Drahtlose Sensornetze: Zusammenfassung

Alexander Osiik
alexander.osiik@student.uni-luebeck.de
Drahtlose Sensornetze (SS 2019)
Institute of Computer Engineering, University of Lübeck

31. Januar 2020

1 Einführung

1.1 Sensorknoten?

- sind autonome Miniaturcomputer
- können:
  - über Sensoren Wahrnehmen
  - über Prozessoren verarbeiten
  - über Funk kommunizieren

1.2 Sensornetz?

- drahtloses Netz aus vielen Sensorknoten
- weiträumig, langlebig
- erlaubt detaillierte Umweltbeobachtung

1.3 Anwendungen?

- kann als wissenschaftliches Instrument benutzt werden
  - Beobachtung von Tieren, Pflanzen, Umwelphänomene
- Industrie
  - Kontrolle von Infrastruktureinrichtungen
  - Energiemanagement
- Gesundheitswesen
  - drahtlose Intensivstationene
  - medizinische Forschung
- Militär / Polizei
  - Erkennung “feindlicher” Aktivitäten

1.4 Vorlesung

Es soll Überblick zu Grundlegenden und einigen weiterführenden Aspekten drahtloser Sensornetze verschaffen werden.
Es ist ein multidiziplinäres Gebiet:

- Verteilte Systeme
- Informationssysteme
- Computer Systeme
- Eingebettete Systeme
- Ambient Computing
- Verteilte Algorithmen

1.5 5G

Handy verbindet sich mit Base Station Subsystem, dieses verbindet sich mit dem Network Switching System

- Enhanced Mobile Breitband: Bisher wurden Frequenzen 1-6Gz genutzt. 5G nutzt nun bis 100GHz
- Ultra reliable und geringe Latenzen
- höhere Datenraten und mehr Frequenzen bei verringertem Energieverbrauch

1.5.1 Kanalbündelung und Small Cells

Kanalbündelung ist eine Bündelung der genutzten Funkfrequenzbereiche eines Netzbetreibers (Kanäle in einem Frequenzblock). Dies erlaubt es, Datenrate pro Nutzer zu erhöhen.
Small Cell ist die Mobilfunkzelle mit geringer Sendeleistung, dementsprechend kleinem Versorgungsbereich. Der Radius liegt bei etwa 150m, die Sendeleistung ist gering

1.5.2 MIMO: Massive Multiple Input Multiple Output

Mehrantennensystem, das die zeitliche und räumliche Dimension nutzt.
Durch Space-Time-Coding wird die Zuverlässigkeit und Daterate gestigert (redundante Datenpakete)

1.5.3 Versteigerung

Man erhöht den Kaufpreis und spekuliert auf das Ausscheiden eines weiteren Anbieters. Falls der Anbieter spät ausscheidet, ist die investierte Summe im Endeffekt um ein vielfaches höher

1.6 Betrachtete Netze

die hier betrachteten Netze agieren anders, als z.B. 5G

- Kommunikation geschieht nicht über Handy Netz
- Die Netzstruktur ist dezentral und selbst-organisiert
- hohe Energieeffizienz enorm wichtig für Sensorknoten -¿ Lange Lebenszeit
- Alles für die wissenschaftliche Messung / Überwachung

2 Anwendungen

Viele durch Sensornetze erreichte Optimierungen für das alltägliche Leben denkbar:

- Neues Wissen ermöglichen
- Sicherheit erweitern
- Ressourcenmanagement
- Prävention von Fehlverhalten usw.

In folgende Kategorien kann unterteilt werden

- Wissenschaftliches Instrument ("Macroscope")
  - Tiere, Pflanzen, Umweltphänomene
- Industrielle Anwendungen
  - Infrastruktureinrichtungen (Pipelines, Maschinen)
  - Energiemanagement
- Landwirtschaft
  - Pflanzen (Wachstum, Reife, Bodenqualität)
  - Tiere (Krankheiten, Fruchtbarkeit, virtuelle Zäune)
- Gesundheitswesen ("Body Sensor Networks")
  - drahtlose Intensivstation
  - Verhaltensauffälligkeiten alter Menschen
  - Lifestyle
- Militär / Polizei
  - Erkennung, Klassifizierung, Lokalisierung des "Bösen"

2.1 Mikroklima

Betrachtet wurden Redwoodbäume an der Küste Kaliforniens. Zur Motivation zählte die starke Variation und Dynamik klimatischer Bedingungen im Baum; kleine Wetterfronten bewegen sich entlang des Stamms.

2.1.1 System Überblick

Aufbau

- Sampling alle 5min
- 40-50 Knoten pro Baum
- Multi Hop Netz
- Messung von Temperatur, Feuchte, Sonneneinstrahlung

Knoten: Mote
Netz: TinyDB Sensor Netzwerk, Verbund über TASK Gateway, Funk über GPRS Modem

2.1.2 Fazit

Beobachtung: Trotz geringer Datenrate gab es einen hohen Datenverlust: 60% oder mehr

Ergebnis waren Zeitreihen pro Knoten, die gesamte Lebensdauer betrug 1,5 Monate. Pro Baum wurden 50 Knoten verwendet. Das Netz war **multihop**, **homogen** und **statisch**

2.2 Brutverhalten

Man möchte ein Modell für Brutpräferenzen des Wellenläufers erstellen. Dabei betrachtet man

- Nestbelegung
- Klimatische Bedingungen in Höhlen
- Umweltbedingungen

Die Beobachtung **muss** dezent erfolgen, da eine Abschreckung der Vögel blöd wäre.

2.2.1 Architektur

- Mehrere verbundene Sensor Knoten bilden ein Sensor Patch
- Über Gateway sind diese mit einem Transit Netzwerk verbunden
- über Basestation mit dem Internet, worüber dann abgelesen werden kann
- **Wetter-**(Feuchte, Sonne, Druck) und **Nestsensoren**(Feuchte, Temperatur, Näherung)

2.2.2 Fazit

Die Infrastruktur für Kalibrierung war sehr komplex. Zudem gab es Schäden durch fehlerhafte Verpackungen, da Batterien ausliefen und den Knoten korrodierten. **Datenrate** und **Ergebnis** waren ähnlich wie oben erwähnt, das Netz war diesmal jedoch **heterogen**(Wetter und Nest + Gateways).
Große Herausforderung war die **Kalibrierung** und **Verpackung**.

2.3 Cane Toad Überwachung

Kröten wurden nach Australien importiert, um Schädlinge beim Zuckerrohranbau zu regulieren. Ohne Feinde führte das zu einer starken Ausbreitung. Ein weit verstreutes **Sensornetz** zur Überwachung der Ausbreitung war gewünscht.

- Die Erkennung der Kröten geschah anhand der Quak-Laute: Dauer, Häufigkeit, Frequenz...
- Das akustische Signal wurde aufgezeichnet, mit Forward Fourier Transformation umgewandelt. Daraus bestimmte man dominante Frequenzanteile, die mit C4.5 Algorithmus (**Entscheidungsbaum**) klassifiziert worden sind

Herausforderung: Akustische Aufnahme mit 10KHz, hohe Ressourcenanforderungen durch **Signalverarbeitung** und **Maschinelles Lernen**

2.3.1 Architektur

Zweischichtiges Netz aus **Motes** und **Stargates**:

- Motes nehmen akustische Signale auf, führen Kompression durch (auf ca. 30%), und schicken sie an Stargate per **Round-Robin-Schedule**
- Stargate klassifiziert auf den Signalen, **Voting Process** entscheidet darüber, ob weiter zum Server durchgelassen wird

2.3.2 Fazit

Separat lag die Korrektheit bei 100%, bei 6 Kröten gleichzeitig lag die Implementierung 50% richtig($\approx$Münzwurf). Latenz betrug 45sec, bei Kröten aber verkraftbar.

2.4 Vulkane

Es sollen die vulkanische Aktivität beobachtet werden. Ein Sensornetz soll als Ersatz für bisherige Messstationen in unerschlossenen Gebieten dienen. Messung von:

- Seiesmischen Schwingungen breiten sich über Boden aus (am Fuß)
- Infraschall (<50 Hz) breitet sich über Luft aus (am Schlot) $\implies$ früher!

2.4.1 Architektur

Früher: Große Box, viele Kabel, zentrale Stromverteilung, Single Point of Failure

- Microphone Motes zum EmpfängerMote verbunden (Freewave Modem)
- Knoten samplen und schicken kontinuierlich
- Modem schickt Signale an das 9km entfernte Observatorium

2.4.2 Fazit

Infraschall ist guter Indikator für Eruptionen.
Im Vergleich war die neue Architektur ähnlich gut wie die alte. Es gab eine mittlere Datenrate, Ergebnis waren **Zeitreihen** und **Ereigniserkennung**. Eine Sterntopologie lag vor, heterogenes Netz.

2.5 Gletscher

Man will ein besseres Verständnis der Dynamik im Gletscherinneren und Untergrund erlangen. Es soll en **Bewegungsmodell** für Gletscher entwickelt werden, und die **Auswirkungen der globalen Erwärmung** analysiert werden.

2.5.1 Architektur

Bisher: Bohren und Proben mit Bohrloch-Kameras
Jetzt: Knoten auf verschiedenen Tiefen des Gletschers. Diese senden an eine auf dem Gletscher befindliche Basistation (auf Batterie und Solar), die das Signal dann an die Referenzstation liefert.
Proben

- PIC Mikrokontroller mit PLastikmanter
- Funkmodul für niedrige Frequenzen
- weitere Sensoren (Orientierung, Druck, Temp)
- Referenzstation war Linux Server

2.5.2 Fazit

Resultate waren Zeitreihen pro Knoten und deren Positionen (Verschiebung des Gletschers). Das Netz hatte eine Sterntopologie, war **heterogen** und **mobil**.

2.6 Lokalisierung von Schüssen

Ein Sensornetz zur Lokalisirung von Heckenschützen, Banden- und Straßenkriminalität. Erkannt werden sollte die Druckwelle der Gewehrmündung.
Sensorknoten war MICA2, Berkeley Mode, erweitert um **Mikrofon** und **Verstärker**.

Teilaufgaben

- Erkennung der Druckwelle
- Zeitsynchronisation und Lokalisierung
- Senden an Basisstation, von wo aus die Abschussposition bestimmt wird

Signal wird kodiert. Ob das Muster passt wird anhand eines Zustandsautomaten bestimmt, Zuverlässigkeit 90%.

2.6.1 Positionsbestimmung

Bei Schuss gibt es Nachrichten von vielen Knoten: Position p_i, Zeit t_i. Es gibt aber auch Fehlinformation durch Echos, falsche Erkennung etc.
Herangehensweise war eine a priori gesetzte Einschränkung des Suchraumes.
Konsistenzfunktion: Maß der Übereinstimmung der Hypothese (p, t) mit tatsächlich gemessenen Werten

$$C_\tau(p,t) = \text{COUNT}_i(|t_i(p,t) - t_i| \leq \tau)$$

Die **Suche des Maximums** erfolgte durch eine **Bisektion** des Lösungsraumes.

2.6.2 Fazit

Mittlerer 2D Fehler lag bei 0,57m, 3D Fehler bei 0,98m.
Die Datenrate war hoch, als Resultat hatte man eine Lokalisierung. Das Netz war **homogen** und **multi-hop**. Zeitsynchronisation von vielen Nodes und fehlerbehaftete Messdaten waren eine Herausforderung.

2.7 Funktionalität und Herausforderung

Zur Funktionalität zählen:

- kontinuierliche Datenströme (kein failure)
- Erkennung von Ereignissen
- Klassifizierung von Ereignissen
- Lokalisierung

Als Herausforderung gelten:

- Energieeffizienz
- beschränkte Ressourcen
- Robustheit und Zuverlässigkeit
- Autonomie, Kosten, Kalibrierung, Skalierbarkeit

3 Hardware

3.1 Motes

Entwickelt an der UC Berkeley in verschiedenen Ausführungen.
Vorhanden waren:

- CPU, RAM (1-10KHz)
- Kommunikation mit bis zu 500kbps, 10-1000m
- Energieversorgung durch Batterie oder Energy Harvesting

In der Regel gibt es viele Varianten für jede Komponente. Wichtige Kriterien sind jedoch **Lebensdauer**, **Leistungsfähigkeit**, **Robustheit**, **Größe**, **Kosten**. Dies alles hängt natürlich von der Applikation ab.

3.2 Prozessor

Für Prozessoren gibt es verschieden Alternativen, wie

- Mikrocontroller, I/O Kanäle, Wandler...
- DSP für Signalverarbeitung
- FPGA mit frei programmierbaren logischen Gattern
- ASIC als applikationsspezifischer Chip

3.3 Kommunikation

Generell kann über **Funk**, **Licht**, und **Schall** kommuniziert werden. Wichtig ist die Betrachtung der Eigenschaften:

- welcher Frequenzbereich? Breitband? Mehrere Kanäle?
- Reichweite? Interface? Datenrate?
- Energieverbrauch zum Senden und Lauschen

RFM TR1000 guter Listener, Chipcon CC2420 guter Sender

3.4 Sensoren

Es gibt passive und aktive Sensoren. Aktive Sensoren (Radar, Laser) messen immer **gerichtet**, während passive Sensoren sowohl gerichtet (Licht, Kamera) als auch **omnidirektional** (Temperatur, Feuchtigkeit) sein können.
Des weiteren wird in **analoge** (Spannung) und **digitale** Sensoren unterschieden.
Beispiele:
Beschleunigungssensor (Gyro). Messung der Auslenkung erfolgt piezoelektrisch und kapazitiv.
Kohlendioxid Änderung der Leitfähigkeit durch Freisetzung von Ionen

3.5 Energieversorgung

Hierzu zählen Batterien, Akkumulatoren, Gold-/Supercaps: Kondensatoren mit hoher Kapazität. Anforderung ist dabei eine geringe **Selbstentladung**, **Wiederverwendbarkeit** und **Spannungsstabilität**

3.5.1 Energiegewinnung

Elektrische Energie kann durch Umwandlung anderer Energieformen gewonnen werden.
Beispiele:

- Brennstoffzellen (10-100 mW)
- Solarzellen (10 μW-15mW)
- Vibrationen (0.1-10.000 μW)
- Druck (piezo) (330 μW)
- Radioaktiver Zerfall (150W / g)

Voraussage: Energiedichte wird sich erhöhen, während Kostenreduktion langsam stagniert.

3.5.2 Energieverbrauch

Es gibt einen Trade-Off zwischen **Rechnen** und **Kommunizieren**. 1.000 Recheninstruktion sind in erwa äquivalent zu 1 Byte senden! Es ist also besser, lokal zu Rechnen statt zu kommunizieren. Dennoch: Ein Prozessor hält mit Akku etwa 12 Tage, Flash etwa 3, das ist nicht lange!
Als Lösung wird **Duty-Cycling** vorgeschlagen. Dahinter stehen **lange Schlafphasen**, währen der alle Komponenten ausgeschaltet werden. Voraussetzung ist außerdem **schnelles Aufwachen** und **kurze Wachphasen**. Damit wird die Lebenszeit auf 100Tage verlängert!

- Zeit zum Aufwachen betrug dabei in etwa 1-10ms
- Beispiel: Telos T-Mote, ZigBee kompatibel, Erweiterungsstecker

3.6 Miniaturisierung

Vision “SmartDust”: Sensorknoten im Format eines Staubkorns. Man hat sich für ein 3D Layout entschieden (lieber hochbauen, statt großflächig planar) mit stapelbaren und flexiblen Platinen. Man wünschte sich ein **System-on-Chip**, alle Funktionen in einem Halbleiterchip.
Problem: WiseNet SoC. Energieverbrauch skaliert nicht! Das Funkmodul hat eine relativ hohe Sendeleistung.
Alternativen zum Funk waren Laser:

- große Distanzen überwindbar
- optische Empfänger sehr einfach und sehr klein

- durch Spiegel (Corner-Cube-Retroflector) im Dust wurde passive Laserkommunikation ermöglicht
- Smart Dust hatte aktive und passive Transmitter, Batterie am größten!

Die Basistation hat dann durch einen **defokussierten Laser** (Raster Scanning) Daten gesendet. Empfangen wurden diese durch eine Kamera mit **hoher Bildrate**. Empfang **mehrere Knoten gleichzeitig** war möglich (Knoten per Pixel)

$\Longrightarrow$ Braucht man das alles?

Was bringt die Zukunft? Wird das Moore'sche Gesetz so weiterlaufen?
Effiziente Funkmodule und neue Energiequellen in Entwicklung!

4 Betriebssysteme

Was ist ein Betriebssystem? Es ist eine Abstraktion des Systems, welches Zugriff auf Systemressourcen vereinfacht. Zur Ressourcenverwaltung gehören **Speicher**, **Prozessor**, **Kommunikation** und **Ein-/Ausgabe**.

Das besondere an Sensornetzen ist, dass nur **wenige Ressourcen** bereitstehen, der **Prozessor primitiv** ist und jeder Node ein **spezielles Anwendungsprofil** hat.

4.1 Nebenläufigkeit

Unter Nebenläufigkeit versteht man die quasiparallele Ausführung mehrerer Aktivitäten. Zum Beispiel Lesen und Empfangen.
Jedes BS hat Prozesse und Threads (Prozesse ohne Adressraum). Probleme sind meist der **konfliktbehaftete Speicherzugriff** und der **Overhead für Prozesswechsel**.
Nebenläufigkeit realisiert durch:

- WHILE-Loop
- Events: asynchrone Ereignisse, Run to completion, auch mehrstufig denkbar
- Zustandsmaschinen

4.1.1 Speicherverwaltung

Der Speicher soll unter den Prozessen effizient aufgeteilt werden. Hierbei ist auf Schutz des Speichers zu achten! (**MUTEX, SEMAPHORE**).
Bei drahtlosen Sensornetzen gibt es keine virtuelle Speicherverwaltung, und damit auch kein Schutz! Dies ist jedoch nicht so tragisch, da eigentlich nur ein Prozess ausgeführt wird.
Speicher ist aufgeteilt in **Reserviert — Global — Frei — Stack**
Ein WSN hat keine Festplatten oder Flash, oder ROM, sondern nur einen **Flashspeicher**, typischerweise Random Access

4.2 Kommunikation

Wichtig für WSN ist der Austausch von Nachrichten zwischen Knoten. Dabei möchte man den Overhead minimal halten, und das Kommunikationsmodell ist **Hop-by-Hop**. Es ist lieber, **effizient** zu sein, als zuverlässig.

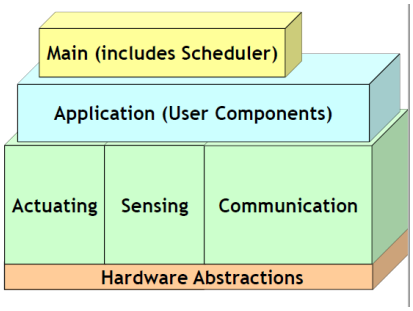
Das Kommunikationsmedium in WSN ist meist Funk. Medienzugriff erfolgt über **MAC**, siehe unbedingt **CSMA/CA**. Das Routing ist oft Baum-basiert (Senke an alle, Knoten an Senke). Die Verwaltung der Nachbarschaft ist schwer, da nicht klar ist, wer der Nachbarknoten ist. Meist wird nach Verbindungsqualität gewertet.

Sonstige Funktionen sind **Hardware-Treiber** (Sensoren, Uhr, I/O), **Energiemanagement** (Sparmodi, DutyCycling), **Programmierung "over the air"**, **Komponenten**system (Ersatz für fehlende Prozesse)

4.3 TinyOS

Populäres Betriebssystem der Berkeley Motes:

- Komponenten System
- Events + Scheduler
- Active Messages ("Remote Events")



Besteht aus **Commands** (=Methoden), **Events** (Ereignisse), **Event-Handler** (kurze Funktion die beim Event abgearbetet wird), und **Task** (längere, asynchrone Funktion)

- **Methode:** Aufruf CALL
- **Event:** (HW) Interrupt: SIGNAL
- **Tasks:** können alles
- **Scheduler:** FIFO
- Split-Phase-Operation: Event/command postet Task; wenn beendet, signalisiere Event

Zu den Komponenten gehören **Interface**, **Modul** (=Klasse), und **Configuration** (Komposition von Modulen, Anwendung). Darüber hinaus gibt es die Möglichkeit, Event-Handler aus der Ferne zu aktivieren (durch **Active Messages**). Eine AM enthält dabei die ID des Handlers und die Nutzdaten.

4.4 Events: Freund oder Feind?

Threads sind in der Regel bequem, aber oft ineffizient.
Events dagegen sind effizienter als Threads, aber oft umständlich.

Die Problematik bei Events ist, dass die Funktion aufgespalten werden muss. Dies führt zu

1. globalen Variablen
2. expliziten Zuständen

Verkettung ist durch Aufruf des Events im Event möglich.

4.5 OSM, andere

5 Vernetzung von Sensoren

Es stellt sich die Frage, wie man einen Sensorknoten zum Netz verbindet.

- **1-Hop:** Jeder Knoten kann mit jedem. Ausdehnung durch **Kommunikationsreichweite beschränkt**
- **Stern:** Alle Knoten mit leistungsfähiger Basisstation (zB SmartDust)
- **Cluster:** Sensorknoten kommunizieren **NUR** mit leistungsfähigem Head, Heads kommunizieren untereinander
- **Multi-Hop, Ad-Hoc:** Knoten kommunizieren mit Nachbarn, und nehmen andere als Knoten Router

Jeder Knoten kann zudem eine gewisse Knotenrolle annehmen. Zur Auswahl stehen dabei **Datenquellen** (Sensor), **Aggregatoren** (empfangen von mehreren Knoten, Reduktion), **Router** (leiten weiter) und **Senken** (Datenbank). Die Zuweisung ist dabei oft natürlich und hängt von der Netzstruktur ab.

5.1 Senken

Es sind Sammelstellen für Daten. Diese Stellen jedoch **Engpässe** dar, da Knoten in der Nähe der Senke viele Daten weiterleiten müssen (zB. EG Auswahl im Fahrstuhl).
Als Lösungsmöglichkeiten könnte man die Verwendung mehrerer Senken vorschlagen, oder eine erhöhte **Aggregation** der Daten im Netz.

5.1.1 Sensor Internet

Senken Könnten auch Gateways zum Internet bilden. Dadurch erhält man ein **globales** Sensornetz!
Hierfür sind aber weitere Schritte notwendig. Die Internet- und Web-Integration stellt eine große Herausforderung dar:

- IP im Sensornetz? Oder soll das Protokoll übersetzt werden?
- IP-Adresse pro Knoten oder pro Netz?
- Wie repräsentiert man es als Web-Service?

5.1.2 Sensor-Aktor-Netze

Bei Robotern oder Lampen gibt es neben Sensorknoten zusätzlich noch einen **Aktorknoten**. Diese werden mittels aus Sensordaten gewonnenen Informationen gesteuert. Hierbei gibt es **keine** Senken.

Im allgemeinen unterscheiden sich Netze aus Sensorknoten gerade dadurch. Bei Sensornetzen stehen die **Messdaten** im Vordergrund, nicht die Knoten. Die **Verarbeitung** der Daten geschieht im Netz, nicht an jedem Knoten selbst. Welcher Knoten Daten liefert ist grundsätzlich egal, da jeder Knoten identisch ist. Hierdurch wird die **Datenbasierte Adressierung** eingeführt: Man spricht mit allen Knoten, auf die eine Eigenschaft/Event zutrifft, nicht deren Adresse.

In WSN werden zudem die Protokollstacks verschmolzen. Dies wird als Hilfsmittel zur Optimierung von Energie- und Ressourcenverbrauch genutzt. Ein **gezielter Austausch** von Informationen wird somit ermöglicht.

6 Modulations Techniken

Unter Modulation versteht man die Modifizierung eines Signals in eine Form, die es möglich macht, dieses Signal zu verschicken. Wird vor allem für langwellige Signale genutzt. Dabei wird das Signal mit einem **sinusförmigen Carrier Signal** verbunden

6.1 Analoge Modulation

6.1.1 Amplitude Modulation

Simpelste Art der Modulation, sehr simpel und kosteneffektiv, jedoch anfällig für Störungen.

$$S_{AM}(t) = [A_c + S_m(t)] \cdot \cos(W_c t)$$

6.1.2 Frequenz Modulation

Frequenz wird moduliert. Schaltung komplexer, aber weniger anfällig.

$$S_{FM}(t) = A_c \cdot \cos([W_c t + S_m(t)]t)$$

6.1.3 Phasen Modulation

Phase wird bei übertritt der x-Achse umgeswitcht.

$$S_{PM}(t) = A_c \cdot \cos(W_c t + S_m(t))$$

6.2 Digitale Modulation

Bei der digitalen Modulation ist der input eine binäre Sequenz, die mit einem sinusoiden Carrier moduliert wird. Dies ist robuster, einfacher zu multiplexen, und sicherer.

6.2.1 Amplitude Shift Key

Eigentlich simples ON-OFF Signal.

6.2.2 Frequency Shift Key

Es gibt 2 vordefinierte Frequenzen für Bit 1 und Bit 0.

6.2.3 Phase Shift Key

Phase des Cariiers wird instantan geshifted.

6.2.4 Quadrature Amplitude Modulation

6.3 Fazit

PSK ist besser, da mehr Energieffizienz und weniger Fehleranfälligkeit. Höherlevelige PSK können die datenrate noch mehr erhöhen.

7 Bitübertragung und Sicherung

Bitübertragung stellt die unterste Schicht des **ISO/OSI Modells** dar. In WSN geschieht die drahtlose Kommunikation meist über Funk: Die Bits werden über Funkwellen übertragen. Es ist wichtig, einige grundlegende Aspekte zu betrachten um sein Verständnis zum Thema Medienzugriff zu festigen.

7.1 Frequenzbereiche

Es gibt unterschiedliche Frequenzbereiche mit unterschiedlichen Eigenschaften, wie **Durchdringung und Dämpfung**, **Reflektion**, **Energieaufwand** etc.

- $<$ 30kHz: VLF
- $<$ 300MHz: Koaxial-Kabel
- 300MHz bis 6GHz: Unser Funk Kommunikationsspektrum
- $>$ 300THz: Licht
- $\Longrightarrow$ Je größer die Frequenz desto geringer die Reichweite/stärker die Dämpfung

Die Frequenzbereiche sind bestimmten Anwendungen zugeordnet, wobei die Verwendung üblicherweise lizensiert ist. Für WSN sind die **lizenzfreien ISM** Bereiche interessant (Industrial, Scientific, Medical).

7.2 Modulation

Modulation ist die Art, wie Bits übertragen werden. Die Übertragung geschiet meist mittels **sinusförmiger EM-Wellen**, welche auf deren Parameter untersucht wird:

- Amplitude
- Phase
- Frequenz

Man unterscheidet dabei in drei Arten des Keying (Auswahl der Modulation aus einem bestimmten Alphabet):

Amplitude Shift Keying: Die Amplitude variiert je nach Bit
Frequency Shift Keying: Die Frequenz variiert (1 Hoch, 0 Tief)
Phase Shift Keying: Die Phase ändert sich bei Flanken

Als Demodulation bezeichnet man nun die Rekonstruktion der übertragenen Informationen aus der empfangenen EM-Welle. Hierbei wird die EM-Welle im Empfänger **abgetastet**.

7.2.1 Probleme

Meist ist die empfangene Welle nicht gleich der gesendeten Welle. Die Welle wird nämlich durch

- Dämpfung
- Reflektion
- Brechung etc

verzerrt. Dadurch ist zwar eine Kommunikation ohne Sichtverbindung möglich, aber mehrere **reflektierte Pfade interferieren**, und **Echos** lassen sich schwer filtern.

Zudem ist die **Synchronisation** schwierig, da festgehalten werden muss, wann ein **Bit**, **Byte** oder **Packet** beginnt.

Die empfangene Signalstärke lässt sich grob abschätzen durch eine Funktion des Funkkanals und des Abstandes d zwischen Sender s und Empfänger r

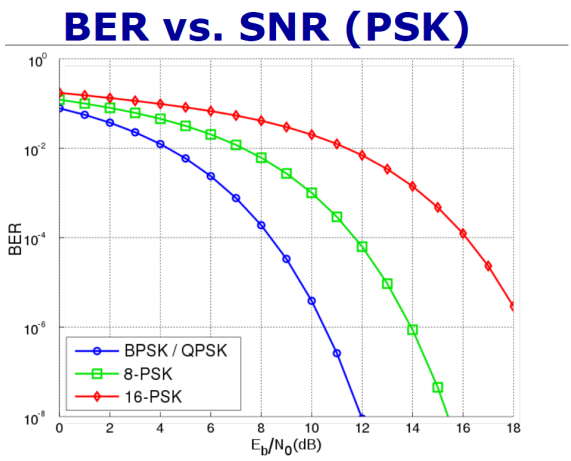
$$P_r = P_s \cdot K \cdot \frac{1}{d^n}$$

n ist dabei in freien Räumen klein, und Gebäuden zwischen 2 und 6.

Weiter Störungen stellen (weißes) Rauschen dar. Außerdem sind Interferenzen möglich, wenn andere Sender im gleichen Frequenzbereich sind. Die **Signal-to-Interference-Noise-Ratio**:

$$\text{SINR} = \frac{P_r}{P_{\text{NOISE}} + P_{\text{INF}}}$$

Aus **SINR** und Art der Modulation resultiert die **Bitfehlerrate (BER)**.



7.2.2 Synchronisation

Synchronisation wird durch **Präambeln** realisiert. Präambeln sind ein spezieller Bitstrom vor jeder Nachricht, der oft sehr lang ist. Dieser erlaubt die Unterscheidung einer **Übertragung** von **Rauschen**.

7.3 Sicherung

Unter Sicherung versteht man die **fehlerfreie Übertragung von Bits** über einen fehlerbehafteten Kanal. Die Sicherungsschicht ist die zweite Schicht im ISO/OSI Modell.

Zur **Fehlerkontrolle** zählen Mechanismen, die fehlerhaft übertragene Bits **erkennen** und **korrigieren**. Hierbei unterscheidet man in zwei Varianten:

- **Backward Error Control:** Ein Fehler wird erkannt und eine Neuübertragung wird veranlasst (kostenintensiv!)
- **Forward Error Control:** Redundante Kodierung des Signals, sodass Bitfehler ohne Neuübertragung erkannt und korrigiert werden können

7.3.1 Backward Error Control

Generell werden hier Bits in Blöcke aufgeteilt. Den Blöcken wird eine Prüfsumme hinzugefügt. Stimmen die Daten überein, schicht Empfänger ein **positive ACK**, ansonsten **negative ACK**. Beim Senden gibt es dabei die Varianten **Stop-And-Wait** (jedes Frame muss bestätigt werden) und **Sliding-Window** (Sequenz wird bestätigt).

Dieses Vorgehen erzeugt nur bei Neuübertragung Mehraufwand und ist bei geringer BER zu bevorzugen.

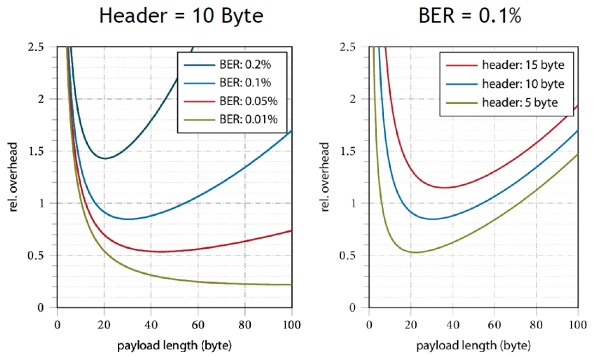
7.3.2 Forward Error Control

Dieses Vorgehen ist dagegen recht komplex. Hier werden Bits zu **Symbolen** zusammengefasst, und es gibt eine injektive Abbildung der Symbole auf Übertragungssymbole. Eine **korrekte** inverse Abbildung ist sogar bei Fehlern möglich, wenn die Azahl nicht überschritten wird.

Dieses Vorgehen erzeugt immer Aufwand, jedoch bei hoher BER zu favorisieren.

7.4 Framing

Unter Framing versteht man das Zusammenführen von Bits/Symbolen zu Paketen. Hierbei muss ein Tradeoff zwischen langen Paketen (evtl. mehr Bitfehler) und kurzen Paketen (hoher Aufwand für Header).



7.5 Link Management

Viele Protokolle gehen davon aus, dass ein Knoten seinen Nachbarn kennt. Ist bei WSN in der Regel jedoch **nicht** der Fall!
Ein Knoten kann als Nachbar vermutet werden, wenn:

- SINR $\geq$ min
- BER, PER $\leq$ max

Die Qualität der Kommunikation zu den Nachbarn ist dabei **irregulär** und nicht zwangsweise proportional zum Abstand. Es ist über die Zeit variabel.

Für manche Anwendungen braucht ein Knoten das Wissen über die Position der Nachbarn. Dabei kann die Nachbarliste gemanaged werden durch eine adaptive Abschätzung der Linkqualität.

$$P_n = \alpha P_{n-1} + (1 - \alpha)\frac{r_n}{r_n + f_n}$$

r_n: im Intervall empfangene Pakete
f_n: verlorene Pakete

8 Medienzugriff

Knoten reden kreuz und quer - Wer bestimmt da den korrekten, konfliktlosen Zugriff von Ressourcen?

Medienzugriff (**MAC**, 2.Schicht ISO/OSI) regelt den Zugriff der Knoten auf ein gemeinsam genutztes Kommunikationsmedium. Dadurch sollen wechselseitige Störungen vermieden werden. Das Hauptziel ist die Reduktion des Energieverbrauchs, denn das Funkmodul ist primärer Energieverbraucher.

Dies impliziert, dass möglichst viel geschlafen werden sollte, dafür aber taub.

8.1 Probleme

Die Energie wird oft einfach verschwendet, zum Beispiel bei **nutzlosem Lauschen**, **Kollisionen** von Nachrichten bei Überlagerung, **Mithören** von Nachrichtenm die man selbst nicht braucht, und **Protokoll-Overhead**.

8.2 Kollisionen

Kollisionen entstehen, wenn eine Empfänger mehrere Sender gleichzeitig hört. Dabei kann ein Sender die Kollision nicht erkennen, wenn er nicht im Empfangsbereich des anderen Senders ist.

Eine mögliche Lösung ist der **Capture-Effekt**, wo der Empfänger die Nachrichten des Senders verstärkt, wenn der Sender stärker ist. Alternativ kann man mehrere Synchronisierte Sender durch ein **logisches ODER** beim Empfänger abhören (gilt nur bei on-off-keying).

Um Energie zu sparen, muss man Abstriche bei anderen Eigenschaften machen. Hierzu zählen **Fairness**, **Durchsatz** und **Nachrichtenverzögerung**.

8.3 Verkehrsaufkommen

Weiteres Einsparpotenzial gibt es durch Beobachtung und Nutzung typischer Muster im Nachrichtenverkehr:

- **Convergecast**: Alle Sensoren senden zur Senke
- **Broadcast**: Senke sendet an alle Knoten
- **lokale Interaktion**: Nachabrn tauschen Nachrichten untereinander aus

Auch die Art der Nachrichtengenerierung ist ein wichtiger Faktor. Man unterscheidet zwischen **gleichmäßigen Strömen** (regelmäßiges Auslesen) und **Burst** (Ereignis löst hohe Aktivität aus).

8.4 MAC Einführung

MAC Protokolle haben verfolgen zwei Ziele: Zum einen **Verdrängung** (Contention), wobei der Zugriff auf Medien wahlfrei ist und Kollisionen weitesgehend vermieden werden. Zum anderen **Zuteilung** (Schedule), wo ein Medium exklusiv zu den Knoten zugeteilt wird. Hierbei ist der Knoten nur während eines bestimmten **time slice aktiv** $\implies$ Zeitsynchronisation notwendig!

8.5 Carrier Sense Multiple Access/Collision Avoidance

Bei **CSMA** wird der Kanal abgehört. Wenn dieser frei ist, wird gesendet.
CA reguliert Kollisionen durch Kanalreservierung:

1. Request to send: Anforderung mit geplanter Dauer T
2. Clear to send: Bestätigung mit T
3. DATA nach Empfang von CTS
4. ACK nach Empfang von DATA

Die Anzahl der **Kollisionen** steigt, je mehr Stationen auf das Übertragungsmedium Zugriff haben wollen. Durch lange Leitungen, sehr viele Stationen und Repeater (Signalaufbereiter und -verstärker) entstehen je nach Ort der Einspeisung unterschiedliche Signallaufzeiten. Sie führen dazu, dass eine Station ein freies Übertragungsmedium feststellt und ihr Signal sendet, obwohl das Signal einer anderen Station bereits unterwegs ist. Es kommt zur Kollision, also der Überlagerung von zwei Signalen.

Verzögerung-Strategien wie **Binary-Exponential-Backoff** (gleichverteilte Verzögerung aus festem Zeitfenster) klappen hierbei nur bedingt. Die Zeitfenster verdoppeln sich nach jeder erneuten Kollision, dies führt unter Umständen zu sehr langen Verzögerungen.

Sift ist eine Alternative zu BEB. Hierbei sind die Zeitfenster ebenfalls fest, aber die Wahl der Slots ist nicht gleichverteilt. Wenige Knoten wählen frühen Slot, wodurch die Kollisionswahrscheinlichkeit klein ist.

Will man nun in WSN CSMA/CA verwendet, so steht man vor einem Problem. Man möchte aus Gründen der Energieeffizienz für die Dauer T das Radio ausschalten. Jedoch verpasst man dadurch andere **RTS/CTS**! Zudem muss der Protokeoll Overhead für jedes Datenpaket betrachtet werden, oder für mehrere Pakete (keine Fairness!).

8.6 S-MAC

Sensor-MAC funktioniert nach dem Rendevouz-Schema. Die Knoten verabreden quasi eine Zeit, in der sie miteinander arbeiten/kommunizieren. Es ist also eine **CSMA/CA mit Duty-Cycling**, die Knoten sind untereinander synchronisiert; auf kurze Aktivphasen folgt eine lange Schlafphase.

Synchronisierung bei S-MAC geschieht während der **SYNCH Phase**. Ein neuer Knoten lauscht dabei, und übernimmt das **Schedule**, ansonsten hält er sich an ein eigenes Schedule. Die **Latenz** bei Hops wird reduziert indem ein Knoten 3 nach **Lauschen des CTS** eine zusätzliche Aktivphase einführt und Knoten 2 **nach ACK direkt ein RTS** an 3 sendet.

$\Longrightarrow$ Problem: Was wenn viele Nachbarn mit verschiedenen Schedules vorhanden sind?

8.7 Timeout MAC

Im Gegensatz zu S-MAC, wo die Aktivphase eine feste Länge hat, wird bei T-MAC die Aktivphase **an das Verkehrsaufkommen adaptiert**. Der Knoten wechselt frühzeitig in die Schlafphase, wenn eine gewisse Zeit T_A nichts passiert.

Early Sleeping: Knoten geht zu früh in die Schlafphase, CTS blockiert Medium. Lösung: **FRTS**
Volle Puffer: Es kann vorkommen, dass man zum Empfangen gezwungen wird. Als Lösung wird vorgeschlagen, direkt ein RTS zu senden, statt mit CTS zu antworten. Jedoch erst, wenn der Puffer mehrmals überlief.

Präambeln: Diese werden zur Synchronisation benötigt. Wenn Knoten während der Aktivphase Präambel hört, bleibt er wach zum Datenaustausch. Jedoch sind Präambeln unnötig lang, wenn der Knoten schon wach ist. Stattdessen könnte man **mehrmals die Daten** schicken oder **Wakeup-Pakete**, die den Zeitpunkt der Datenübertragung enthalten.

8.8 B-MAC

Bei B-MAC wird das digitale Tonsignal in der Austastlücke eingespielt, also mit den Bildsignalen zeitgemultiplext. Da es nicht genügend Platz für eine geeignet große Anzahl an Tonkanälen bietet, wurde es hauptsächlich für Satelliten-Überspielungen zwischen Fernsehstationen verwendet sowie (verschlüsselt) für die Versorgung im Ausland stationierter Einheiten der US-Army mit Satellitenfernsehen.

B-MAC basiert auf einfachen Präambeln, die Philosophie ist: Minimum an Funktionalität mit einfacher Schnittstelle (TinyOS) $\Longrightarrow$ maximale Flexibilität!

8.9 IEEE 802.15.4 “ZigBee”

Bei ZigBee gibt es mehrere Knotenrollen:

- PAN Coordinator
- Coordinator
- Device (Reduced Function Device)

Stern (Beacon-Mode) und Peer-to-Peer (Non-Beacon-Mode) Topologien möglich.

8.9.1 Beacon Mode

Coordinator und Device bilden sternförmiges Netz, wobei der Coordinator ständig Beacon sendet zwecks **Synchronisation** und **Zeitanteilung** im Frame.
Im **Active Period** werden Daten per CSMA übertragen (oder TDMA). Im **Inactive Period** wird geschlafen.

9 Routing

Klassische Netze basieren typischerweise auf Adressen. Die verwendete Hauptkommunikationsform ist **Unicast** (1zu1 Kommunikation) oder **Multicast** (Nachricht an mehrere).

Bei Sensornetzen ist das etwas anders. Es wird vorwiegend **Convergecast** (alle senden Nachrichten an die Senke) und **Broadcast** (Senke sendet Nachricht an alle) verwendet. Zudem gibt es auch lokale Interaktion zwischen zwei Knoten.

Convergecast ist die wichtigste Primitive. Es ist ein Spannbaum mit Senke an der Wurzel. Jedoch ist nicht klar, wie der TradeOff zwischen **Baumtiefe** und **Knotenentfernung** ist.

Es stellt sich also die Frage, wie man stabile Spannbäume erstellt bzw. auswählt. Die Prioriätatn wären dabei:

1. Link-Qualität
2. Nachbar-Verwaltung
3. Routen-Auswahl

9.1 Link-Qualität

Man benötigt einen Mechanismus zur Abschätzung der Qualität der Verbindung zu einem Knoten. Gemessen wird das anhand der **Paketverlustrate (PER)**

$$P_n = \alpha P_{n-1} + (1-\alpha) \underbrace{\frac{r_n}{r_n + f_n}}_{1-\mathrm{PER}}$$

Exponentially Weighted Moving Average

$$P_n(t) = \alpha P_{n-1}(t) + (1-\alpha)(1-\mathrm{PER})$$

9.2 Nachbarverwaltung

In WSN sind die Sensornetze dicht (> 200 Nachbarn), viele Nachbarn haben dabei eine schlechte Link-Qualität, nur wenige sind gut. In der Regel wird eine **Nachbarschaftstabelle** verwaltet, aus der T beste Nachbarn gefunden werden müssen.

Für die **Auswahl des** T gibt es verschiedene Herangehensweisen:

- Xue and Kumar: $T > 5.1774 \log n$: fast sicher verbunden
- Penrose: Wenn jeder T Nachbarn hat, dann gibt es zwischen 2 Knoten wahrscheinlich T Pfade

9.3 Gute Nachbarn

Jeder Knoten sendet ein "Hallo". Jeder Empfänger notiert sich Knoten, von denen die meisten "Hallo" empfangen worden sind. Ist er schon drin, wird er verstärkt. Ist die Tabelle voll, so wird per **LRU** oder **FIFO** ein alter Knoten gelöscht.

Dabei könnte man auch für jedes Inkrement ein Dekrement durchführen. Am Ende gibt es einen **einzigen Kandidaten**.

Am Ende sind die Nachbarn wahrscheinlich gut, aber durch **häufige Einfügeoperationen** ändern sich die Nachbarn oft, **nicht stabil!**. Ein Einfügekriterium wäre, die Nachbarn mit Wahrscheinlichkeit $P = \frac{T}{N}$ einzufügen, wobei N nun nicht bekannt ist...

9.3.1 Nachbarn schätzen

Man kann die Anzahl der Nachbarn mithilfe einer Hashfunktion schätzen... siehe: "Probabilistic Counting Algorithms for Data Base Applications"
$\Longrightarrow$ nun hat jeder noten eine stabile Menge an guten Nachbarn!

9.4 Gute Links

Ein guter Link ist ein Link mit **geringem Paketverlust**. Man vedient sich einer Routing-Metrik:

$$m(L) = \frac{1}{Q_{in}(L)} \cdot \frac{1}{Q_{out}(L)}$$

Dabei sind Links häufig asymmetrisch, die Knoten müssten also die Nachbarn über die Qualität derer Nachbarn aufklären.

9.4.1 Guter Baum

Guter Spannbaum bedeutet, dass es einen guten Pfad von jedem Knoten zur Senke gibt. Man möchte also den kürzesten Pfad gemäß Routing-Metrik finden:

$$\min \sum m(L)$$

Verfahren? **Distance Vector Routing:** Jeder Knoten merkt sich Abstand zur Senke und zu aktuellem Vater V. Beim **Update** schicken die Knoten ein Paket P mit Abstand D zur Senke an Nachbarn. Der Nachbar empfängt dann P über Link L.
Falls $P.D + m(L) < D$:
$D := P.D + m(L)$
$V := S$
Update an Nachbarn sofort verschickt.

Stabilität (periodische Ausführung führt zu geringem Vaterwechsel), **Zyklen** (selbst gesendete Pakete erkennen) und **Fairness** (lokale Nachrichten haben Priorität vor anderen)

9.5 Aggregation und Interaktion

Theoretisch könnte jeder Knoten die Daten seiner Kinder aggregieren. Man muss jedoch berücksichtigen, welche Daten von Interesse sind, von welchem Knoten sie kommen, wie oft sie aggregiert werden etc.

Fluten mit beschränktem Hop-Radius gilt als eine Art der lokalen Interaktion. Dabei wird die Nachricht mit Hop-Radius r verschickt, diese wird dann mit $r-1$ weitergeleitet usw.
Probleme beim Fluten wäre eine **Implosion** (identische Daten erreichen Knoten über mehrere Wege) oder eine **Überlappung** überlappende Sensorknoten hat selbe Daten mehrfach.

FireCracker: Die Nachricht entlang des Spannbaums verteilen und dann Fluten.

Andere Methoden:

- Geo Routing (Schick an bestimmte Position)
- Face Routing (Polygone im Netz)

10 Zeitsynchronisation

Zeitsynchronisation dient dem gemeinsamen Zeitverständnis. Alle Uhren zeigen die selbe Zeit, Knoten A weiß also, wie spät es bei Knoten B um 12 war. Nützlich für Interaktion mit **Nutzer**, **Koordination im Sensornetz** und die Interaktion des Netzes mit der **realen Welt**.

- Nutzer: Wann soll was beobachtet werden? Wie lange dauert ein Vorgang?
- Reale Welt: konsistente verteilte Messungen, Fusion verteilter Beobachtungen
- Sensorknoten: drahtlose Kommunikation, Energieeffizienz, Lokalisierung

10.1 Arten von Synchronisation

- **Intern/Extern:** Knoten einigen sich auf eine Zeit vs. Zeit wird von außen vorgegeben
- **Lebenszeit:** immer, oder dann, wenn benötigt? Energie!
- **Scope:** Welche Knoten? Alle oder Manche?
- **Rate/Offset:** Uhren laufen gleich schnell, es ist also die **Dauer** wichtig, oder die Uhren zeigen alle die **gleiche Zeit**
- **Skalen und Uhren:** Lokale Uhren auf gleicher Zeit oder Uhren mit unterschiedlichen skalen?
- **Zeitpunkte/Intervalle:** fehlerhafter Zeitpunkt oder korrektes Intervall?

Zu den Hindernissen zählen natürlich die Eigenschaften der Hardware Synchronisation und die Eigenschaften der drahtlosen Kommunikation. **Hardware Uhren** zählen Intervalle fester Länge, kann aber von HW zu HW abweichen! Eine Atomuhr ist zwar 10^5 genauer, aber verbraucht 100mW.

Bei der Kommunikation gibt es zudem Schwankungen durch **Interrupts**, **Zugriffe**, **Encoding**, **Decoding** etc.

10.2 Synchronisation von Knoten

Wenn Klinet R die Zeit h_R anzeigt, was zeigt dann die Zeit von K h_K an? Klient kann seine Zeit dann $h_R - h_K$ korrigieren.

10.2.1 Unidirektional

R sendet Nachricht, die h_R enthält, der Klient nimmt dann $h_K := h$ an. Dies ist mit mehr Klienten per Broadcast Problemlos möglich.

10.2.2 Round Trip

h_1 an h_R, h_R an h_2. h_K nimmt dann die Zeit

$$h_K := \frac{h_1 + h_2}{2}$$

Dies ist mit mehreren Knoten jedoch aufwändig!

10.2.3 Referenz Broadcast

Beacon sendet Broadcast an Referenz und Client. Danach kommunizieren die Knoten. Vergleichsweise genau, weil Broadcast nahezu Zeitgleich empfangen wird.

10.2.4 Dauerhafte Synchronisation?

Eine dauerhafte Synchronisation ist schwer umzusetzen. Dabei gibt es 2 Varianten:

- periodisches Wiederholen: Die Genauigkeit hängt von der Synchronisierung ab. Stetigkeit ist ein Nachteil, verbunden mit hohem Energieaufwand
- Anpassung der Ganggeschwindigkeit: Man misst, wie viel h_K schneller als h_R läuft, passt dann die Geschwindigkeit an. Möglichkeit sind **Lineare Regression** oder **Phase-Locked Loops** (Steigung abhängig von der Messung ändern)

10.3 Mehrere Klienten

10.3.1 Out-of-Band

Klient hat eine Clique, und mindestens einen Referenzknoten als Nachbarn. Hierfür ist eine genaue Infrastruktur notwendig!

10.3.2 Cluster

Das Netz ist in Cluster strukturiert, und man hat Gateways, die die überlappenden Cluster verbinden. Wartung der Struktur ist aufwändig!
BSP: Reference Broadcast Synchronization

10.3.3 Baum

Referenz ist an der Wurzel, Kinder synchronisieren sich mit Vätern. Das führt zu langen Pfaden und größeren Ungenauigkeiten, und vergleichsweise ungenauen Pfaden
Bsp: Time Sync Protocol for Sensor Networks

10.3.4 Unstrukturiert

Kein Overlay - kein Aufwand! Knoten nutzen dann beliebige Pfade zur Synchronisation.
Bsp: Time Stamp Synchronization, die Zeitstempel werden entlang eines Pfades syncronisiert, Round Trip zwischen Nachbarknoten

11 Lokalisierung

Man will die Koordinaten eines Knotens in einem (Euklidischen) Koordinatensystem bestimmen. Ziel ist eine genauere **Interpretation** der Sensordaten, **Daten-Fusion** etc. Hier sind die Unterscheidungen ähnlich der Zeitsynchronisation.

GPS für alle Sensorknoten ist aus Gründen der Energie nicht machbar, und Kosten im Vergleich zum Sensorknoten deutlich mehr. Und man braucht eine Sichtverbindung zum Satelliten.

11.1 Lokalisierungs-Algorithmen

In der Regel drei typische Phasen:

1. **Relationen zwischen Knoten messen**
2. **Platzierung**
3. **Verfeinerung**

11.2 Relationen zwischen Knoten

Meist Abstand basiert, **Winkel im freien Feld** jedoch besser. Gemessen wird anhand von **Schall**, Indikatoren sind **Signallaufzeit** und Stärke. Das Signal ist speziell kodiert, der Empfänger sucht nach dem ersten Match (**Echos!**). Genauigkeit best case sind 1cm, Reichweite 10m

ABER: Schall hängt von **Temperatur** ab (Sensor rauf), **Orientierung** (spezielle Reflektoren) und **Hindernisse** (große Fehler durch Multi-Path)

Die empfangene Signalstärke lässt sich abschätzen:

$$P_r(d) \approx P_s \frac{1}{d^n}$$

oder **Received Signal Strength Indicator**

$$\text{RSSI} = 10 \log_{10} P_s - 10n \log_{10} d$$

Bestenfalls sind 2m-3m möglich.

Ausrichtung der Antennen zueinander und Höhe über Boden hat starken Einfluss (Reflektion/Dämpfung)! Indoor ist RSSI quasi **unkorreliert**!

Interferometrie ist genauer, aber Funktionalität ist unklar!

11.3 Platzierung

Die Position ist durch **Anker mit bekannter Position** und Relationen zwischen Knoten gegeben. Gesucht sind die Positionen von Nicht-Anker-Knoten.

11.3.1 Zentroid

Knoten liegt im Schwerpunkt aller Anker.

$$x, y = \frac{1}{n} \sum_{i=1}^{n} x_i, y_i$$

11.3.2 Bounding Box

Platziere Knoten im Schnitt der Bounding Boxen um alle Anker.

$$x = \frac{1}{2}(\min_i(x_i + d_i) + \max_i(x_i - d_i))$$

11.4 Multi-Hop

Bisher hat jeder Knoten 3 Anker gebraucht, skaliert schlecht! Sehr viele Anker sind keine Lösung.

Multi-Hop: Jeder Anker flutet das Netz, in jeder Nachricht ist der kürzeste Abstand zum Anker vermerkt

Rekursion: Berechnete Positionen werden als weitere Anker betrachtet $\implies$ Fehler akkumuliert sich!

11.5 Verfeinerung

Jeder Knoten bekommt eine Abschätzung seiner Position. Diese verbessert er **iterativ**, indem er alle seine Nachbar als Anker verwendet. So lange, bis keine signifikante Veränderung der Position mehr da ist.

12 Datensammlung und Verarbeitung

12.1 Sampling

Sampling bedeutet Messung einer bestimmten Quantität $q(x, y, t)$ in regelmäßigen Abständen. Die Frage ist, wie groß man f_t und f_x wählt. Man möchte eine **Aliasing** verhindern, bei dem hohr Frequenzen wie tiefe Frequenzen erscheinen, weil die Sampling-Rate zu gering ist.

Nyquist Kriterium: f_t doppelt so groß wie der höchste **zeitliche** Frequenzanteil, f_x doppelt so groß wie der größte **räumliche** Frequenzanteil

12.2 Resampling

Die Verteilung der Sensoren ist typischerweise **nicht uniform**. Dadurch **interpoliert** man stückweise das Signal, eine **Approximation**, mit der man zB Werte an Gitterpunkten bestimmt.

12.3 Coverage

Man möchte sicherstellen, dass eine hinreichende räumliche Sampling-Rate f_x ausreichend ist. Oftmals hat man mehr Sensoren als notwendig, man schaltet also nur so viele wie nötig ein, und behält die restlichen als **Reserve**.

k-Coverage: Jeder Punkt ist im Sensorbereich von mindestens k Knoten

12.4 Kalibrierung

Unterschieden wird in **absolute** Kalibrierung und **relative** Kalibrierung.

12.5 Cleaning

Unter Cleaning versteht man das Bereinigen roher Messwerte. Dazu gehören **"Outlier"**, **fehlende Messwerte**, und **Rauschen**.

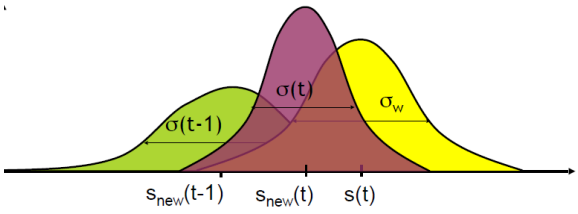
Realisiert wird dies durch verschiedene **Filter**.

- Hochpassfilter (keine Tiefen, Betonung hoher Frequenzen)
- Tiefpassfilter (keine Hohen)
- Mittelwert Filter (Mittelwert in Nachbarschaft der Größe N)
- Median Filter (Sortieren und die Mitte nehmen)
- Konvolution
- Kantenerkennung (Sobel)
- EMWA: $s_{new}(t) = (1 - \alpha)s_{new}(t - 1) + \alpha s(t)$

12.6 Kalman Filter

$$s_{new}(t) = (1-\alpha)\, s_{new}(t-1) + \alpha\, s(t)$$
$$\sigma^2(t) = (1-\alpha)\,\sigma^2(t-1)$$

$$\alpha = \frac{\sigma^2(t-1)}{\sigma^2(t-1)+\sigma_w^2}$$



Man hat ein **Zustandsmodell** und ein **Messmodel**. Man verbinden die beiden Zustände zu einer besseren Abschätzung, je nachdem ob man der Messung oder der Odometrie mehr traut.

Kalman Filter kann mehrere Sensoren mit optimaler Gewichtung integrieren!

12.7 Partikel Filter

Die Verteilung ist beliebig, multimodal. Particle-Filter sind jedoch weniger effizient. Die Idee ist, das eine Verteilung durch die Dichte der Partikel symbolisiert wird.

12.8 Kompression und Prädiktive Filter

Wenn $s(t)$ aus der Vergangenheit berechnet werden kann, dann wird $s(t)$ nicht effizient übertragen!

Man kann Filter wie bisher anwenden, jedoch deren Ausgabe als Vorhersage verwenden!

Mit **Least Mean Squares Adaptation** lässt sich der Kernel automatisch anpassen, wodurch die Vorhersage möglichst genau wird. Das funktioniert so gut, weil **zeitlich benachbarte Daten** sehr ähnlich sind.

12.9 Tracking

Unter Tracking versteht man das Verfolgen der Bewegung eines Objektes (**Position, Richtung, Geschwindigkeit**). Das Objekt taucht dabei auf, bewegt sich auf einem Pfad, und verschwindet dann wieder.

Objekte lassen sich mittels von ihnen emittierten Signale detektieren. Klassifikation erfolgt dann anhand eines **Spektrogramms**. Die Lokalisierung des Objekts ermittelt sich durch den Sensor mit dem höchsten Signal.

Die Detektion von mehreren Objekten ist jedoch schwierig! Objekte sind für Sensoren meist ununterscheidbar.

13 Dezentrale Ansätze

Man möchte ein dezentrales und impliziertes Synchronisieren von Phasen, wie bei Glühwürmchen, erreichen.

Das **Glühwürmchen-Modell** sieht dabei folgendermaßen aus:
Das Licht wird alle L Sekunden angemacht. Wenn Mehrheit der Nachbarn bereits Licht an hat, dann soll die Uhr vorgestellt werden, ansonsten nichts tun. Alternativ vice versa.

13.1 Kuramoto Modell

Es handelt sich um gekoppelte Oszillatoren. Global gilt:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i)$$

Für eine Nachbarschaft $\mathcal{G}_i$ gilt dann lokal:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{|\mathcal{G}_i|}\sum_{j\in\mathcal{G}_i}\sin(\theta_j - \theta_i)$$

13.2 Millenium Bridge

Der Fußgänger auf der Brücke wird als ein Inverses Pendel beobachtet:

$$\ddot{\theta} - \frac{g}{l}\sin\theta = 0$$

13.3 Van-der-Pol-System

Ein Van-der-Pol System ist ein schwingungsfähiges System mit nichtlinearer Dämpfung und Selbsterregung.

$$\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + x = 0$$

- gekoppelte van-der-Pol-Systeme

$$\ddot{x}_i + \lambda_i(\dot{x}_i^2 + x_i^2 - a^2)\dot{x}_i + \omega_i^2 x_i = -\ddot{y}$$
$$\ddot{y} + 2h\dot{y} + \Omega^2 y = -r\sum_{i=1}^{n}\ddot{x}_i$$

- y Amplitude Brücke
- x_i Amplitude Fußgänger (quer zur Gehrichtung)
- λ Dämpfung Fußgänger (bei kleinen Amplituden neg.)
- a Parameter, bestimmt Amplitude
- ω Parameter, bestimmt Frequenz
- Ω Eigenfrequenz der Brücke
- h Dämpfung der Brücke, r Kopplungsstärke

13.4 Skalierbarkeit

Man möchte untersuchen, wie vernetzte Systeme mit der Anzahl der Knoten skalieren. Es werden also dem Netz mehr und mehr Knoten hinzugefügt und geschaut, wie sich die Leistung des gesamten Netzes entwickelt.

- Menschliche Teams: Ringelmann-Effekt
- Computer Server

Bei geteilten Ressourcen erzeugen **Zugriffskonflikte** lange Wartezeiten.

13.4.1 Universal Scalability Law (USL)

Systemkapazität $C(N)$

$$C(N) = \frac{N}{1 + \alpha(N-1) + \beta N(N-1)}$$

α Grad an Zugangskonflikten, β Mangel an Kohärenz.

Man Unterscheidet in vier Bereiche

1. equal bang for the buck
2. Kosten durch Ressourcenteilung
3. Vermehrte Zugangskonflikte
4. negative Rendite
5. “inoffiziel”: Superlineare Skalierung

Unter **Payback-Zone** versteht man die sublineare Skalierung nach einer superlinearen skalierung.