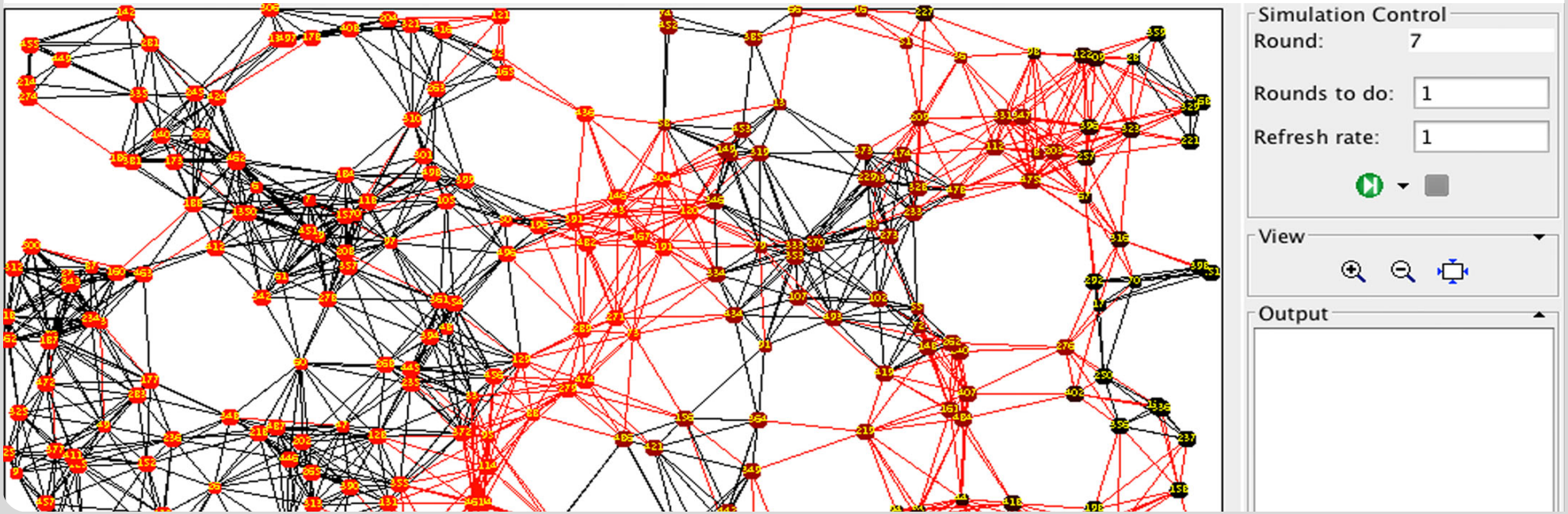
Karlsruher Institut für Technologie

Algorithmen für Ad-hoc- und Sensornetze

Übung 2 – Greedy Routing

Fabian Fuchs | 05. November 2015 (Version 1)

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



Überblick

- Organisatorisches
- Implementierung der Algorithmen
- Besprechung: Leader Election in allgemeinen Graphen
- Greedy Routing

Organisatorisches

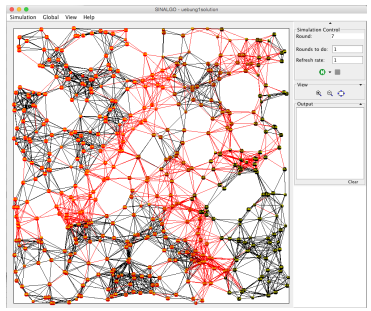
- Übung
  - Es sollen ausgewählte Algorithmen im Netzwerksimulator Sinalgo implementiert werden
  - Von den folgenden Übungsblättern soll mindestens eins in der Übung (teilweise) vorgestellt werden um zur Prüfung zugelassen zu werden.
  - Übungsblätter auch digital unter: `http://i11www.iti.kit.edu/teaching/winter2015/sensornetze/index`
- Bei Fragen: Email oder Sprechstunde
  - Dienstag: 13:00-14:00 (kurze Anmeldung erwünscht!)
  - Oder nach Vereinbarung: `fabian.fuchs@kit.edu`
  - Raum 317, Gebäude 50.34

Überblick

- Organisatorisches
- Implementierung der Algorithmen
- Besprechung: Leader Election in allgemeinen Graphen
- Greedy Routing

Verwendete Frameworks

- Sinalgo: Simulator für Netzwerkalgorithmen in (Drahtlos-)Netzwerken
  - Implementiert in Java, dadurch plattformübergreifend nutzbar
  - Modularer Aufbau ermöglicht parallele Projekte
  - Nachteil: Implementierung grundlegend anders als auf Sensorknoten

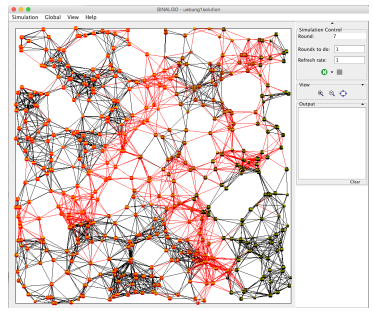


- Arduino Uno: Hands-on Sensorknoten
  - Komplexere (bzw. oftmals ungewohnte) Umgebung, daher mehr Focus auf Kennenlernen der Plattform, weniger auf Algorithmen.
  - Implementierung in leicht erweitertem C

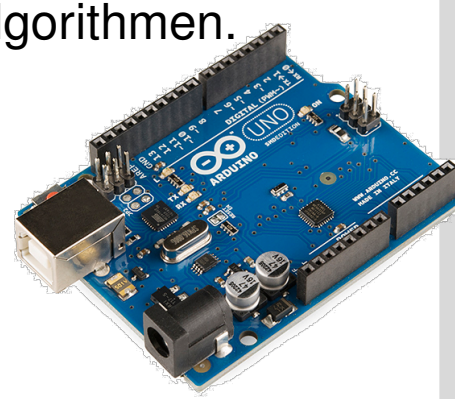
- Aufteilung: 5 Blätter zu Sinalgo, ein Blatt zu Arduino

Verwendete Frameworks

- Sinalgo: Simulator für Netzwerkalgorithmen in (Drahtlos-)Netzwerken
  - Implementiert in Java, dadurch plattformübergreifend nutzbar
  - Modularer Aufbau ermöglicht parallele Projekte
  - Nachteil: Implementierung grundlegend anders als auf Sensorknoten



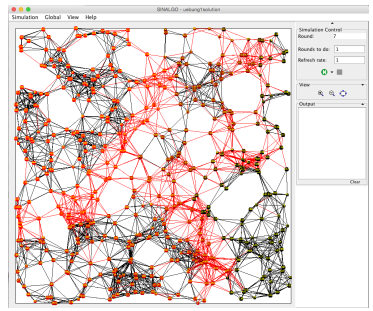
- Arduino Uno: Hands-on Sensorknoten
  - Komplexere (bzw. oftmals ungewohnte) Umgebung, daher mehr Focus auf Kennenlernen der Plattform, weniger auf Algorithmen.
  - Implementierung in leicht erweitertem C



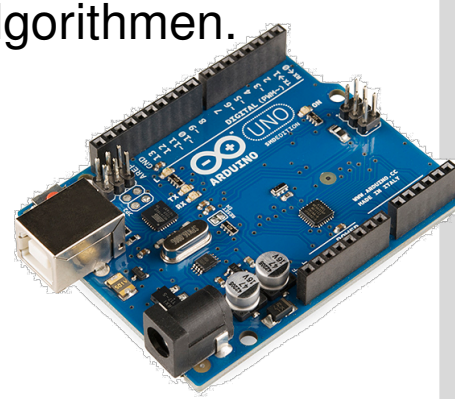
- Aufteilung: 5 Blätter zu Sinalgo, ein Blatt zu Arduino

Verwendete Frameworks

- Sinalgo: Simulator für Netzwerkalgorithmen in (Drahtlos-)Netzwerken
  - Implementiert in Java, dadurch plattformübergreifend nutzbar
  - Modularer Aufbau ermöglicht parallele Projekte
  - Nachteil: Implementierung grundlegend anders als auf Sensorknoten



- Arduino Uno: Hands-on Sensorknoten
  - Komplexere (bzw. oftmals ungewohnte) Umgebung, daher mehr Focus auf Kennenlernen der Plattform, weniger auf Algorithmen.
  - Implementierung in leicht erweitertem C



- Aufteilung: 5 Blätter zu Sinalgo, ein Blatt zu Arduino

Überblick

- Organisatorisches
- Implementierung der Algorithmen
- Besprechung: Leader Election in allgemeinen Graphen
- Greedy Routing

Leader Election

Leader Election, Knoten p_i kennt $\texttt{ID}_i$

sende $\texttt{ID}_i$ an alle Nachbarn
setze $\texttt{ID}_+ \leftarrow \texttt{ID}_i$ und $\texttt{parent} \leftarrow \perp$
wenn $\texttt{ID}$*s empfangen wurden* **dann**
  wähle maximale empfangene $\texttt{ID}$ und zugehörigen Sender s
  wenn $\texttt{ID} > \texttt{ID}_+$ **dann**
    setze $\texttt{ID}_+ \leftarrow \texttt{ID}$ und $\texttt{parent} \leftarrow s$
    sende Nachricht „new follower“ an $\texttt{parent}$
    sende $\texttt{ID}_+$ an restliche Nachbarn
wenn *„new follower“ empfangen wurde und* $\texttt{parent} \neq \perp$ **dann**
  sende Nachricht „new follower“ an $\texttt{parent}$[a]

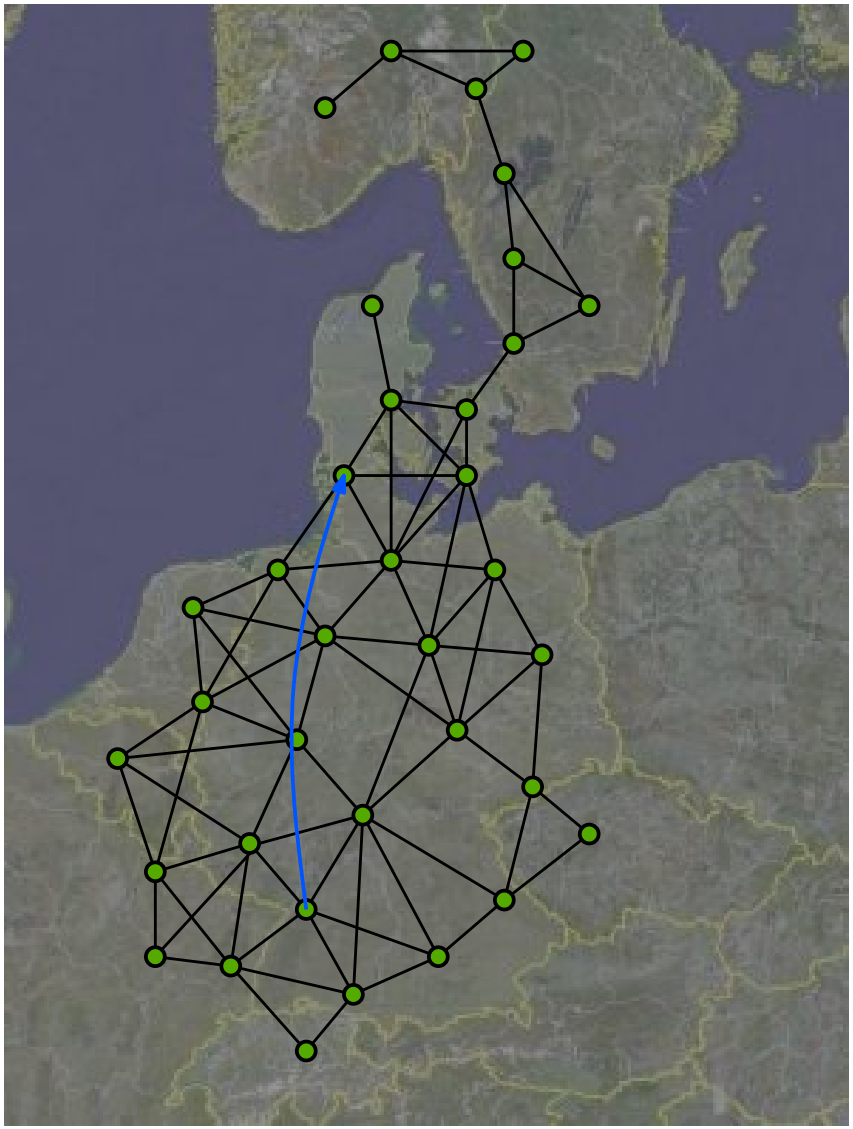
[a] außer in selber Runde schon geschehen

Ein Knoten mit parent = $\perp$ darf sich zum Leader erklären nachdem zwei Runden keine höhere ID und keine new follower-Nachricht kam.

Überblick

- Organisatorisches
- Implementierung der Algorithmen
- Besprechung: Leader Election in allgemeinen Graphen
- Greedy Routing

Greedy Routing



Geographisches Routing: Problem

Gegeben: Eingebetteter Unit-Disk-Graph $G = (V, E)$, zwei Knoten $s, t \in V$.

Gesucht: Positionsbewusste Routingstrategie, um ein Paket, das die Position $\mathbf{p}(t)$ enthält, von s nach t zu bringen.

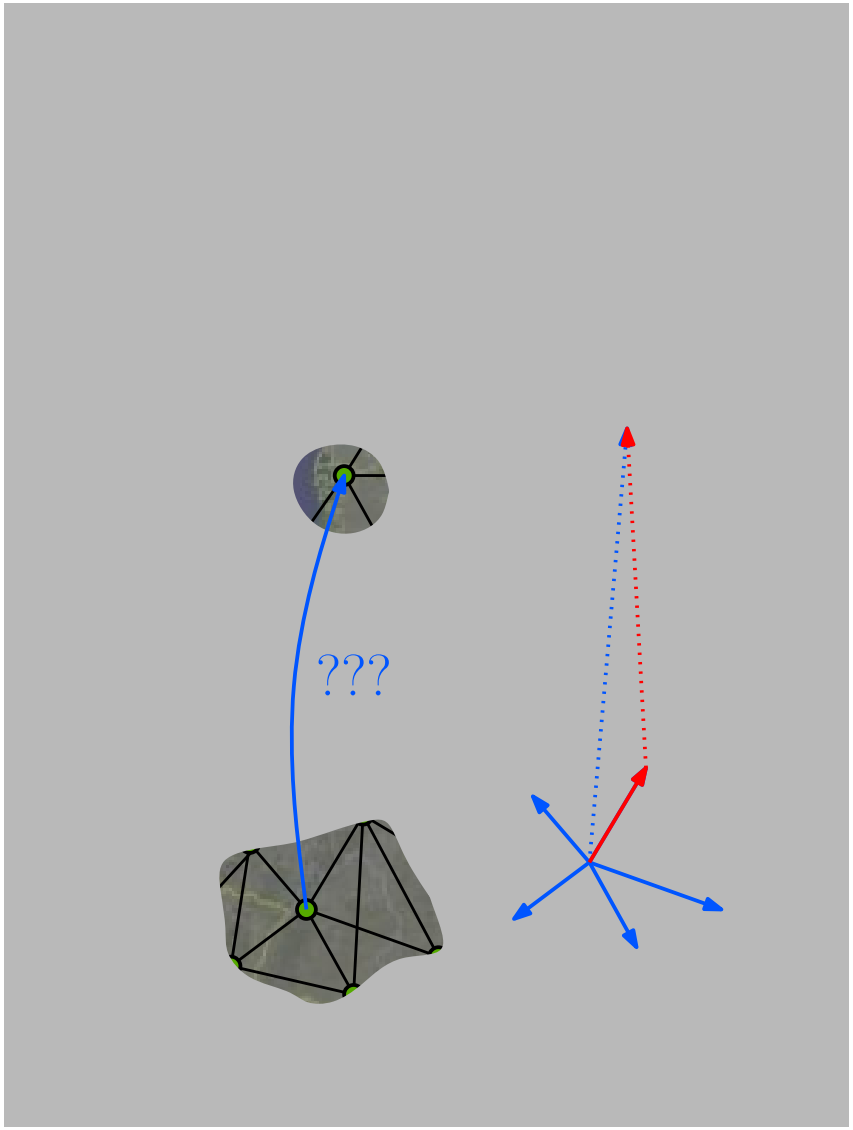
- Es ist erlaubt, „etwas“ Routinginformationen im Paket zu speichern (neben der Zielposition)
- Hier: Anstatt Positionsbewusstsein anzunehmen werden initial die Positionen unter Nachbarn ausgetauscht.

Greedy Routing

Greedy Routing

Leite Paket an den Nachbarn, der dem Ziel am nächsten ist.

Was, wenn es nicht weitergeht?

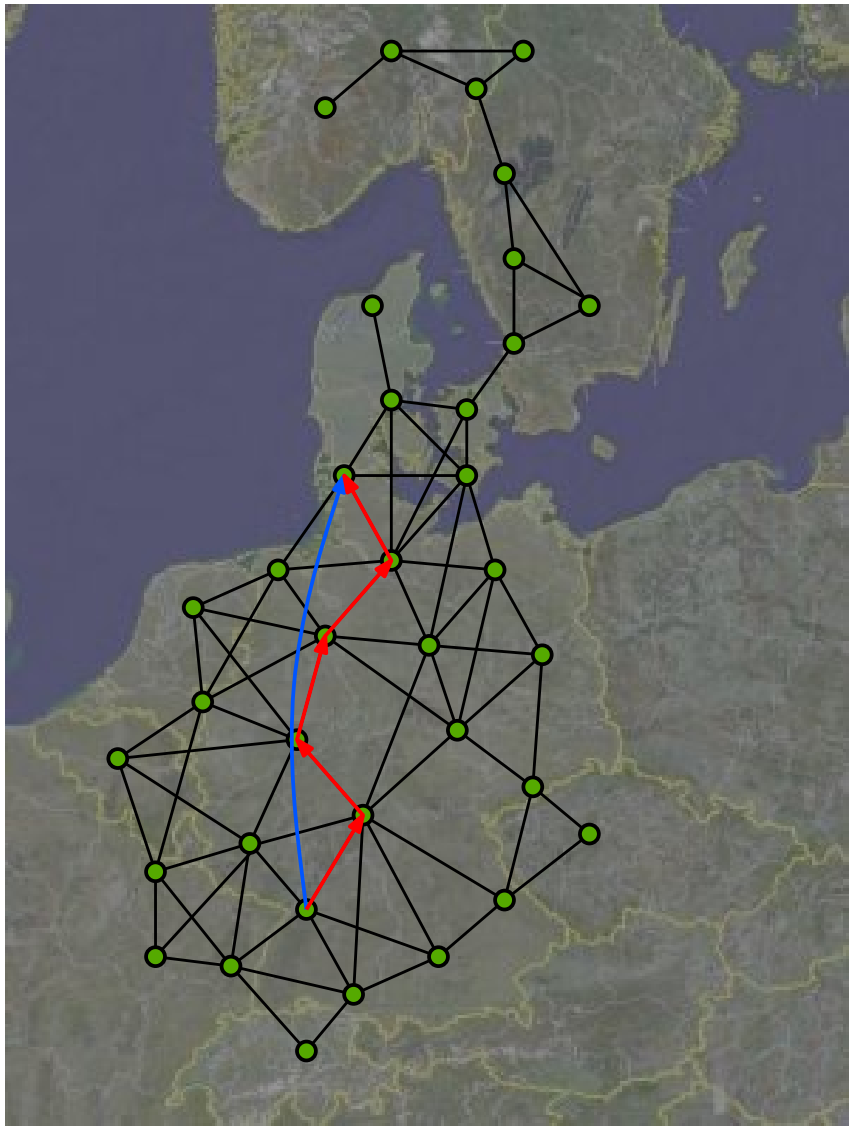


Greedy Routing

Greedy Routing

Leite Paket an den Nachbarn, der dem Ziel am nächsten ist.

Was, wenn es nicht weitergeht?

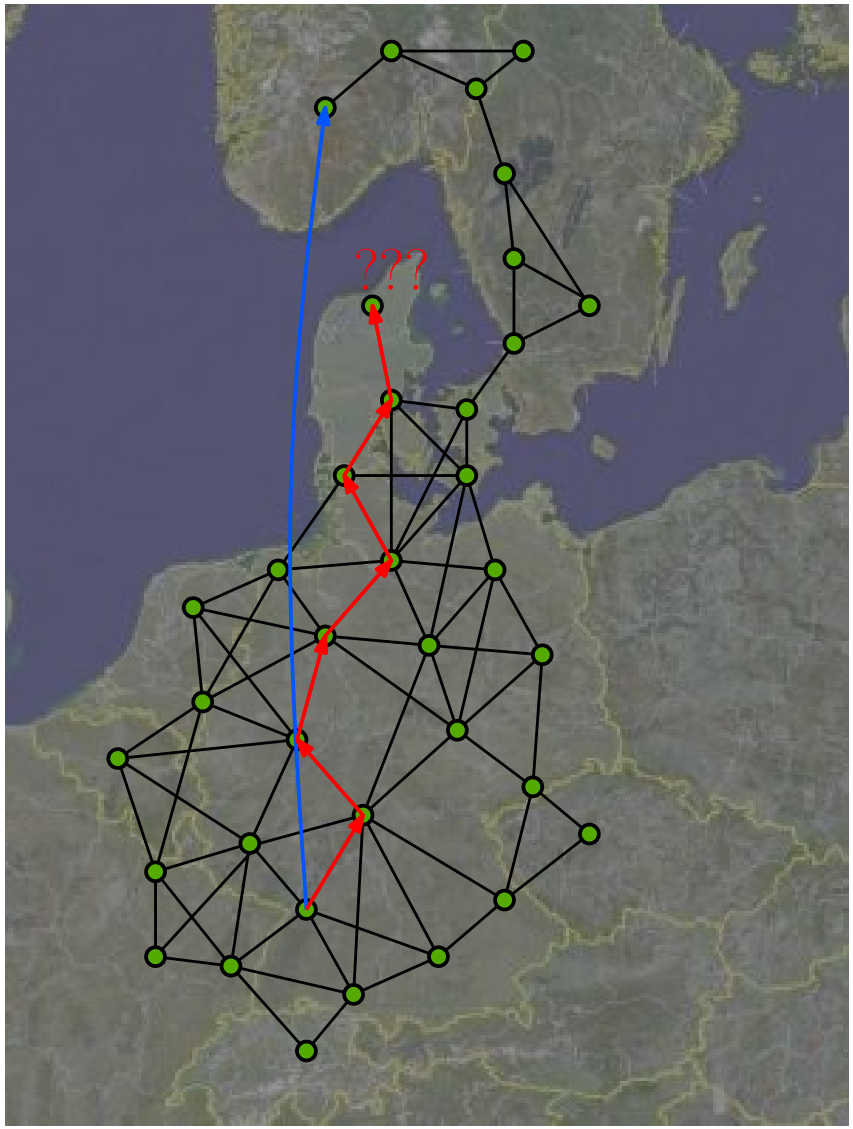


Greedy Routing

Greedy Routing

Leite Paket an den Nachbarn, der dem Ziel am nächsten ist.

Was, wenn es nicht weitergeht?



Greedy Routing

Greedy Routing

- Knoten kennen ihre eigenen Geokoordinaten sowie Zielkoordinaten für Pakete
- Pakete enthalten mindestens die Zielkoordinaten
- Pakete werden immer an den “besten” Nachbarn weitergeleitet

Modell

- Synchrone Runden, synchroner Start
- Pro Runde kann jeder Knoten eine (potentiell unterschiedliche) Nachricht an jeden Nachbarn senden
- Verteilung: Random, Grid, Manuell, ...
- Weiteres: UGD, keine Mobilität, keine Interference, reliable delivery, . . .

Abschluss

Wir hatten heute:

- Nachtrag Location Service: MLS (VL 03)
- Wdh: Organisatorisches zur Übung
- Besprechung Leader Election
- Greedy Routing

Weitere Fragen?

Links zur Übung

- Übungsblatt auf VL-Homepage:
  `http://i11www.iti.uni-karlsruhe.de/teaching/winter2015/sensornetze/index`
- Sinalgo Tutorial: `http://disco.ethz.ch/projects/sinalgo/tutorial/Documentation.html`
- Sinalgo Framework: VL-Homepage oder Sinalgo Tutorial Seite
- Grundgerüst für Leader Election Projekt: VL-Homepage
- Tipp: Wer schon mit Arduino rumspielen möchte:
  https://123d.circuits.io/