Algorithmen für Ad-hoc- und Sensornetze

VL 03 – Location Services

Dr. rer. nat. Bastian Katz

Lehrstuhl für Algorithmik I
Institut für theoretische Informatik
Universität Karlsruhe (TH)
Karlsruher Institut für Technologie

6. April 2009
(Version 3 vom 7. Mai 2009)

Organisatorisches

- Die Vorlesung am 27. Mai fällt aus!
  - Inhalte vom 20. und 27. werden zusammengefasst bzw. auf letzte VL verschoben.
- Die Vorlesung am 24. Juni muss verschoben werden!
  - Alternativtermine: 22., 23. (vm.), 29., 30.
  - nächste Woche Umfrage, im Zweifelsfall zwei Termine!

Erinnerung: 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?

Location Service

Definition

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

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

Location Service — alternative Sichtweise

Alternative Sicht

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

- ≫ Sender schickt Paket mit Ziel-ID ab
- ≫ LS leitet Paket an entsprechende Zielkoordinate
- ≫ (wenn Sender eigene Adresse mitschickt, kann die Antwort dann direkt per Georouting kommen)

Was muss ein Location Service können?

`publish` : Veröffentlichung von Knotenpositionen
- ≫ Knoten, die sich bewegen, müssen das mitteilen
- ≫ Positionsinformationen 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 Lösungen 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 Lösungen 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

„Typische“ Randbedingungen

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

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

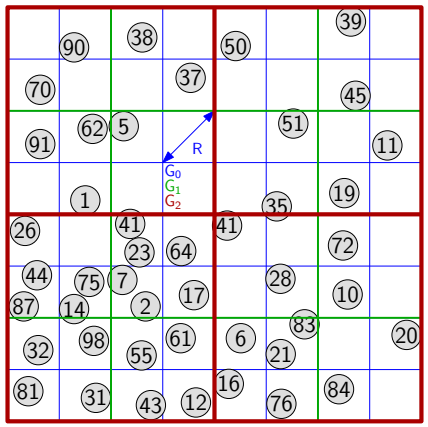
Anfragen zu nahen Knoten verursachen nur geringe Kommunikation. Idealerweise: Geringe Bewegungen erzeugen nur geringe Kommunikation.

Skalierbarkeit

Kosten wachsen möglichst gering in der Knotenzahl

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
- Nachbarzellen in G_i sind die Zellen, die in G_{i+1} zusammengefasst werden!

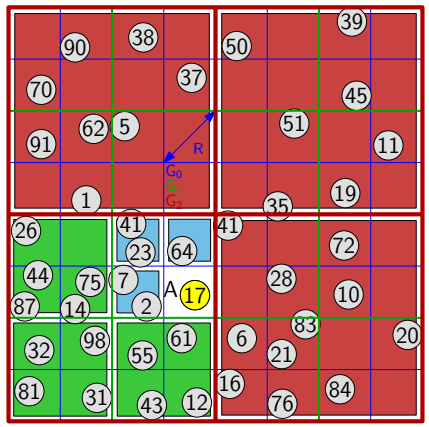


Jeder Knoten wählt einen Server pro Nachbarzelle in jedem G_i!

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
- Nachbarzellen in G_i sind die Zellen, die in G_{i+1} zusammengefasst werden!



Jeder Knoten wählt einen Server pro Nachbarzelle in jedem G_i!

Das sind $3M$ Server für jeden Knoten.

Auswahl der Server

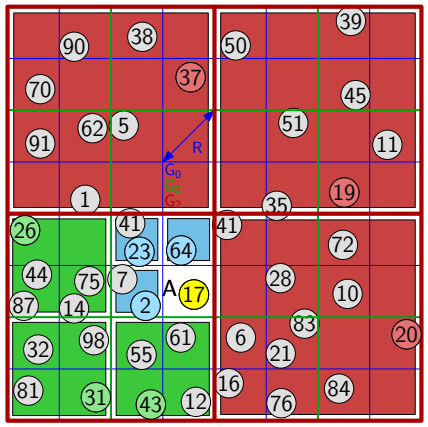
Serverauswahl

In einer Zelle ist immer der Knoten X Server für einen Knoten mit A, der

$$\mathtt{ID}_X - \mathtt{ID}_A \mod \mathtt{ID}_{\max}$$

minimiert.

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



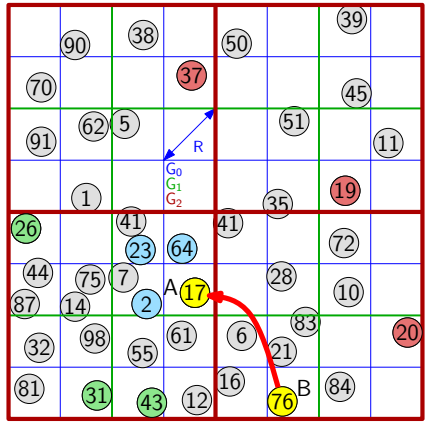
Garantien nur bei gleichverteilten Knoten und zufälligen IDs!

Ein kleiner Zeitsprung

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 B sucht Knoten A.

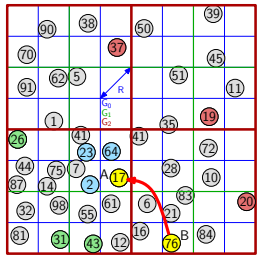


Lookup (Versuch 1)

Satz

Seien A, B beliebige Knoten. B ist Server für einen Knoten X mit $\mathtt{ID}_B > \mathtt{ID}_X \geq \mathtt{ID}_A \mod \mathtt{ID}_{\max}$[1].

- ≫ jeder Knoten ist Server von einem Knoten X mit $\mathtt{ID}_B \geq \mathtt{ID}_X \geq \mathtt{ID}_A$, ggf. sich selbst.
- ≫ sind A und B in benachbarten G_i-Zellen,
  - ≫ liegt kein X mit $\mathtt{ID}_B > \mathtt{ID}_X \geq \mathtt{ID}_A$ in Bs Zelle ⇒ B ist Server für A.
  - ≫ liegt ein größtes X mit $\mathtt{ID}_B > \mathtt{ID}_X \geq \mathtt{ID}_A$ in Bs Zelle ⇒ B ist Server für X.



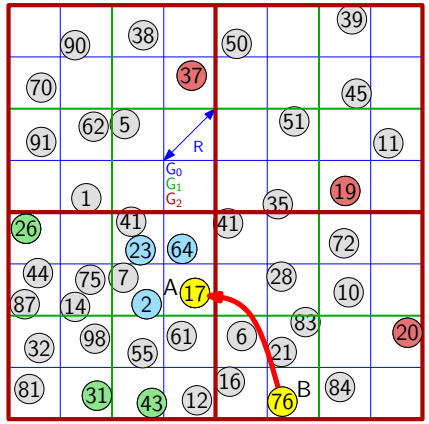
[1] So zu lesen: Wenn man die IDs von $\mathtt{ID}_A$ an aufsteigend im Restklassering $\mathbb{Z}/\mathtt{ID}_{\max}\mathbb{Z}$ abzählt, kommt erst $\mathtt{ID}_X$, und dann $\mathtt{ID}_B$.

Lookup (Versuch 1)

Satz

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

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



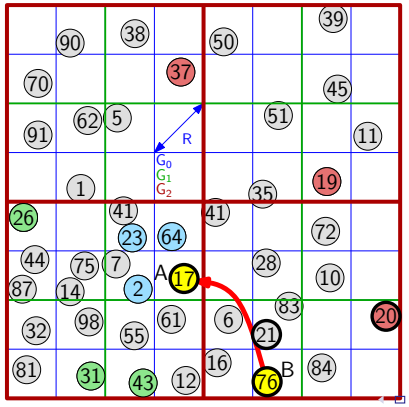
GLS-lookup

Definition

Seien A,B beliebige Knoten. Dann ist B_i^A der Knoten in der G_i-Zelle von B, der $\mathtt{ID}_{B_i^A} - \mathtt{ID}_A$ minimiert.

Beobachtung

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



Location Services GLS MLS

GLS-lookup

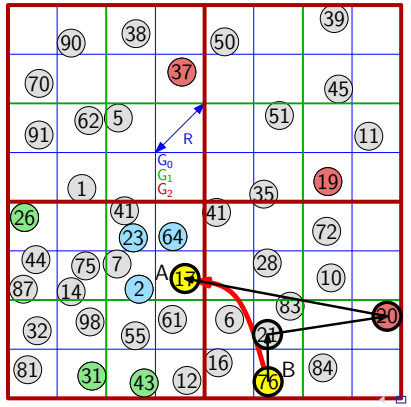
Definition

Seien A,B beliebige Knoten. Dann ist B_i^A der Knoten in der G_i-Zelle von B, der $\mathtt{ID}_{B_i^A} - \mathtt{ID}_A$ minimiert.

Lemma

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

- $\gg$ B kennt komplette G_0-Zelle.
- $\gg$ $\mathtt{ID}_{B_i^A}$ ist in G_i-Zelle von B am wenigsten größer als $\mathtt{ID}_A$
- $\Rightarrow$ B_i^A ist Server für alle Knoten in Nachbarzellen zwischen A und B_{i+1}^A, auch B_{i+1}^A!



GLS-lookup

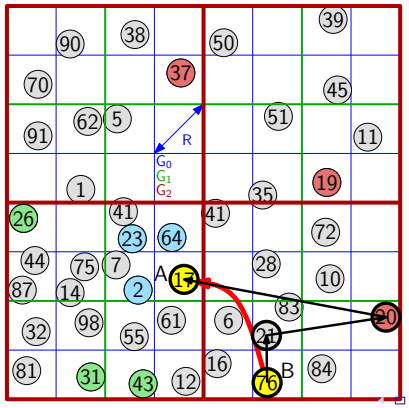
Definition

Seien A,B beliebige Knoten. Dann ist B_i^A der Knoten in der G_i-Zelle von B, der $\mathtt{ID}_{B_i^A} - \mathtt{ID}_A$ minimiert.

GLS-lookup

Auf Suche nach Knoten A schickt B die Anfrage an $B_0^A, B_1^A \ldots, A$.

- $\gg$ Weglänge?
- $\gg$ B B_0^A: maximal R.
- $\gg$ B_i^A zu B_{i+1}^A: maximal $2^{i+1}R$.
- $\gg$ insgesamt Summe der „Luftlinien“ $\sum_{i=0}^{k} 2^i \in O(2^i)$
- $\gg$ 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 bei z. B. bei GOAFR?

- $\gg$ das Paket
  - $\gg$ umrundet die Facette, die die Zielkoordinate einschließt
  - $\gg$ lernt es alle Knoten der Facette kennen
  - $\gg$ in Gabriel Graph: sieht dichtesten Knoten zur Zielkoordinate

Lemma

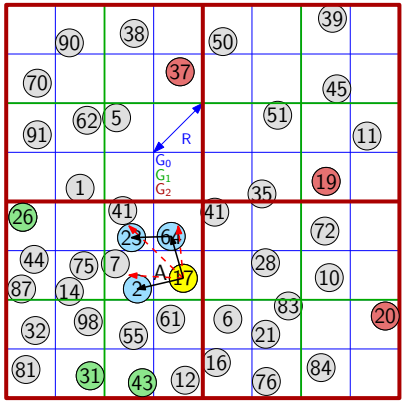
Wird ein Paket zur Mitte einer Gitterzelle C geroutet, in der ein Knoten liegt, dann erreicht das Paket mindestens einen Nachbarn eines solchen Knotens.

(Zumindest in $1/\sqrt{2}$-QUDG, Ohne Beweis)

Initialisierung

Woher weiß A, 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

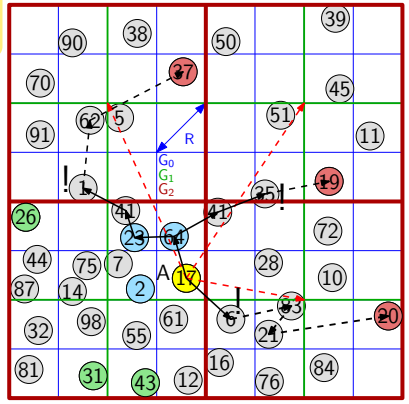


Location Services GLS MLS

Initialisierung

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

- Ebene für Ebene!
- wenn G_i-Zellen-Server bekannt sind
- route künstliches lookup zu irgendeinem Knoten B in entsprechende G_{i+1}-Zellen.
- der reicht die Anfrage bis B^A_{i+1}
- dieser Knoten wird dann Server (Position aus Paket)



Bastian Katz – Algorithmen für Ad-hoc- und Sensornetze
Lehrstuhl für Algorithmik I
Institut für theoretische Informatik
Universität Karlsruhe (TH)
Karlsruher Institut für Technologie
18/ 31
Location Services GLS MLS

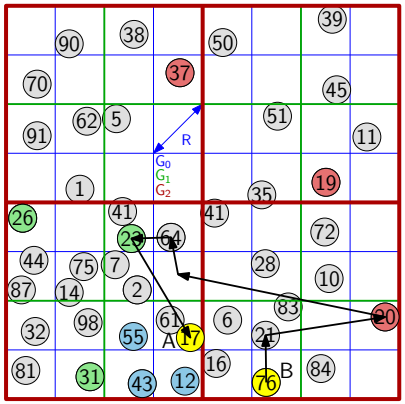
Lazy Publishing

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 $O(\log n)$

Bastian Katz – Algorithmen für Ad-hoc- und Sensornetze
Lehrstuhl für Algorithmik I
Institut für theoretische Informatik
Universität Karlsruhe (TH)
Karlsruher Institut für Technologie
20/ 31

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 um Koordinaten nie „verwaisen“, dass z. B. immer ein Knoten in Abstand $R_{\min}/3$ bei 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

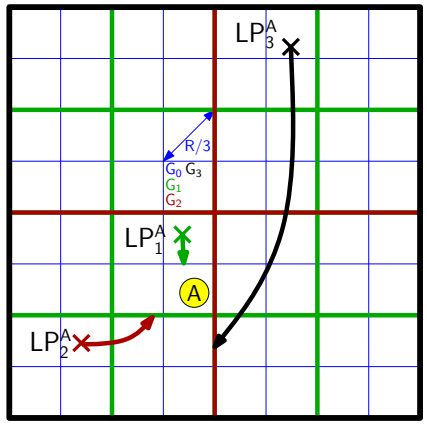
MLS

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

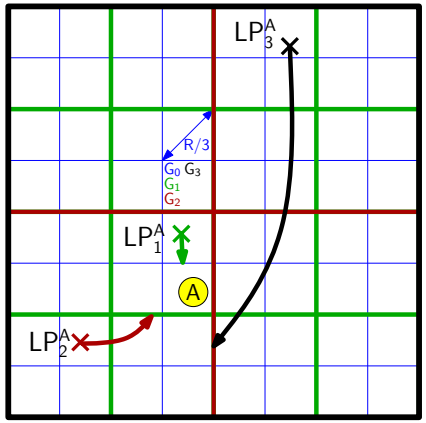


MLS

publish

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

- ≫ das ist ganz einfach



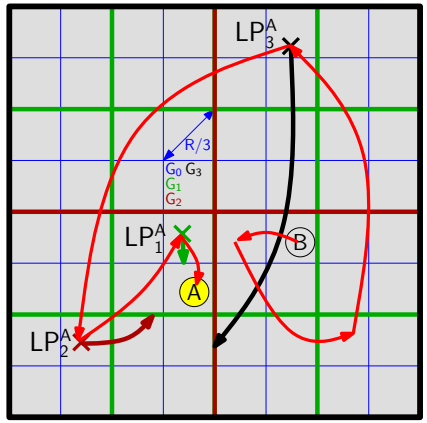
MLS

lookup

In welchen Gitterzellen sollte man Pointer suchen?

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

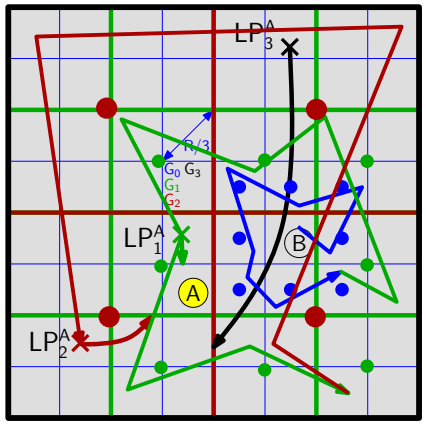
Wie kann man das verhindern?



Suchspirale

MLS-lookup

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

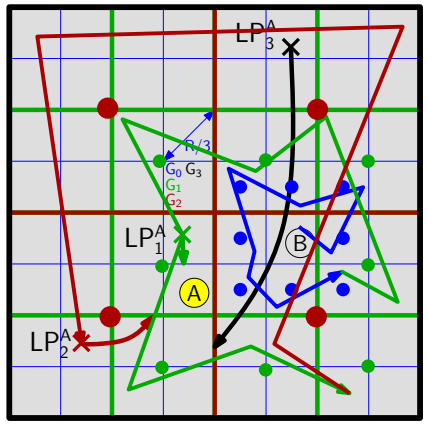


Suchspirale

Lemma

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

- Folgt aus Größe der Gitterzellen!

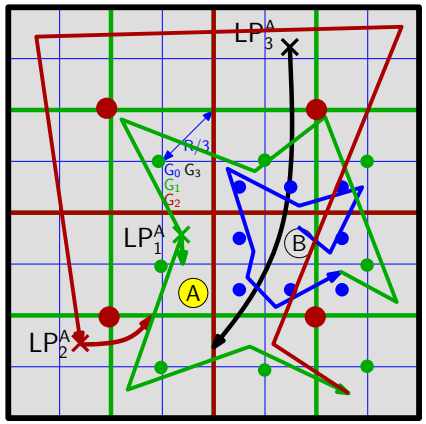


Suchspirale

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
- $\sum_{j=0}^{i} 2^j R \leq 16 \cdot 2^i R = 2^{i+4}$.

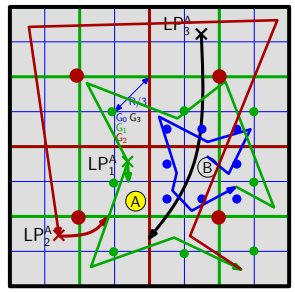


Suchspirale

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 $O(d)$

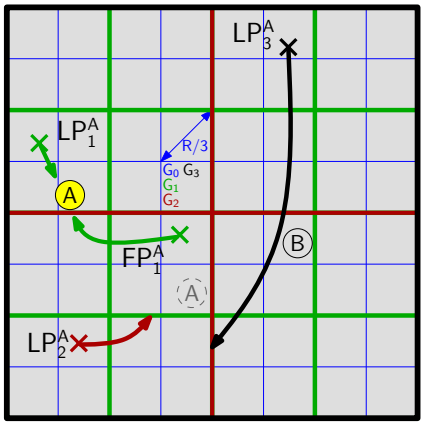


Mobilität & Forwarding (vereinfacht)

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?

- Vorlesungsausfall am 27. 5., Verlegung vom 24. 6.
- Location Services: Proaktives Georouting mit IDs
  - Anforderungen. Lastverteilung, Robustheit, Verhältnismäßigkeit
- GLS: Hierarchische Server, individuell für jeden Knoten
  - Stärken: Gute Aufgabenverteilung
  - Schwäche: Bewegungen/Routen über Gittergrenzen unverhältnismäßig teuer
- MLS: Server*positionen* statt Knoten
  - Stärke: beweisbare Schranken in Bewegung/Entferung
  - 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: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing