Seminar *Algorithmen für Ad-hoc und Sensornetze*

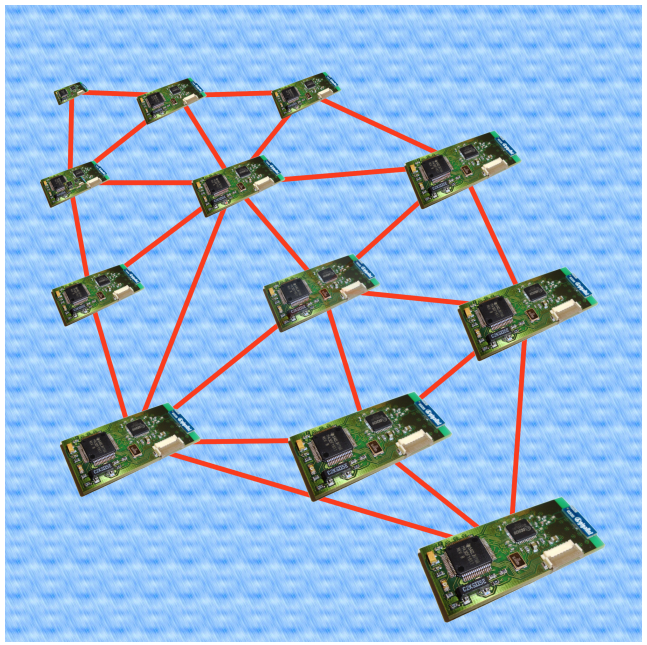
Einführung in Sensornetzwerke

23. April 2007

Inhalt — Einleitung

Motivation

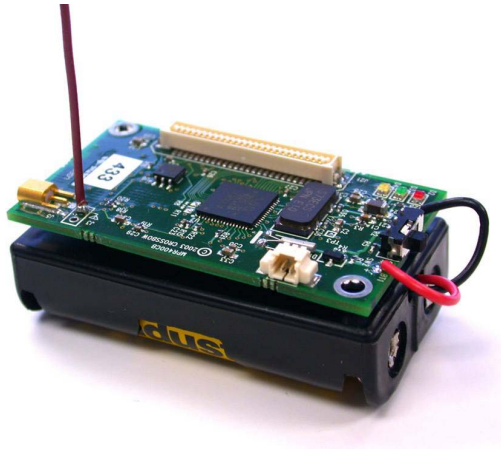
- Computer lassen sich immer kleiner herstellen
  - Moore's Law in die andere Richtung: kleiner bei gleicher Leistung
  - Kommunikation von Bluetooth bis ZigBee
- Phänomene lassen sich beobachten, indem man sie von Sensorknoten durchsetzt
- Idealvorstellung:
  - Preise bis 5$ pro Stück
  - Knoten werden ausgestreut/ausgebracht
  - Knoten organisieren sich selbst
  - Knoten führen Messungen durch
  - beantworten Anfragen kollektiv
  - detektieren Ereignisse, melden sie



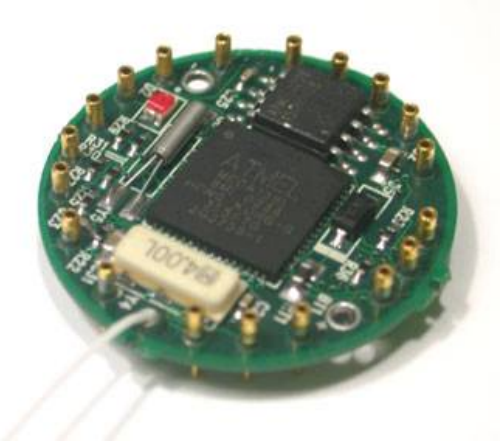
Sensorknoten – eine Auswahl

berkeley mote

micaZ



mica2



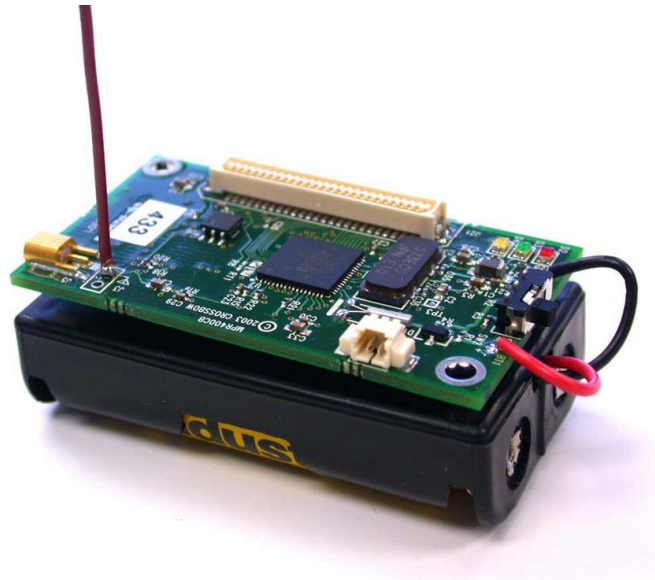
mica2-dot

sun-spot

tmote-sky

Bestandteile

- Vollwertiger Rechner
  - Microcontroller
  - lokaler Speicher
  - Kommunikation
  - Energieversorgung
  - Peripherie (Sensorik, Aktorik)
- Entwurfsziele
  - Miniaturisierung
  - Langlebigkeit
  - niedriger Preis



Begrenzte lokale Ressourcen

- leistungsschwache Prozessoren
- sehr begrenzter Speicher (KBs statt GB)
- hohe Ausfallwahrscheinlichkeit von Knoten
- kein zentrales Lösen von Problemen möglich
- „Masse statt Klasse“, Tausende von Knoten?

Kommunikation

- Knoten kommunizieren per Funk
- Reichweite zwischen 10 und 500 Metern
  - notwendigerweise auch *multi-hop*
- gemeinsames Medium, gegenseitige Störung
  - oft allerdings mehrere Kanäle
- unzuverlässig, keine aufwendigen Protokolle
- geringe Bandbreite, Daten müssen im Netz vorverarbeitet werden

Energieversorgung

- Sensornetze sollen Verkabelung einsparen
  - selten: Strom steht trotzdem zur Verfügung
  - ideal: Knoten „gewinnen“ Strom (*energy harvesting*)
    - Solarzellen, mechanische Bewegung
  - meist: Batterien/Akkus, oft größter Teil
- alles kostet Energie
  - Senden/Empfangen von Nachrichten
  - Berechnungen
  - Warten auf Nachrichten (Empfangsbereitschaft)
  - aktive Sensoren
- Aber: Knoten sollen jahrelang arbeiten
  - Schlafzyklen
  - geschickte Auswahl von Infrastruktur

Sensorik (Aktorik)

- Sensorik
  - Temperatur, Feuchtigkeit, Licht, Druck
  - Lage, Beschleunigung
  - Gaskonzentrationen
  - modular, problemspezifisch
- (Aktorik)
  - Anzeigen (LEDs, Summer)
  - Magnetventile, Pumpen
  - Impulsgeber
  - Motoren (auch zur Fortbewegung)
  - ebenfalls sehr problemspezifisch

Ad-hoc- und Sensornetzwerke

- Gemeinsamkeiten
  - Verteiltes System aus Sensorknoten
  - Drahtlose Kommunkation
- Viele problemspezifische Varianten
  - Art der Knoten (z.B. homogen/heterogen, anonym)
  - Kommunikation (single-/multiple-channel)
  - Verteilung, Mobilität
  - Ausrüstung (GPS, Sensorik)
  - Infrastruktur (Datensenken, Ankerknoten)

Anwendungen

- Überwachung
  - Deiche, Gletscher, Brücken, Meere (Strukturen, Bewegung)
  - Anbaugebiete (Aufzeichnung der Bedingungen)
  - Tiere, Kinder (habitat monitoring, smart kindergarten)
  - Gewässer (Schadstoffkonzentrationen)
  - Waldgebiete (Brände)
- Erkundung unzugänglicher Gebiete
  - Unterseeisch, extraterrestrisch
  - Aufklärung (militärisch)
- Intelligente Umgebungen
  - Überprüfung von Lagerbedingungen
  - automatisierte Logistik
  - Smart-Home-Anwendungen
  - Health-Care-Anwendungen
  - Car/Car-Kommunikation

Inhalt — Modellierung

Algorithmische Modellierung

- Abstraktionslevel
  - Verbergen von technischen Details
  - Abbilden aller relevanten Eigenschaften
- Kommunikationsnetzwerk als Graph
  - Knoten sind Sensorknoten
  - Kanten sind Verbindungen (symmetrisch?)
- Verteilte Algorithmen
  - derselbe Code auf allen Knoten
  - Kommunikation nur mit benachbarten Knoten

Algorithmenanalyse

- Worst-Case-Analyse
  - benötigt sehr einfache Modellierung von Sensornetzen
- Average-Case-Analyse
  - oft durch Simulation
  - basiert auf statistischen Modellen
- Wenig Vorhersage für echte Sensornetze
  - teilweise starke Vereinfachung
  - „relistische“ Szenarien kaum abzusehen

Sensornetze als Verteilte Systeme

Verteiltes System

Ein verteiltes System ist eine Menge von selbständigen Recheneinheiten, die miteinander kommunizieren können.

- Beispiele: Multiprozessorrechner, Internet, Sensornetzwerke
- shared memory $\leftrightarrow$ nachrichtenbasiert (*message passing*)
- synchron: Runden aus Kommunikation und Berechnung auf allen Knoten
- asynchron: (fast) beliebige Folge von Kommunikation und Berechnungen
  - Kommunikation kann sich verzögern
  - Knoten berechnen nicht „im Takt“
- Sensornetzwerke sind asynchron und nachrichtenbasiert
  - Asynchronität zum Teil vernachlässigbar
  - Nachrichtenaustausch als *broadcast*

Komplexität von Algorithmen

- Zeitkomplexität
  - Terminierung: Jeder Knoten kennt „sein“ Ergebnis
  - synchron: Zahl der Runden
  - asynchron: Normalisiere Zeit an längster Verzögerung, maximiere Ausführungszeit über alle möglichen Ausführungen
- Nachrichtenkomplexität
  - Anzahl der (einzelnen) Nachrichten während der Ausführung
  - Kein wesentlicher Unterschied zwischen synchron und asynchron
- besondere Modelle/Klassen von Algorithmen
  - *k-lokal* sind Algorithmen, die in k Runden terminieren
  - *k-lokalisiert* sind Algorithmen, in der jeder Knoten maximal k mal sendet

Uniformität, Anonymität

- Uniformität:
  - Knotenzahl ist den einzelnen Knoten unbekannt
- Anonymität:
  - Knoten sind ununterscheidbar (keine IDs)
  - Knoten führen identische Programme aus

Leader Election im Ring

- Ziel: Bestimmung genau *eines* Knotens
- Unmöglich für anonyme Knoten (Induktion über Schrittzahl):
  - Jeder Knoten tut & empfängt dasselbe
- Uniformer nichtanonymer Algorithmus
  - sende ID an rechten Nachbarn
  - bei Empfang größerer ID leite ID nach links weiter
  - bei Empfang eigener ID sende „terminiere“ nach links, terminiere als Leader
  - bei Empfang von „terminiere“ leite Nachricht nach links weiter, terminiere als Non-Leader

Leader Election im Ring

- Ziel: Bestimmung genau *eines* Knotens
- Unmöglich für anonyme Knoten (Induktion über Schrittzahl):
  - Jeder Knoten tut & empfängt dasselbe
- Uniformer nichtanonymer Algorithmus
  - sende ID an rechten Nachbarn
  - bei Empfang größerer ID leite ID nach links weiter
  - bei Empfang eigener ID sende „terminiere“ nach links, terminiere als Leader
  - bei Empfang von „terminiere“ leite Nachricht nach links weiter, terminiere als Non-Leader
- Laufzeitkomplexität: $\mathcal{O}(n)$ Runden
- Messagekomplexität: $\mathcal{O}(n^2)$
- weder lokal noch lokalisiert

Radiomodelle

- Vernetzung in SN weder zufällig noch gewählt
  - Knotenpositionen und Terrain bestimmen, welche Knoten miteinander kommunizieren können
  - nahe Knoten sehen sich, wenn nichts im Weg ist
  - mit zunehmender Entfernung wird Kommunikation unmöglich
- Ermöglicht das besseren Algorithmenentwurf und -analyse?

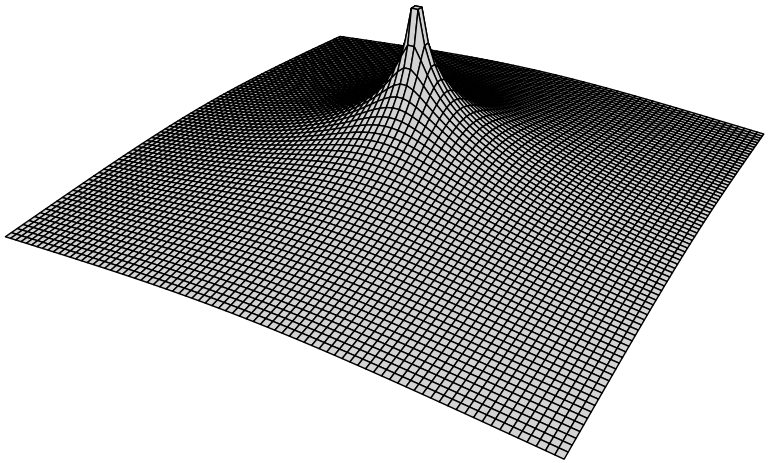
Signalausbreitung

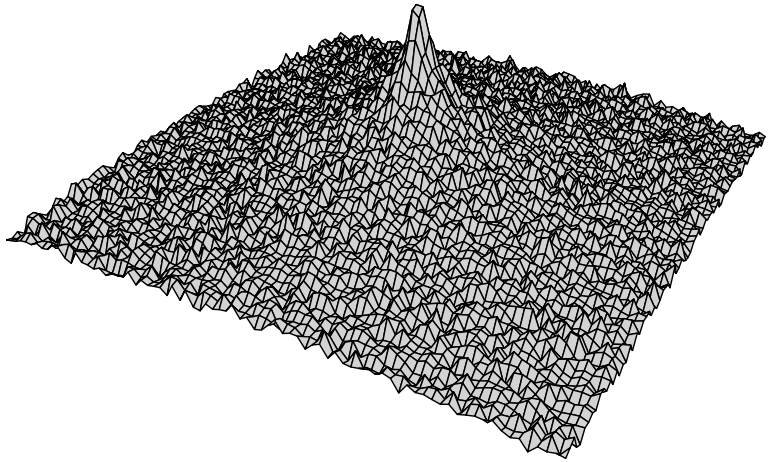
Modell: Exponential Path Loss

Bei einer Sendestärke P_{transmit} (in Watt) ergibt sich die in einer Entfernung d zu messende Signalstärke P_r als

$$P_{\text{receive}}(d) = c \cdot \frac{P_{\text{transmit}}}{d^{\alpha}}$$

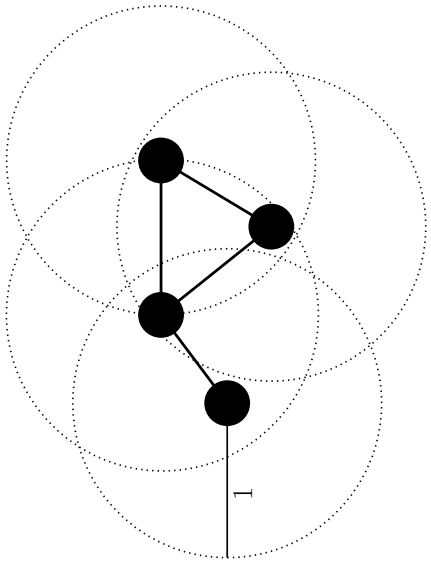
dabei ist α typischerweise zwischen 1.5 und 6, je nach Umgebung.





Unit-Disk-Graphen

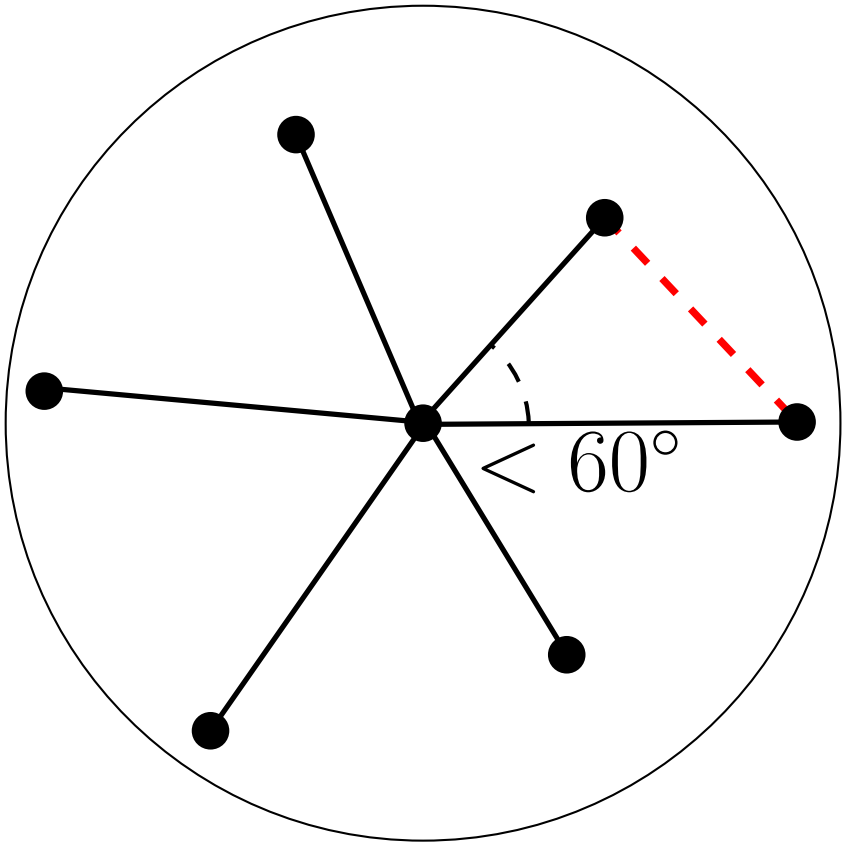
- einfachste Annahmen:
  - einheitliche Sendestärke
  - keine Störungen oder Hindernisse
  - Signal wird empfangen genau für $P_{\text{receive}} \geq \beta$
- $\Rightarrow$ jeder Knoten hört Knoten in festem, einheitlichem Radius
- solche Graphen heißen *Unit-Disk-Graphen*
- interessante Eigenschaften
  - maximal 5 *unabhängige* Nachbarn
- leider sehr idealisierte Modellierung



Nutzen von Unit-Disk-Graphen

Lemma

Ein Unit-Disk-Graph kann keinen $K_{1,6}$ als knoteninduzierten Subgraphen enthalten (Ein Knoten hat maximal 5 unahbängige Nachbarn).



Nutzen von Unit-Disk-Graphen II

Definition

Dominierend ist eine Knotenmenge, zu der jeder Knoten entweder gehört oder adjazent ist. *Unabhängig* ist eine Knotenmenge, die keine zwei verbundenen Knoten enthält.

Theorem

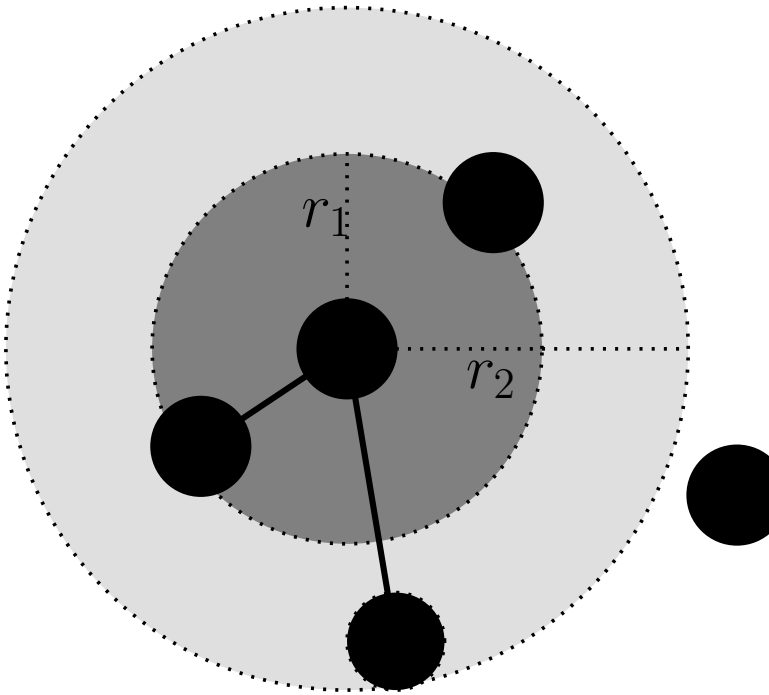
Sei D_{OPT} ein dominierende Knotenmenge minimaler Kardinalität* und I eine inklusionsmaximale unabhängige Knotenmenge**. Dann gilt, daß $I \leq 5|D_{\mathrm{OPT}}|$.

* *NP*-schweres Problem

** sehr leichtes Problem

Quasi-Unit-Disk-Graphen

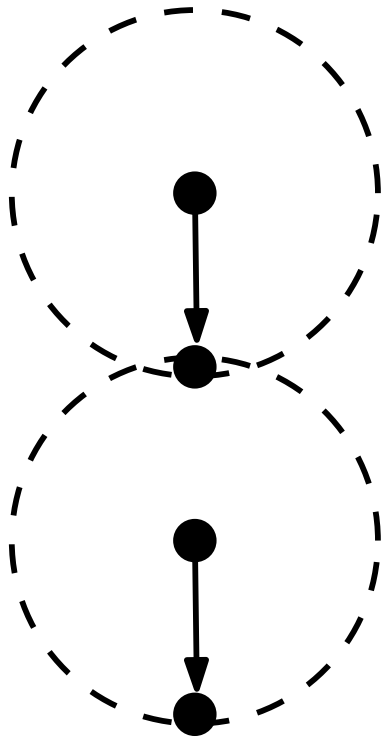
- etwas realistischer: zwei Radien $r_1 \leq r_2$
  - innerhalb von r_1 gesicherte Kommunikation
  - außerhalb von r_2 keine Kommunikation
  - keine Aussage dazwischen
- andere Modelle beschreiben nur Graphstruktur
  - *bounded independence*
  - *doubling metric*



Inhalt — Klassische Algorithmische Probleme

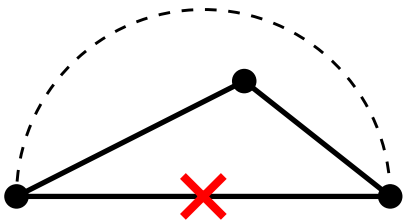
Medium Access Control

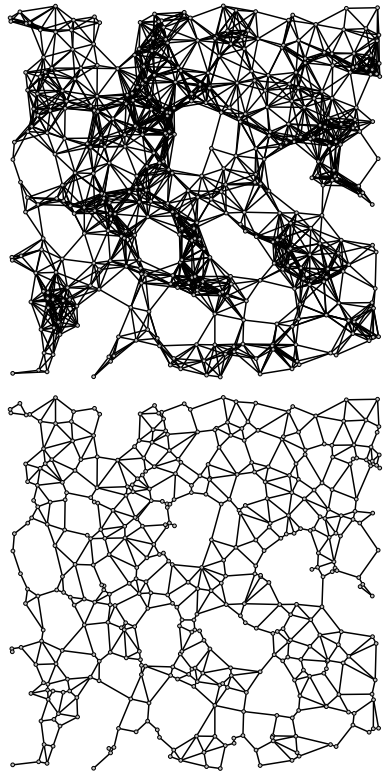
- Alle Knoten nutzen Medium gemeinsam
  - stören sich gegenseitig durch Interferenz
  - Koordination: *Medium Access Control*
- MAC in Sensornetzwerken: CSMA/CA
  - *Carrier Sense Medium Access*:
    - "Mithören", ob Medium frei
    - ggf. Versenden verschieben (wie Ethernet)
  - *Collision Avoidance* statt *Collision Detection*:
    - Kollisionen beim Senden fallen nicht auf!
- weitere Maßnahmen zur Kollisionsvermeidung
  - Verwendung verschiedener Frequenzen
  - Zeitscheibenverfahren
  - erfordert Absprache
  - $\Rightarrow$ algorithmische Probleme



Topologiekontrolle (Power Assignment)

- Viele Sensorknoten erlauben Einstellen der Sendestärke
- geschickte Wahl
  - verringert Interferenzprobleme
  - verringert Knotengrade
  - sichert wichtige Eigenschaften zu
    - Erhalt des Zusammenhang
    - geringe Verlängerung von kürzesten Wegen
- Beispiele: Gabriel-Graph-Ausdünnung
  - enthalten alle energieminimalen Wege





Clustering

- Knoten müssen ihre eigene Infrastruktur aufbauen
  - Arbeitsteilung naheliegend
  - Auswahl von speziellen Knoten, „Clusterheads“
- eine Ausprägung *Connected Dominating Sets*:

Connected Dominating Set

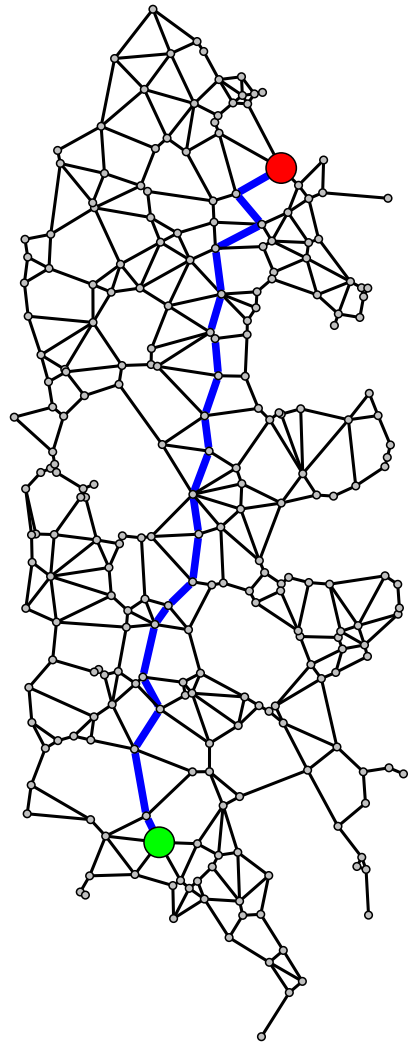
Ein *Connected Dominating Set* eines Graphen (V, E) ist eine Knotenteilmenge $V_{\mathrm{CDS}} \subset V$, so dass

- für jeden Knoten $v \in V$ v oder ein Nachbar von v in V_{CDS} ist
- die Knoten V_{CDS} einen zusammenhängenden Graphen induzieren

- diese Knoten können Nachrichtenverkehr übernehmen

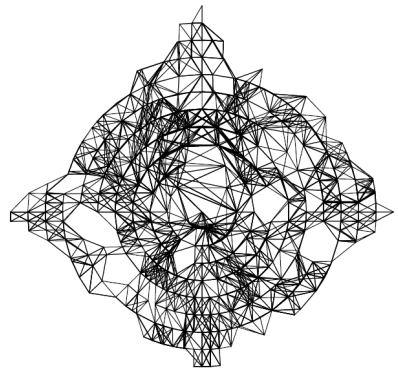
Routing

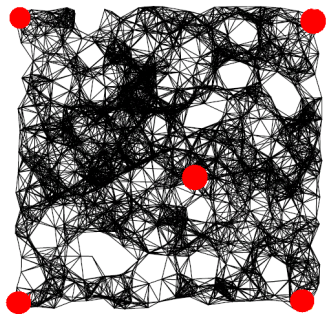
- Knoten wollen gezielt mit anderen Knoten kommunizieren
  - Fluten des Netzes viel zu viel Traffic
  - keine Routingtabellen wie im Internet
- Routingalgorithmus:
  - trifft zu eingehendem Paket Entscheidung, zu welchem Nachbarn es weitergeleitet wird
  - durch Vorberechnung (*compact routing*: $o(n)$ Bits je Knoten)
  - auf Basis von Koordinaten (*geographic routing*)
- Beispiel: Greedy-Georouting
  - „Gib das Paket an den Nachbarn, der am dichtesten am Zielknoten liegt“
  - klappt nicht immer



Positioning

- Knoten kennen ihre Positionen nicht unbedingt
  - GPS oder Infrastruktur aufwendig
- Positionen sind wichtige Informationen
  - Anfragen beziehen sich auf Regionen
  - Ereignisse müssen räumlich zugeordnet werden
  - Algorithmen nutzen oft Geometrie
- zur Verfügung stehen zum Beispiel
  - Kommunikationsnetz und Radiomodell
  - Signalstärkeabfall zur Entfernungsschätzung
  - Laufzeitdifferenzen zu Ultraschallsignalen
  - Richtungsinformationen durch mehrere Antennen
- Rekonstruktion oft schwer





Location-Service-Protokolle

- Knoten können sich bewegen, müssen aber auffindbar bleiben
  - bewegte Knoten können nicht alle ständig benachrichtigen
- Location-Service-Protokoll:
  - schreibt vor, was Knoten tun müssen, wenn sie ihre Positionen ändern
  - lassen Anfragen nach dem Ort eines bestimmten Knotens zu
- strenge Anforderungen
  - gleichmäßige Lastverteilung
  - geringer Overhead

Zusammenfassung

- sehr viele Algorithmische Probleme, nicht angesprochen
  - Anfrage und Aggregation von Daten
  - Synchronisation
  - Deployment/Wake-up
  - Koordination von *node states*
  - Sicherheit
  - ...
- starke Verzahnung
- keine klare Aufteilung in Layer (wie OSI)
- viele Probleme zunächst auf wenige Aspekte reduziert