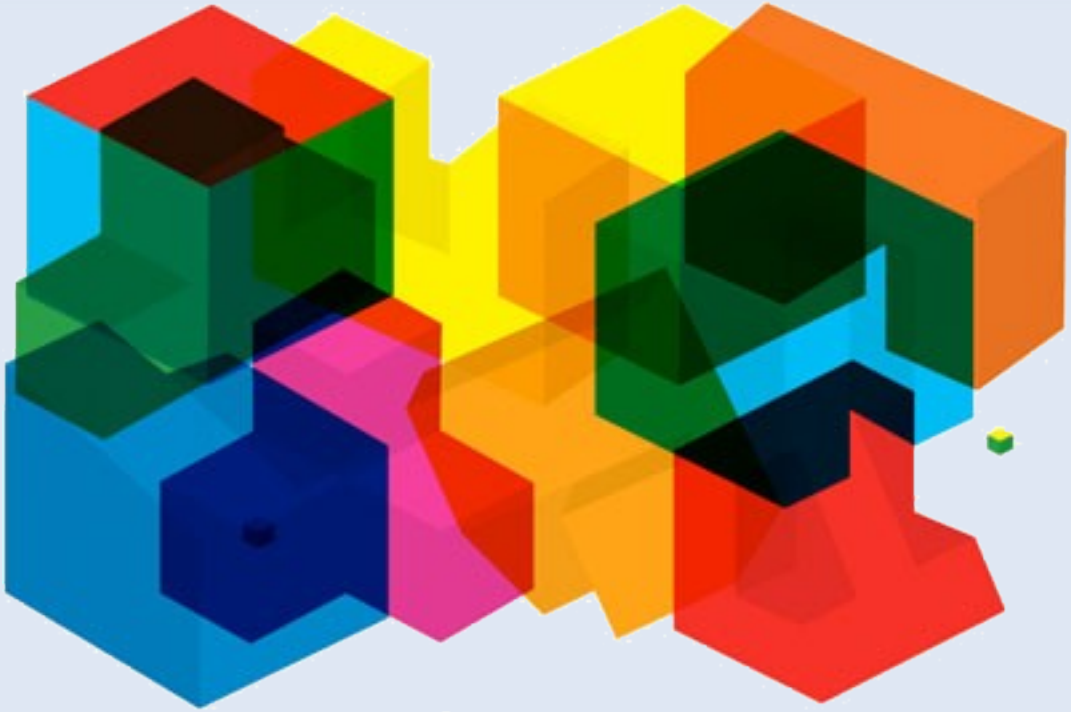
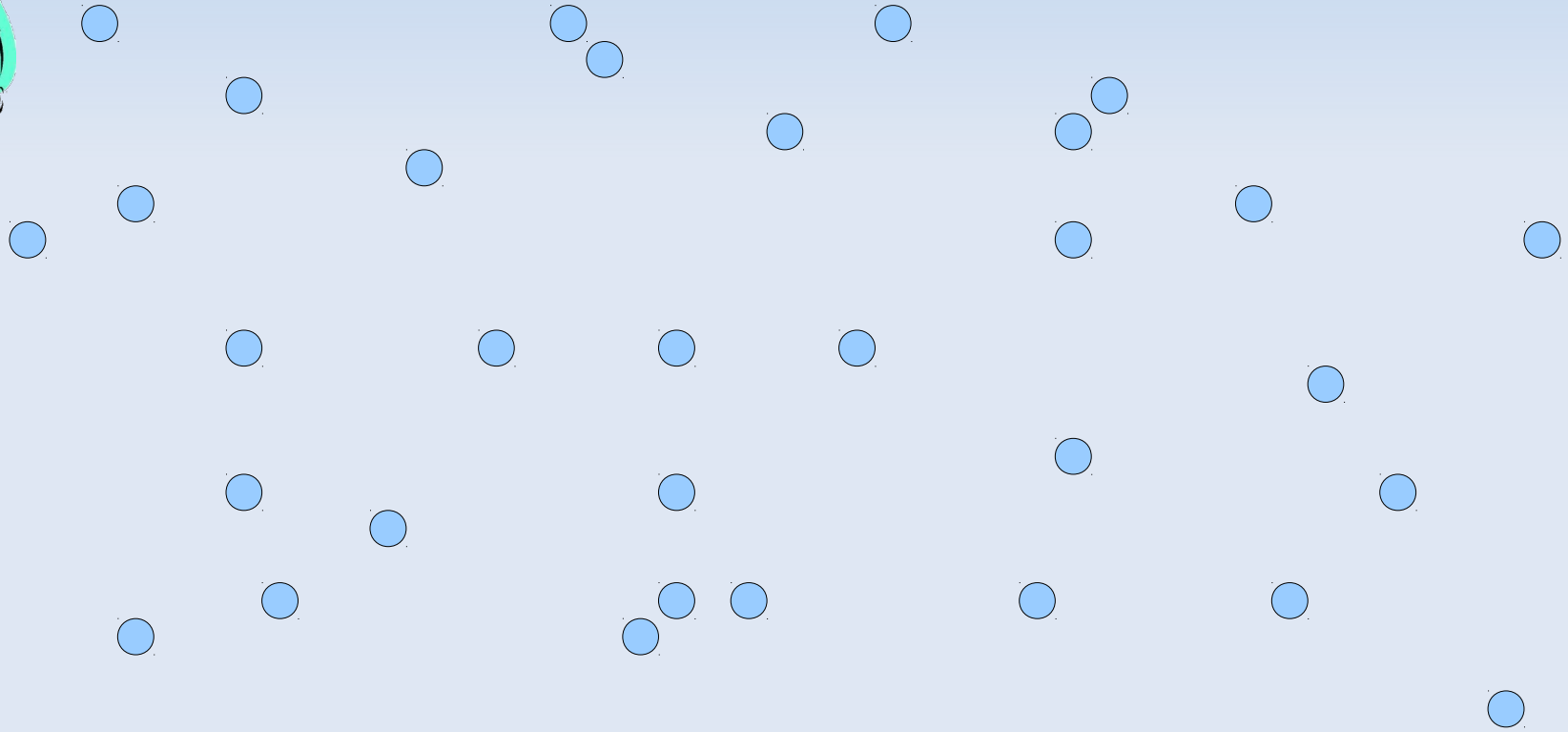
Komponentenbasiertes Clustering in der Wiselib



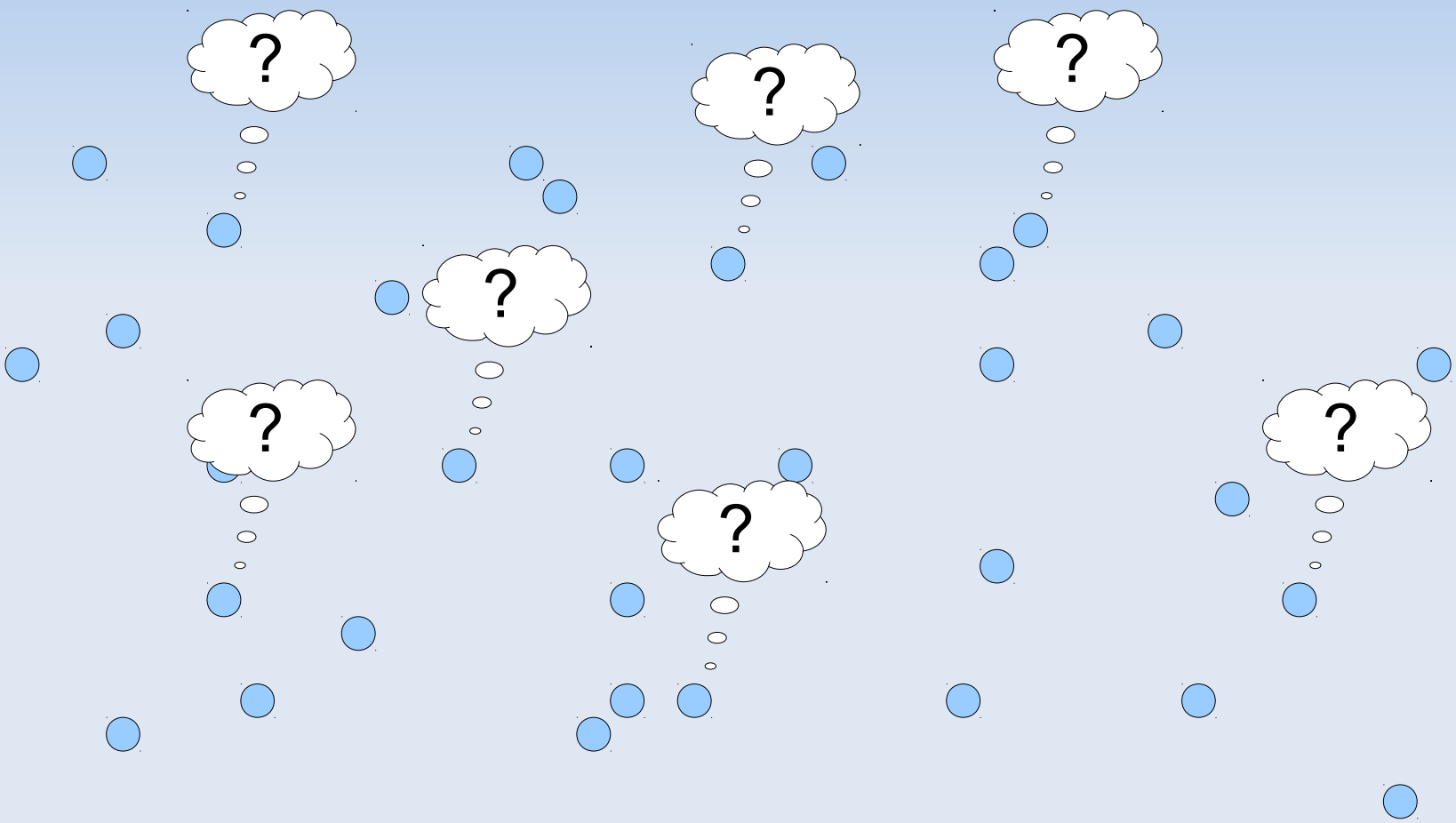
Komponentenbasiertes Clustering in der Wiselib

- Wozu Clustering?
- Grundidee Clustering
- Clustering-Ansätze
- Clustering in der Wiselib
  - Die Komponentenstruktur
  - Vorteile einzelner Komponenten
  - Bisherige Ergebnisse
- Aktuelles
- Ausblick

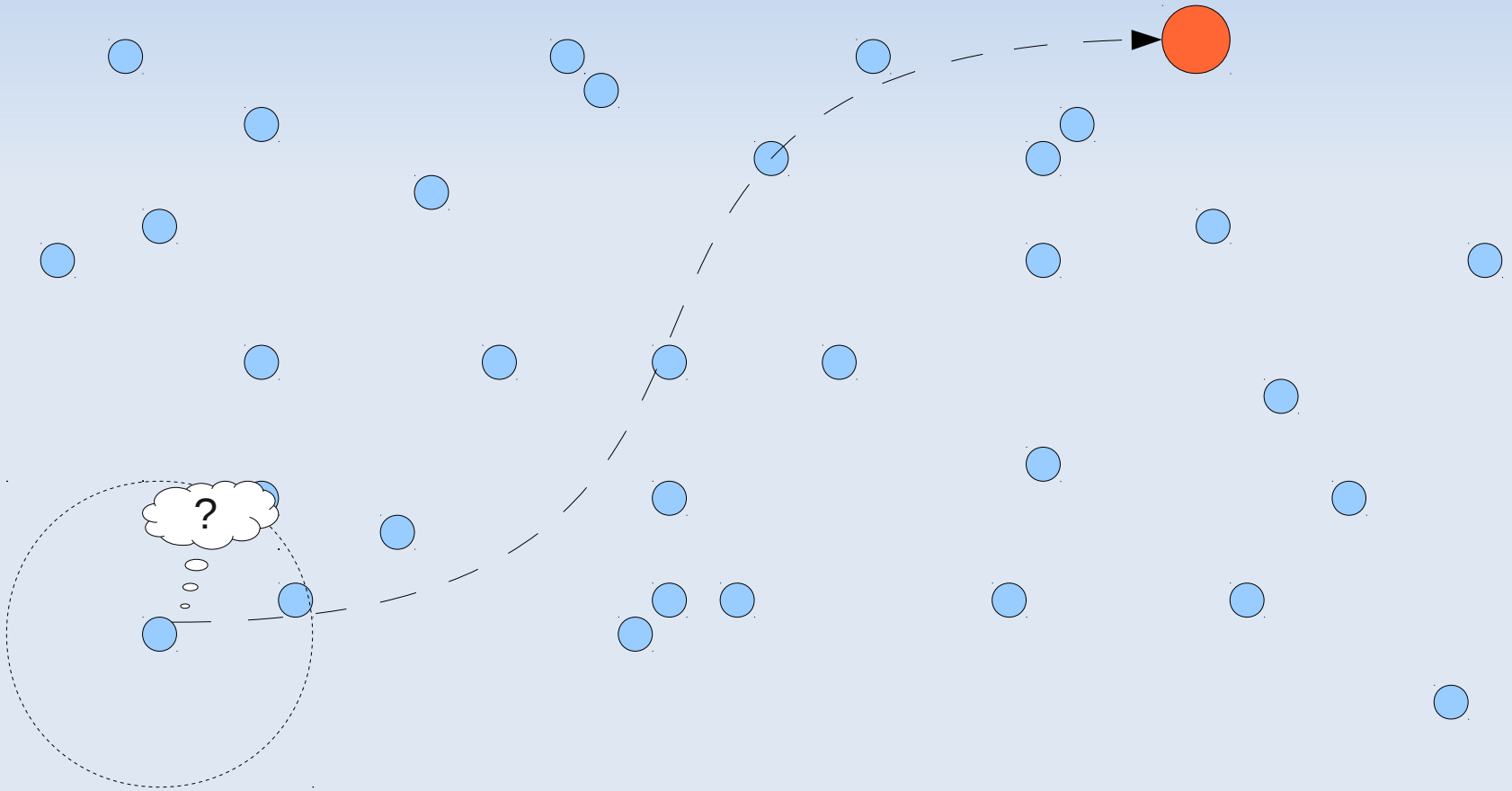
Wozu Clustering?



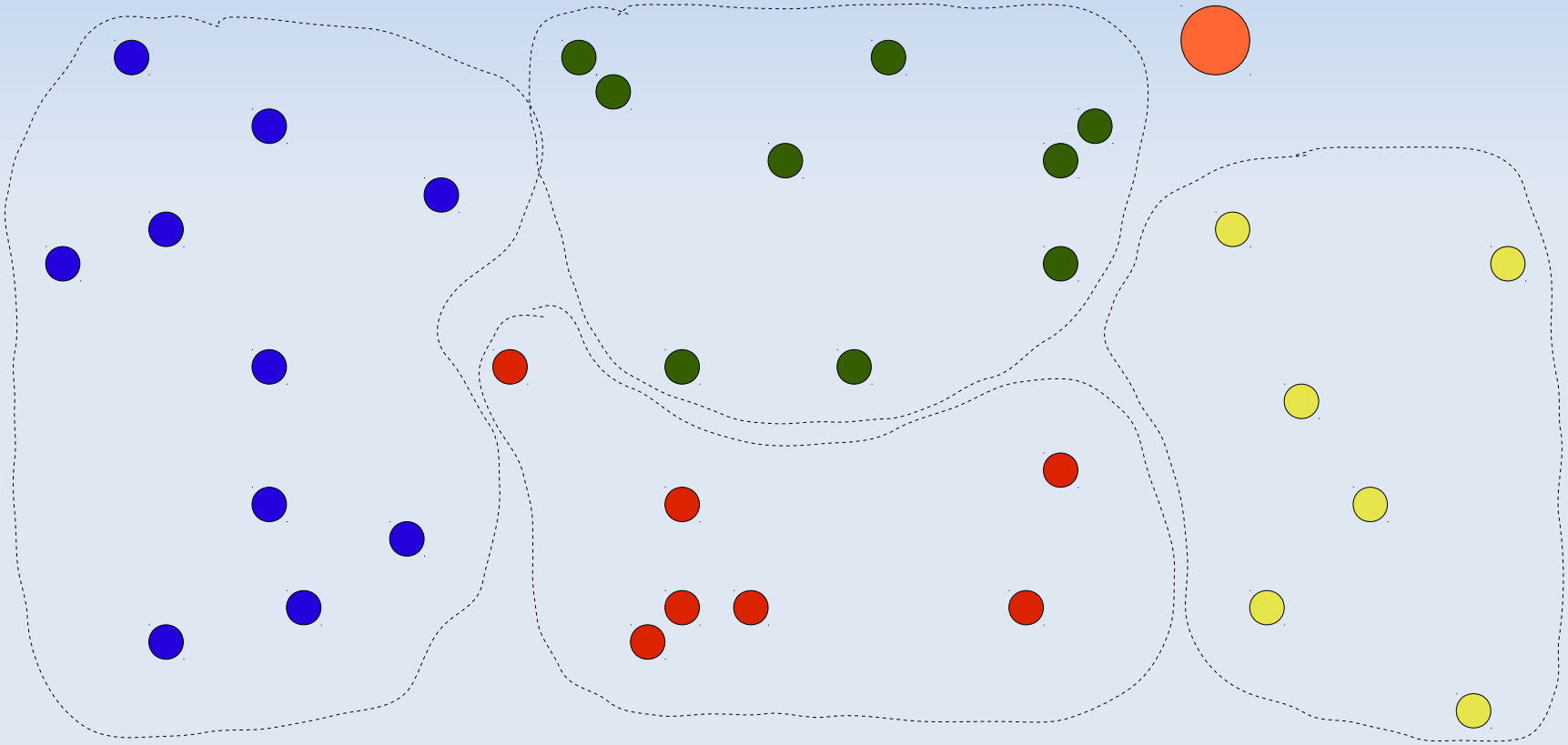
Wozu Clustering?



Wozu Clustering?

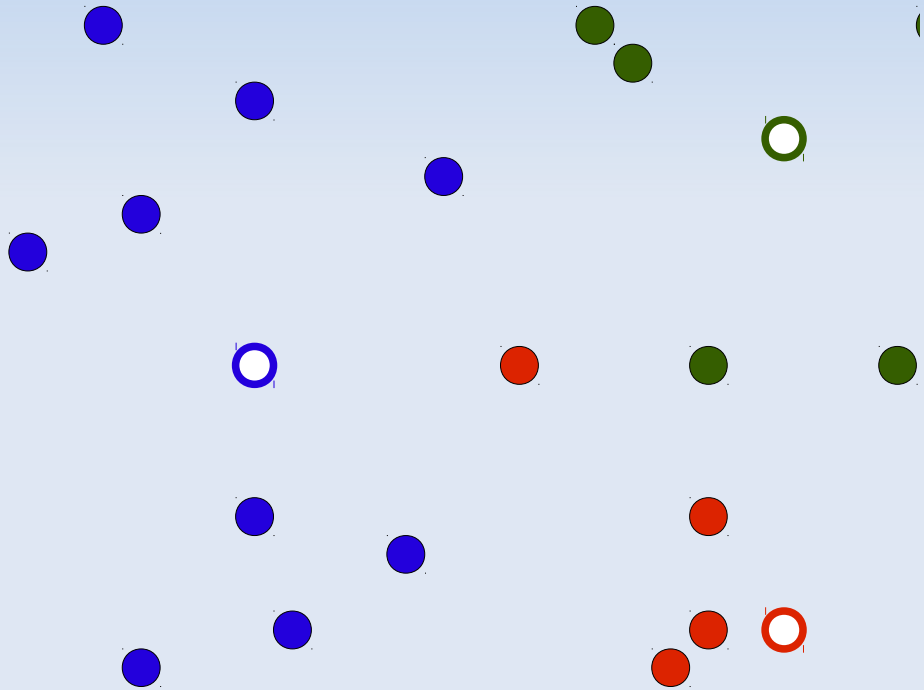


Wozu Clustering?



Grundidee ”Clustering”

- Aufteilung des Netzes in lokale Ansammlungen (Cluster)
- Dabei Ernennung eines ”Cluster-Heads”, der diesen Cluster verwaltet
- Statt Kommunikation von Knoten zu Knoten z.B. effizienter von Cluster (Head) zu Cluster
- Skalierbar, Adaptiv, Hierarchisch
- Ausgestaltung je nach primären Anforderungen (Energieeffizienz, Mobil/Statisch, Re-Clustering, max./min. Größe, ...)



Clustering Ansätze

- Ganz klassisch: LCA
  - Alle Knoten können mit der gleichen Wahrscheinlichkeit p Cluster Head werden
  - Wenn Cluster Head, so werden bis zu k Hops entfernte Knoten darüber informiert
  - Wenn nicht Cluster Head, treten die Knoten dem nächstgelegenen Cluster Head bei
  - Erweiterung: LEACH
    - Cluster Heads wechseln periodisch
    - Somit Aufteilung des zusätzlichen Energiebedarfs
    - Heads informieren nur 1-Hop Nachbarn

Clustering Ansätze

- TCCA
  - Ebenfalls zufallsbasiert
  - Die Wahl der Cluster Heads basiert zusätzlich auf der verbleibenden Energie der Knoten
- MOCA
  - Erzeugt überlappende k-Hop Cluster
  - Knoten können also mehreren Clustern angehören

- Und noch viele weitere, die teilweise völlig anders aufgebaut sind...

Clustering in der Wiselib

- Ausgangspunkt
  - Viele Clustering-Algorithmen bekannt
  - Meist nur speziell für ein WSN-System implementiert
  - Dabei oft tief verwurzelt mit der umgebenden Software, oder gar nicht Open Source
  - Somit schwierig Vergleiche zu ziehen, die Performanz zu untersuchen, oder allgemein die Nutzbarkeit auf verschiedene Szenarien

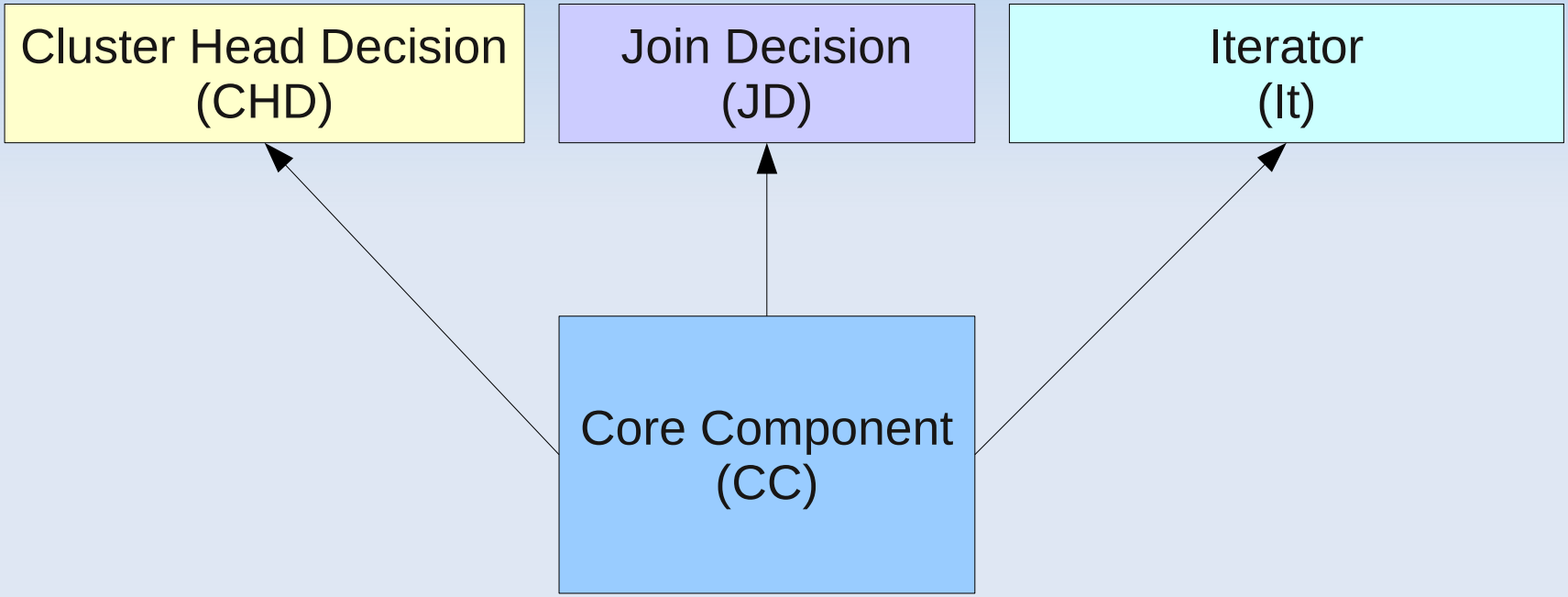
Clustering in der Wiselib

- Ziele
  - Bibliothek verschiedenster Clustering-Algorithmen
  - Einfache Nutzung auf verschiedenartigen Sensorknoten und in beliebigen Anwendungen
  - Wiederverwendbarkeit möglichst großer Teile des Codes
  - Einfache Implementierung neuer Clustering-Algorithmen auf Basis des Vorhandenen

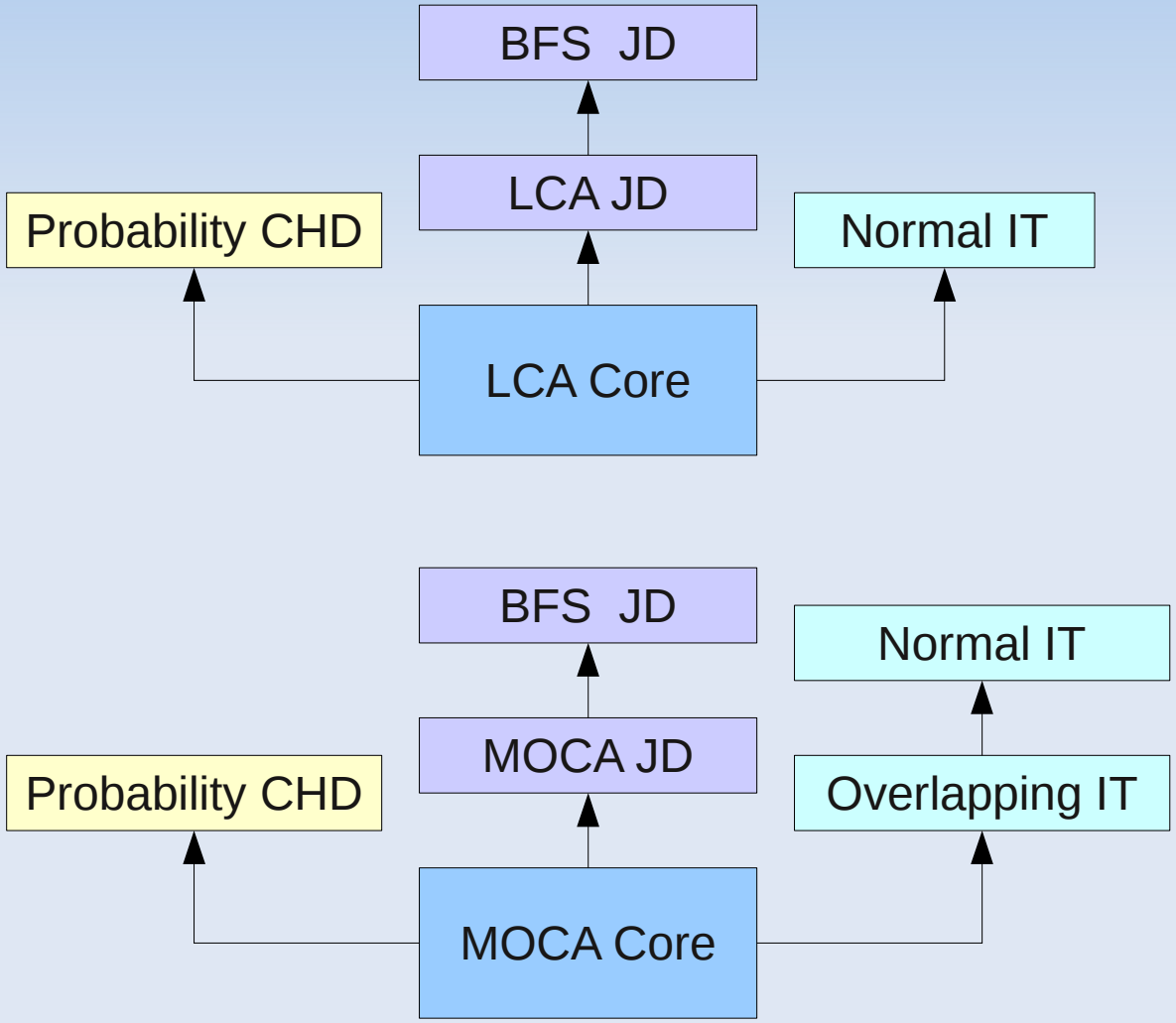
Clustering in der Wiselib

- Idee
  - Identifizierung von in verschiedensten Clustering-Algorithmen auftretenden ”üblichen” Komponenten
  - Vereinheitlichung ähnlicher Komponenten zur Nutzung in gleich mehreren Clustering-Algorithmen
  - ”Baukastenprinzip”: Austausch von Komponenten ermöglicht auch einfaches Bilden neuer Clustering-Ideen und -Abwandlungen
  - Einheitliche Testumgebung für direkte Vergleiche der Komponenten und Algorithmen (via Shawn)

Die Komponentenstruktur



Beispiele


BFS JD
LCA JD
Probability CHD
Normal IT
LCA Core
BFS JD
Normal IT
MOCA JD
Probability CHD
Overlapping IT
MOCA Core

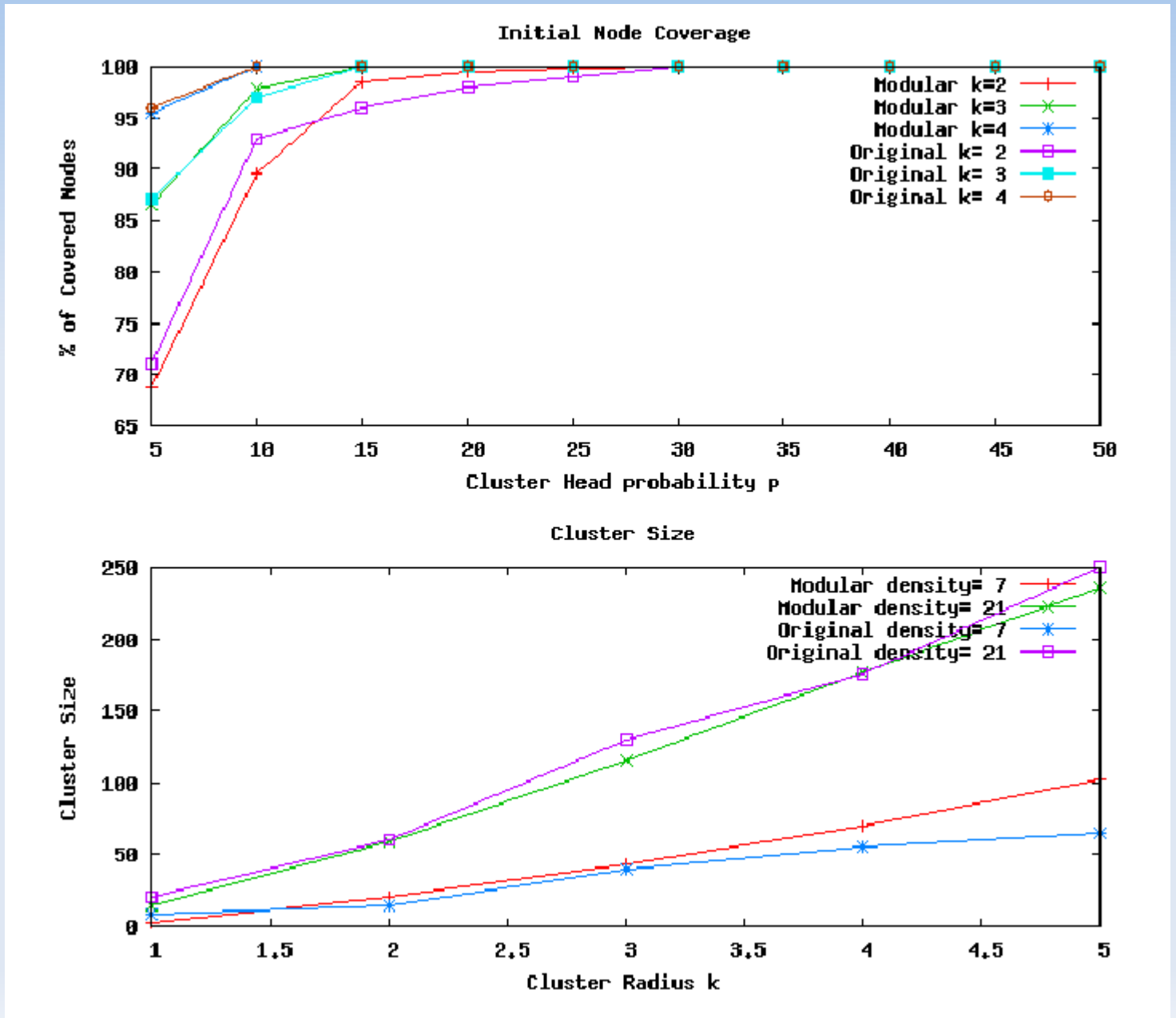
Templates!

```
typedef wiselib::MaxmindClusterHeadDecision<Os> maxmindCHD_t;
typedef wiselib::MaxmindJoinDecision<Os> maxmindJD_t;
typedef wiselib::MaxmindIterator<Os> maxmindIT_t;
typedef wiselib::MaxmindCore<Os, maxmindCHD_t, maxmindJD_t, maxmindIT_t>
      Mmaxmindclustering_t;
```

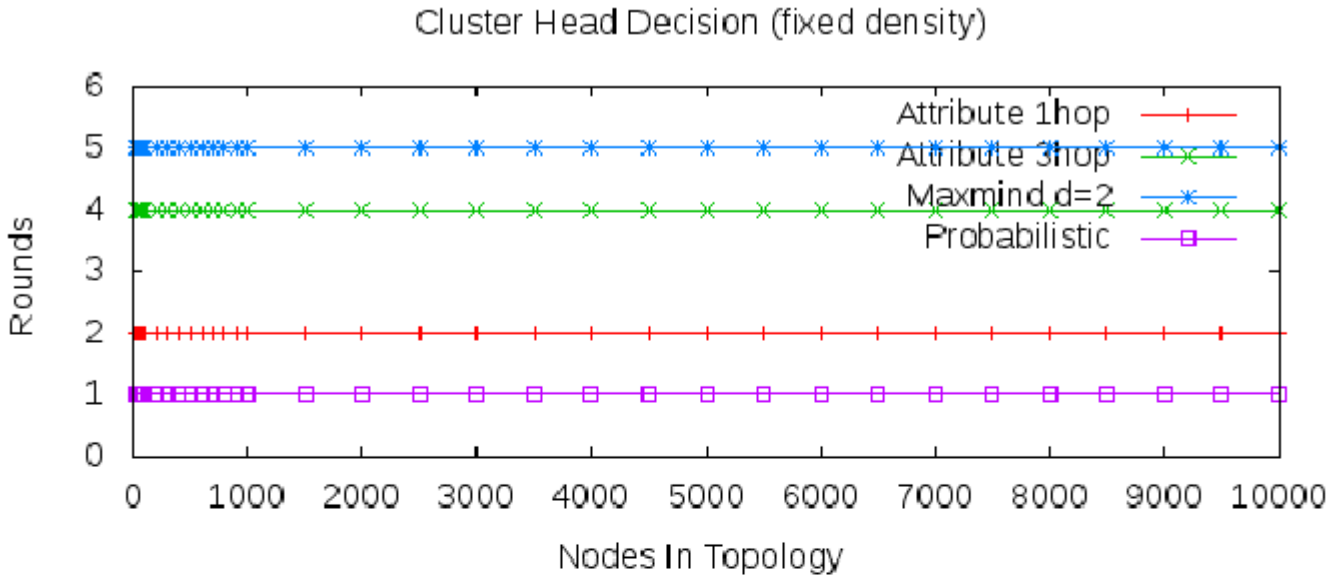
```
typedef wiselib::ProbabilisticClusterHeadDecision <Os, Os::Radio> probabilisticCHD_t;
typedef wiselib::MocaJoinDecision<Os> mocaJD_t;
typedef wiselib::OverlappingIterator<Os> overlappingIT_t;
typedef wiselib::MocaCore<Os, probabilisticCHD_t, mocaJD_t, overlappingIT_t>
       Mmocaclustering_t;
```

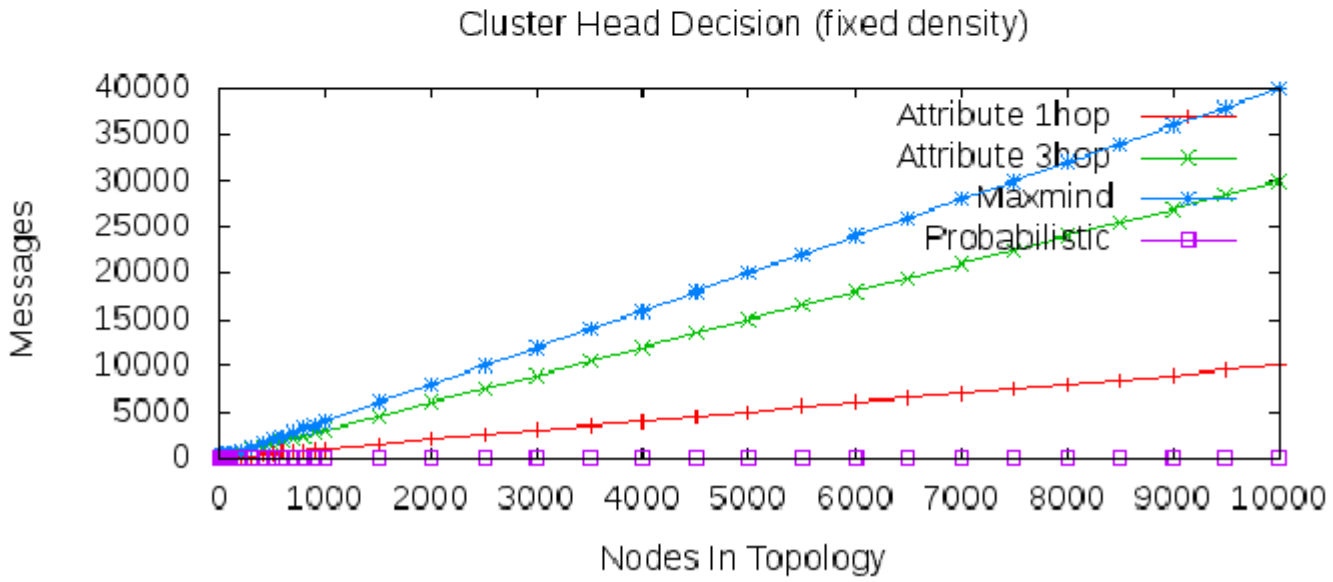
```
typedef wiselib::WaitingTimerClusterHeadDecision<Os> wtCHD_t;
typedef wiselib::SimpleJoinDecision<Os> simpleJD_t;
typedef wiselib::NormalIterator<Os> normIT_t;
typedef wiselib::CawtCore<Os, wtCHD_t, simpleJD_t, normIT_t> Mcawtclustering_t;
```

Einige Ergebnisse



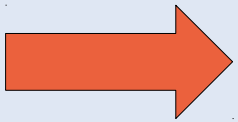
Einige Ergebnisse





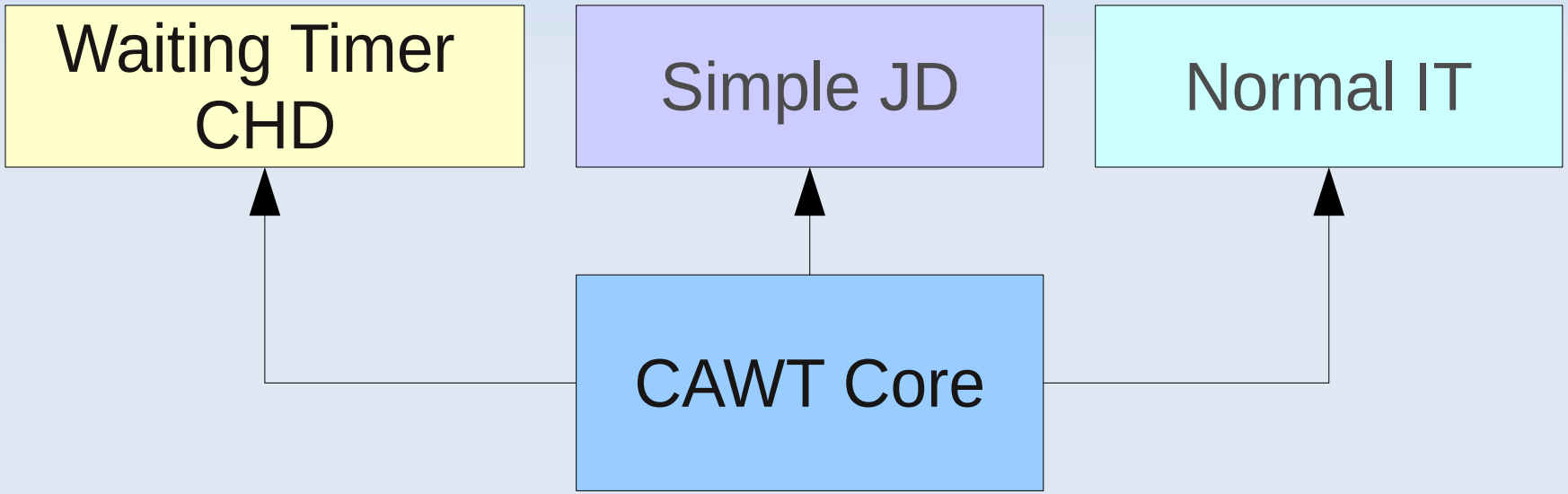
Aktuelles

- CAWT-Algorithmus
  - ”Clustering Algorithm via Waiting Timer”
  - Verwendet neuartige zeitgesteuerte CHD-Komponente
  - Verwendet vorhandene JD und IT Komponenten
  - Somit nur CHD und Core Komponenten wirklich neu
  - Läuft bereits auf Shawn und Flur (juuu!)



Kleine Live Demo am Schluss!

Aktuelles



Aussicht

- Einheitliche Benchmarking-Umgebung, inklusiver weiterer Vergleiche mit Originalpapern sowie untereinander
- Benchmark-Topologien, die möglichst viele Anwendungsszenarien abdecken
- Aufräumen etlicher älterer Fragmente (möglichst einheitliche Schnittstellen, etc.)
- Implementierung weiterer Algorithmen und Komponenten (soweit nötig)

Das war's auch schon

- Danke...

… und tschüss!