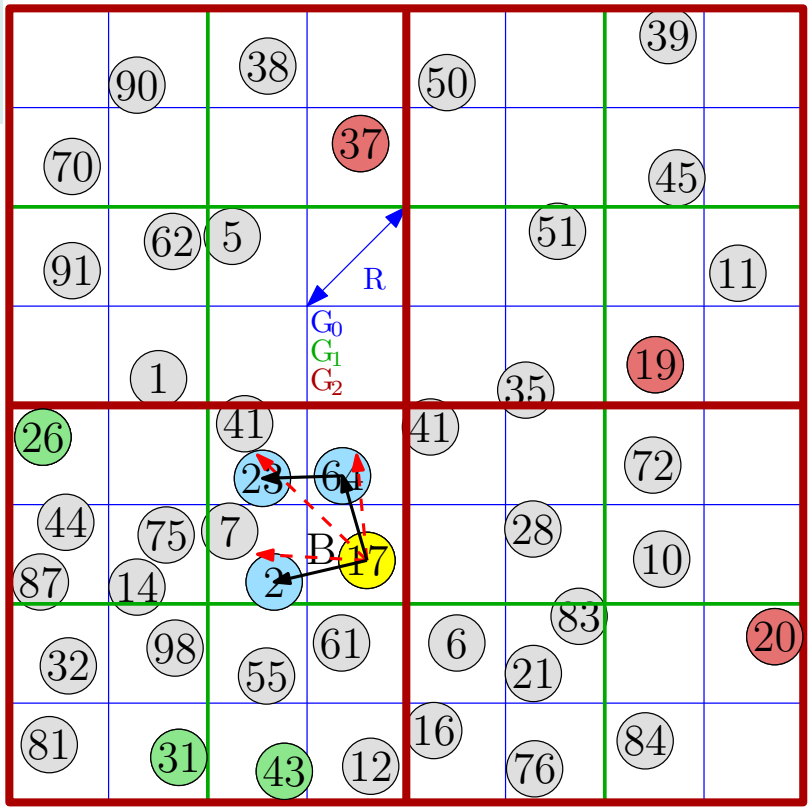
Wird ein Paket zur Mitte einer Gitterzelle C geroutet, in der ein Knoten liegt, dann erreicht das Paket mindestens einen Nachbarn eines solchen Knotens.

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

Initialisierung / `publish`-Operation

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

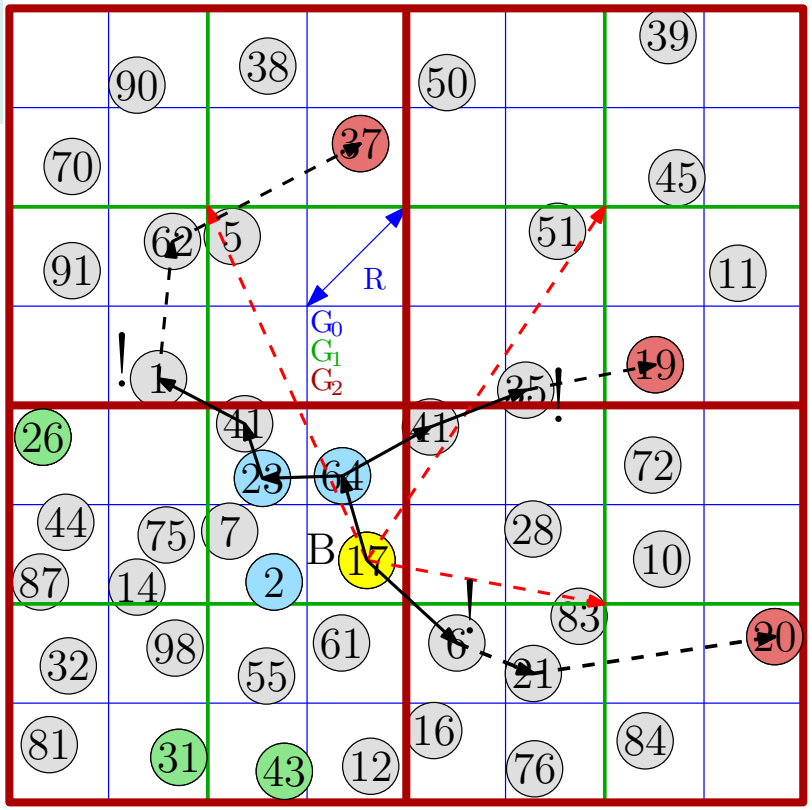
- Ebene für Ebene!
- G_0-Zellen-Server: Schicke Paket an Zentrum der Zellen
  - in nichtleerer Zelle kommt das Paket bei einem Knoten der Zelle an
  - der kennt alle Knoten der Zelle und benachrichtigt den zuständigen Knoten



Initialisierung / `publish`-Operation

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^B_{i+1} durch
- A^B_{i+1} wird dann Server für B (Position aus Paket)



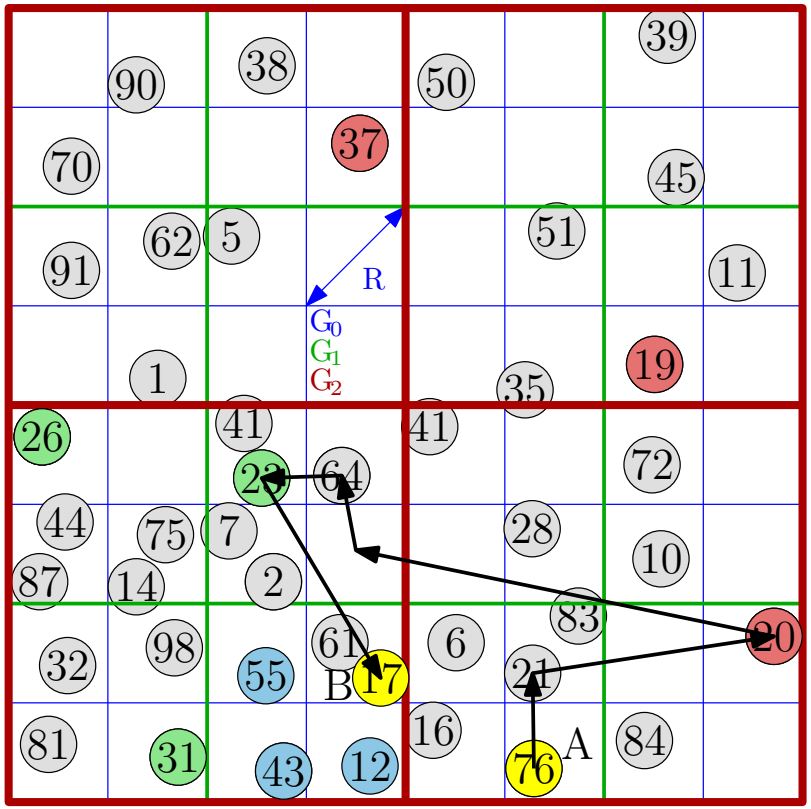
Lazy Publishing bei Dynamik

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



Bestandsaufnahme

- ✓ 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 $O(\log n)$

Überblick

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

Geographic Hash Tables

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

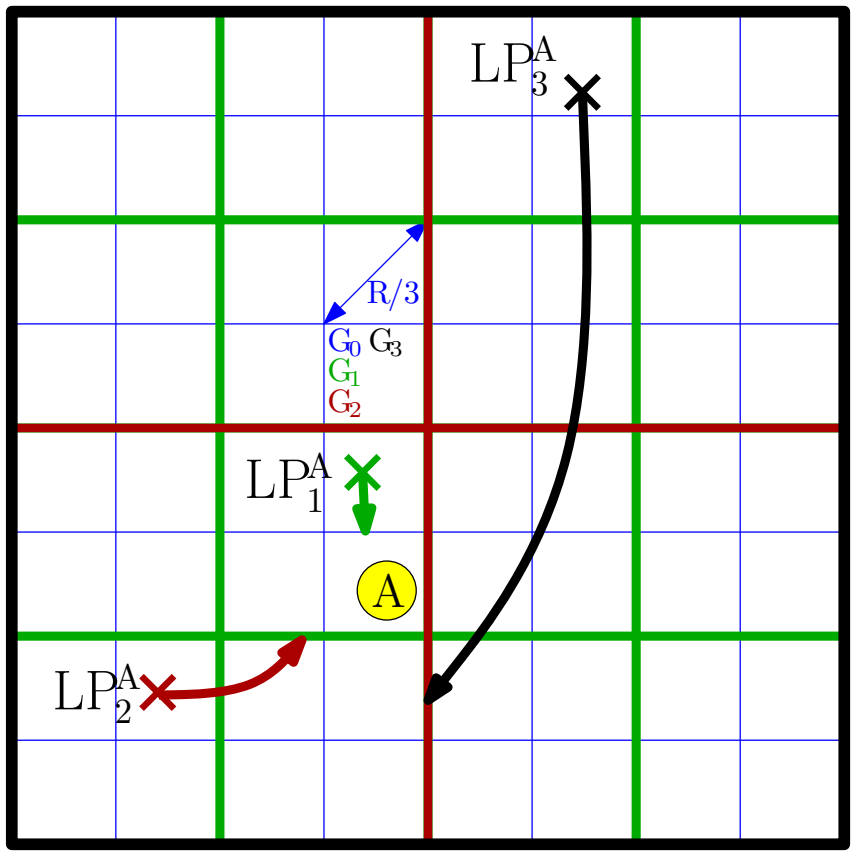
a) dichtester Knoten ist verantwortlich
   - bewegen sich Knoten von Position weg, übergeben sie die Verantwortung
   - setzt voraus, dass Koordinaten nie „verwaisen“, dass z. B. immer ein Knoten im Abstand $R_{\min}/3$ von jeder genutzten Position ist.

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

MLS

Gitter & Geohash

- $G_0, \ldots G_M$, 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 $LP_i^A = g(C_i^A, \texttt{ID}_A, i)$ nur C_{i-1}^A.

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

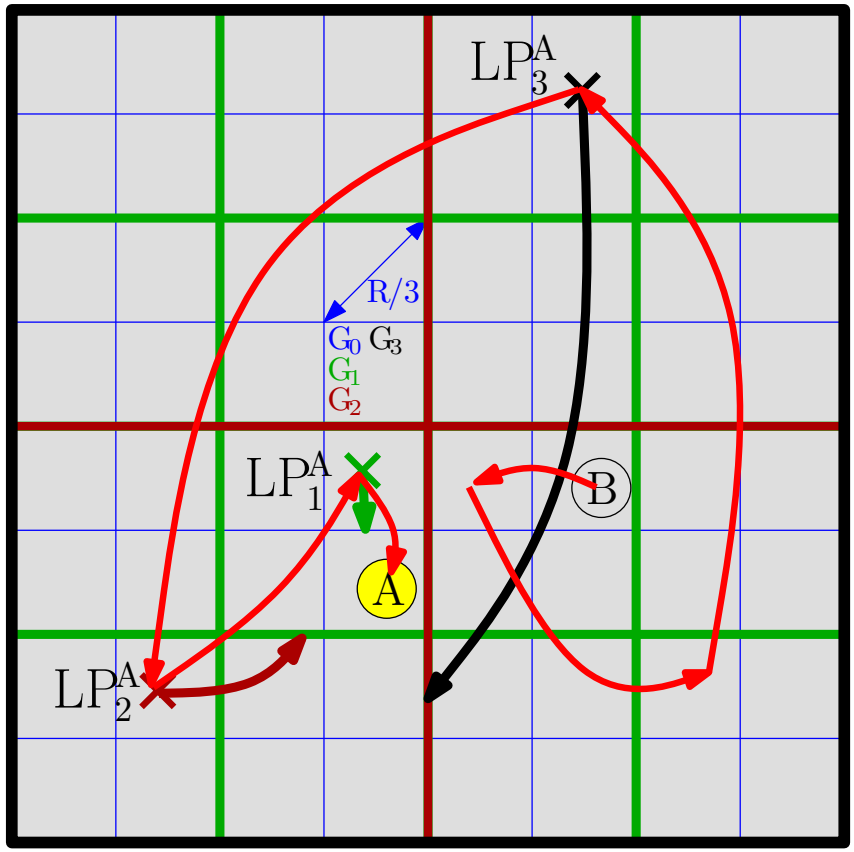
MLS

lookup

In welchen Gitterzellen sollte man Pointer suchen?

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

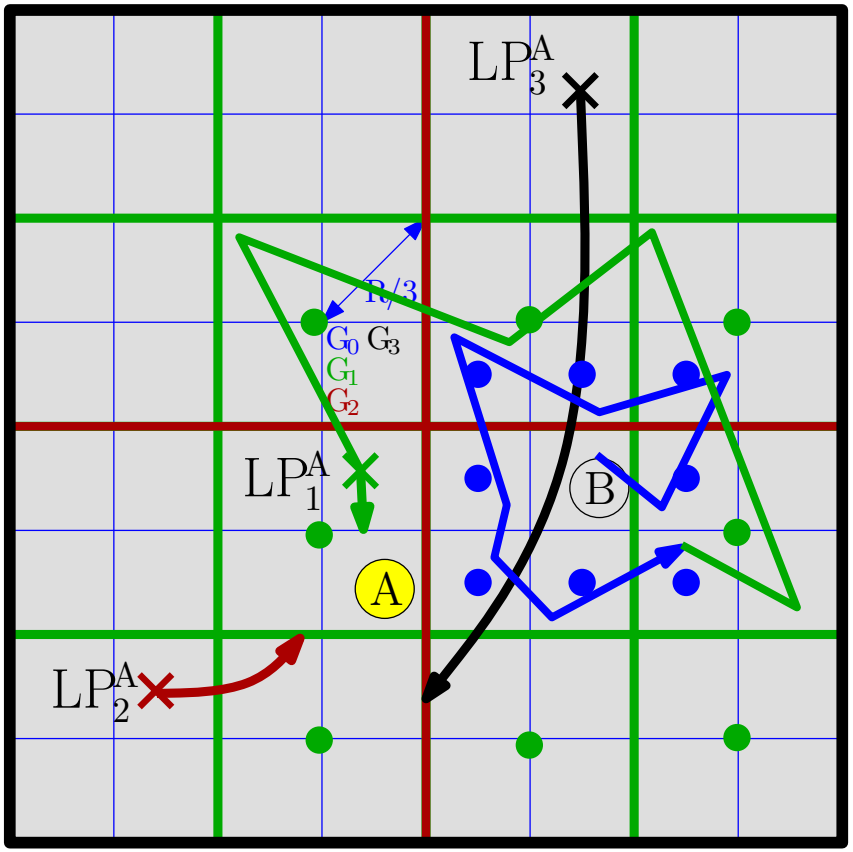
Wie kann man das verhindern?



Such*spirale*

MLS-`lookup`

- durchsuche umliegende Zellen spiralförmig
  - je 8 Zellen in G_0, G_1,...
  - in jeder Zelle C liegt $g(C, \texttt{ID}_A)$ an irgendeiner (bekannten) Position

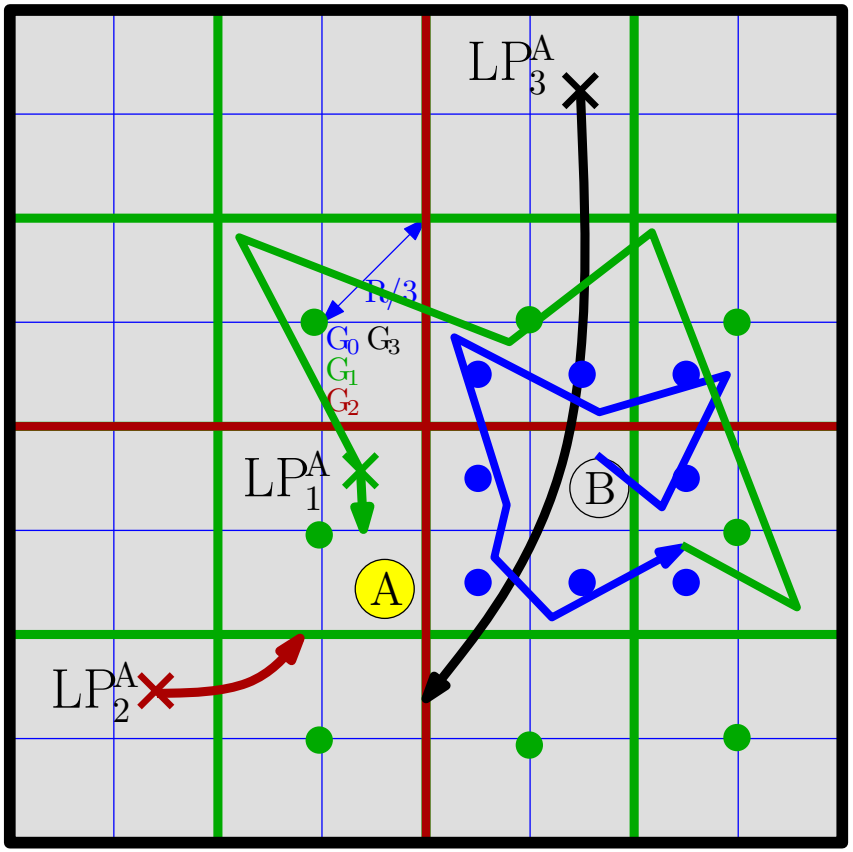


Such*spirale*

Lemma

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

- Folgt aus Größe der Gitterzellen!

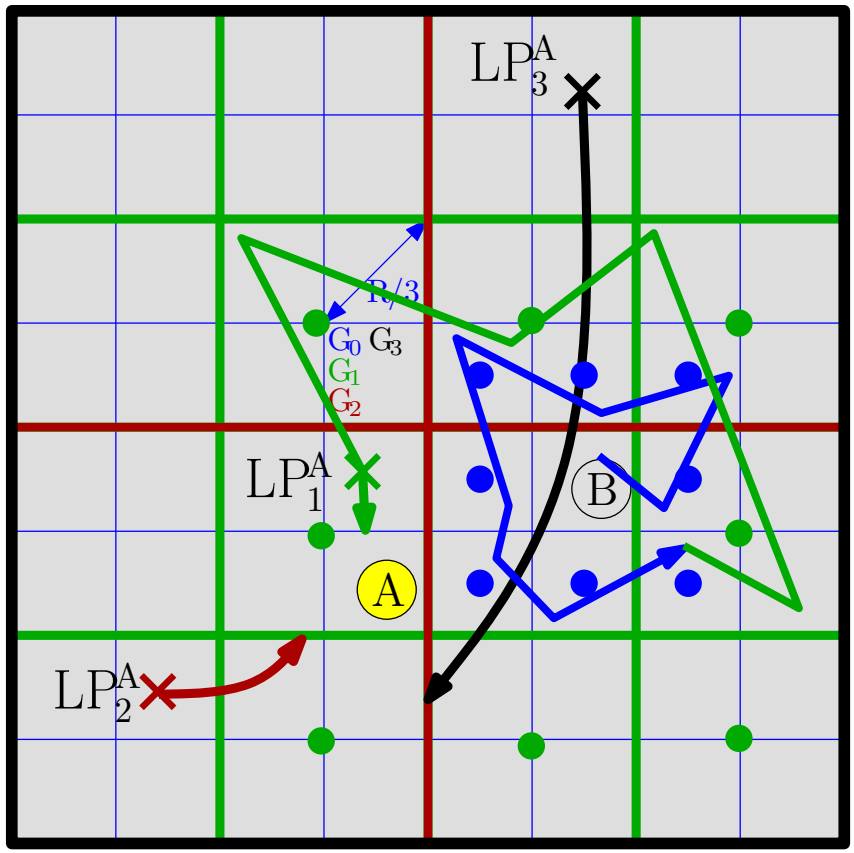


Suchspirale

Lemma

Sei B ein beliebiger Knoten. Die Summe der zu routenden Teilstrecken, bis ein `lookup` alle umliegenden G_i-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
- $\sum_{j=0}^{i} 8 \cdot 2^j R < 16 \cdot 2^i R$.

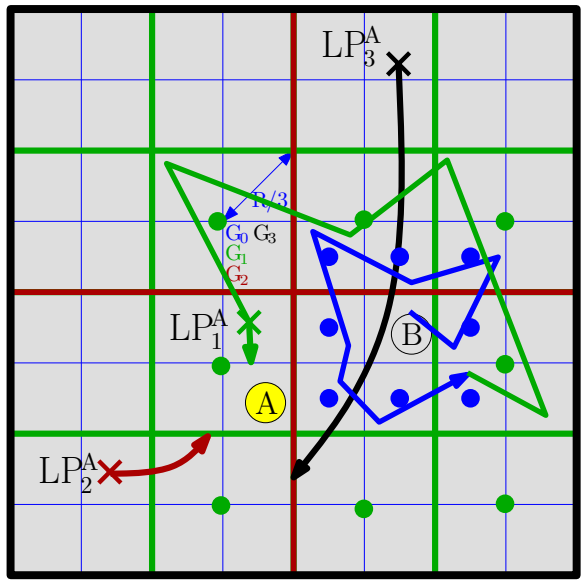


Suchspirale

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 $O(d)$

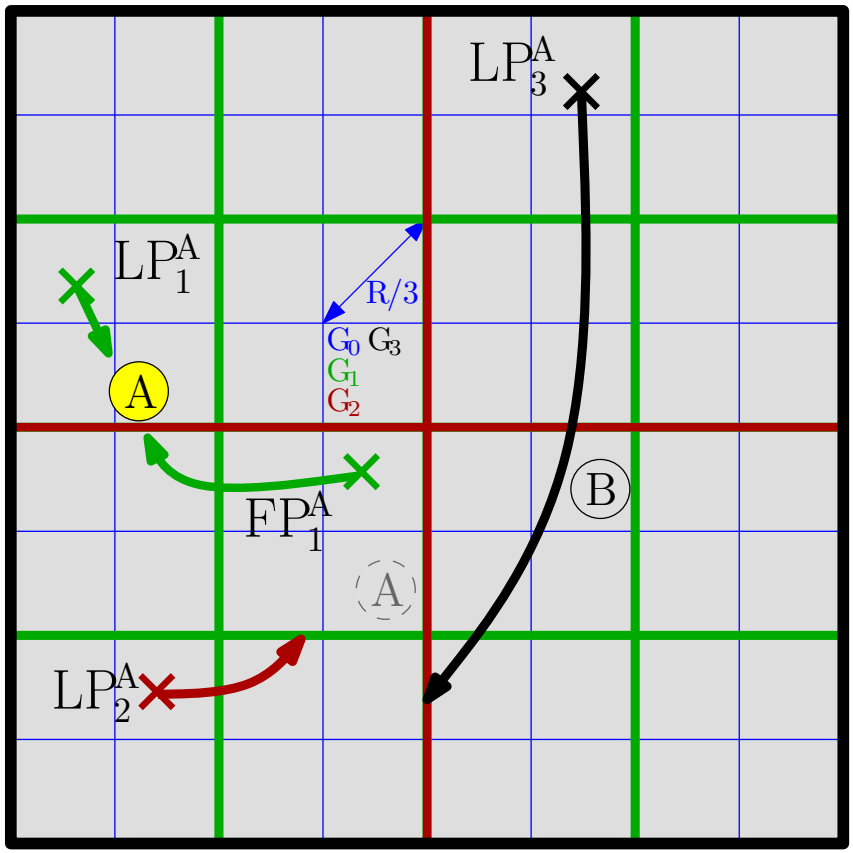


Mobilität & Forwarding (vereinfacht)

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



Was mitnehmen?

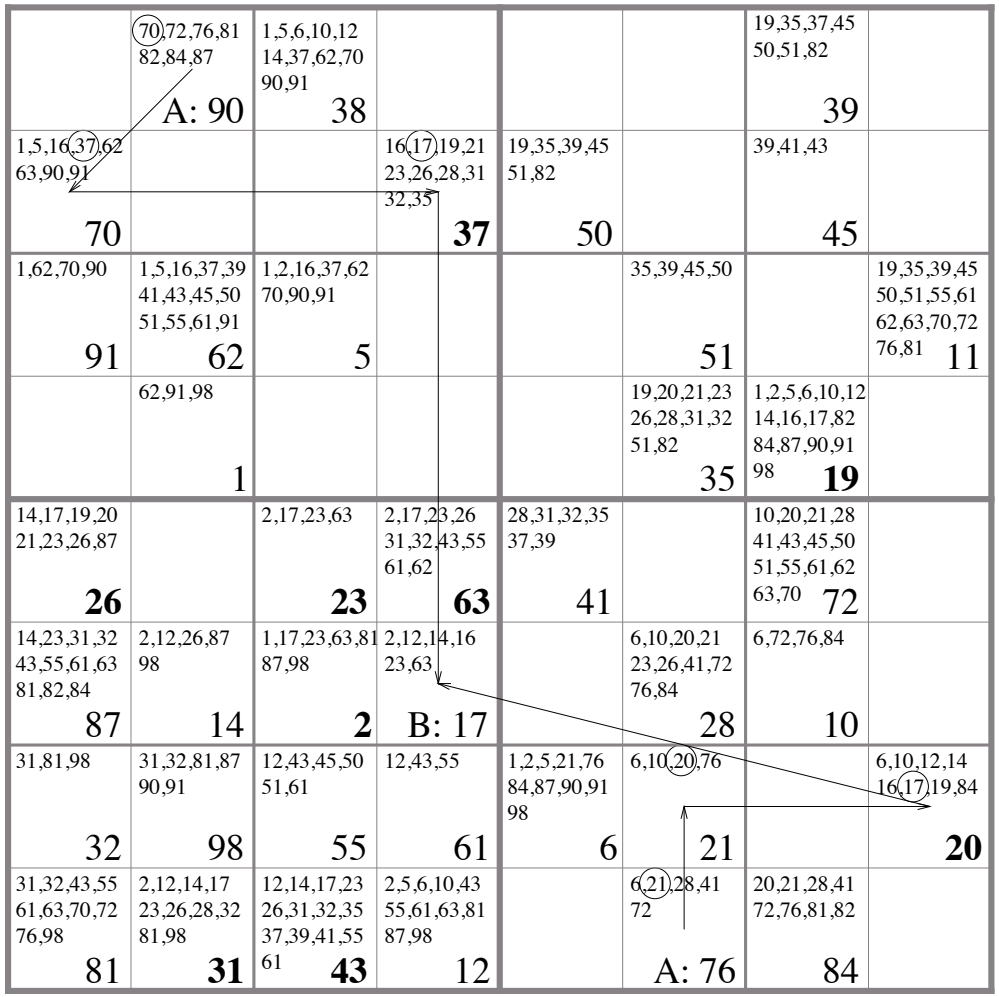
- Location Services: Proaktives Georouting mit IDs
  - Anforderungen: Lastverteilung, Robustheit, Verhältnismäßigkeit
- GLS: Hierarchische Server
  - individuelle Server für jeden Knoten
  - Stärke: Gute Aufgabenverteilung
  - Schwäche: Bewegungen/Routen über Gittergrenzen unverhältnismäßig teuer
- MLS: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

Literatur

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



(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”).