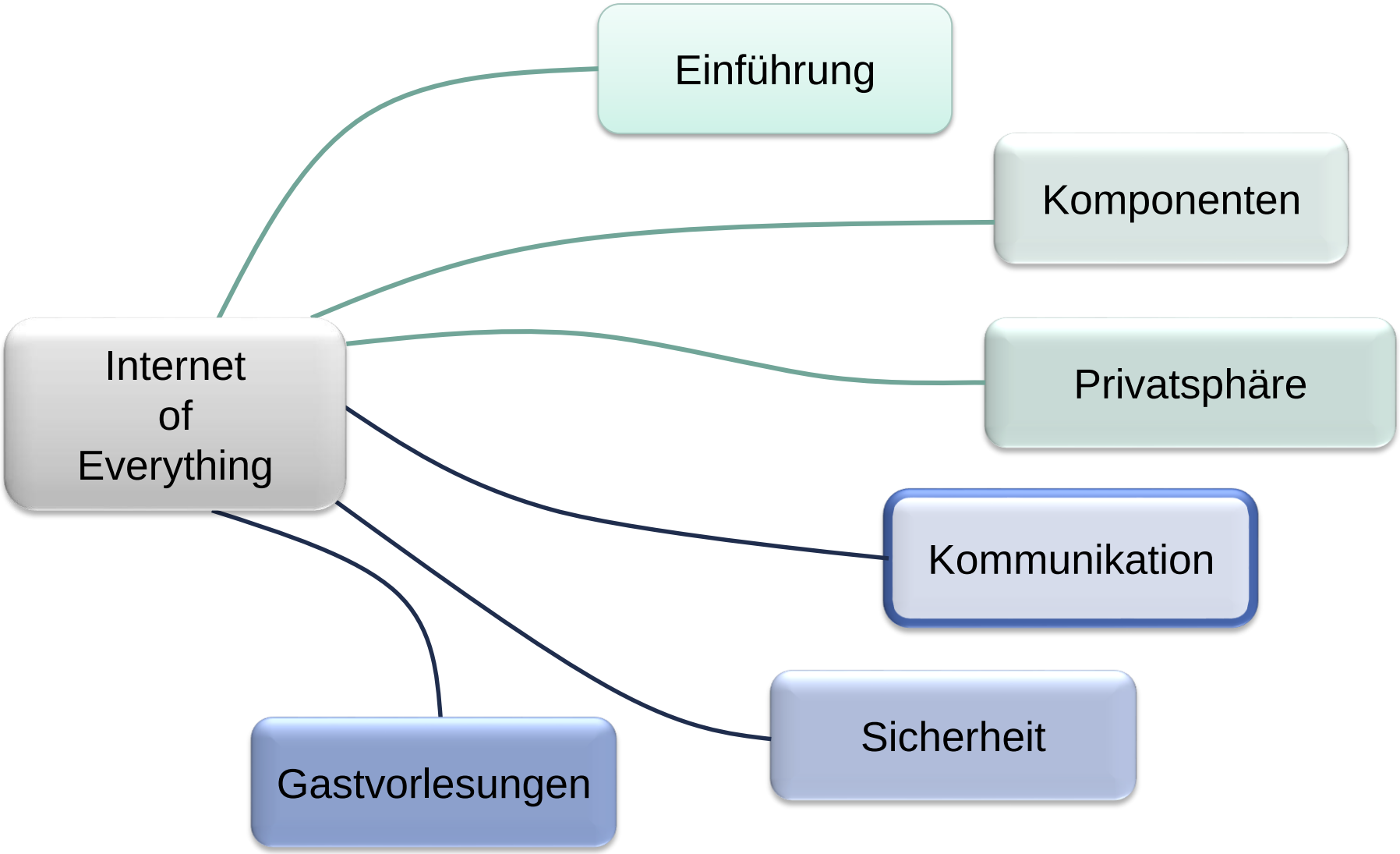
Karlsruher Institut für Technologie

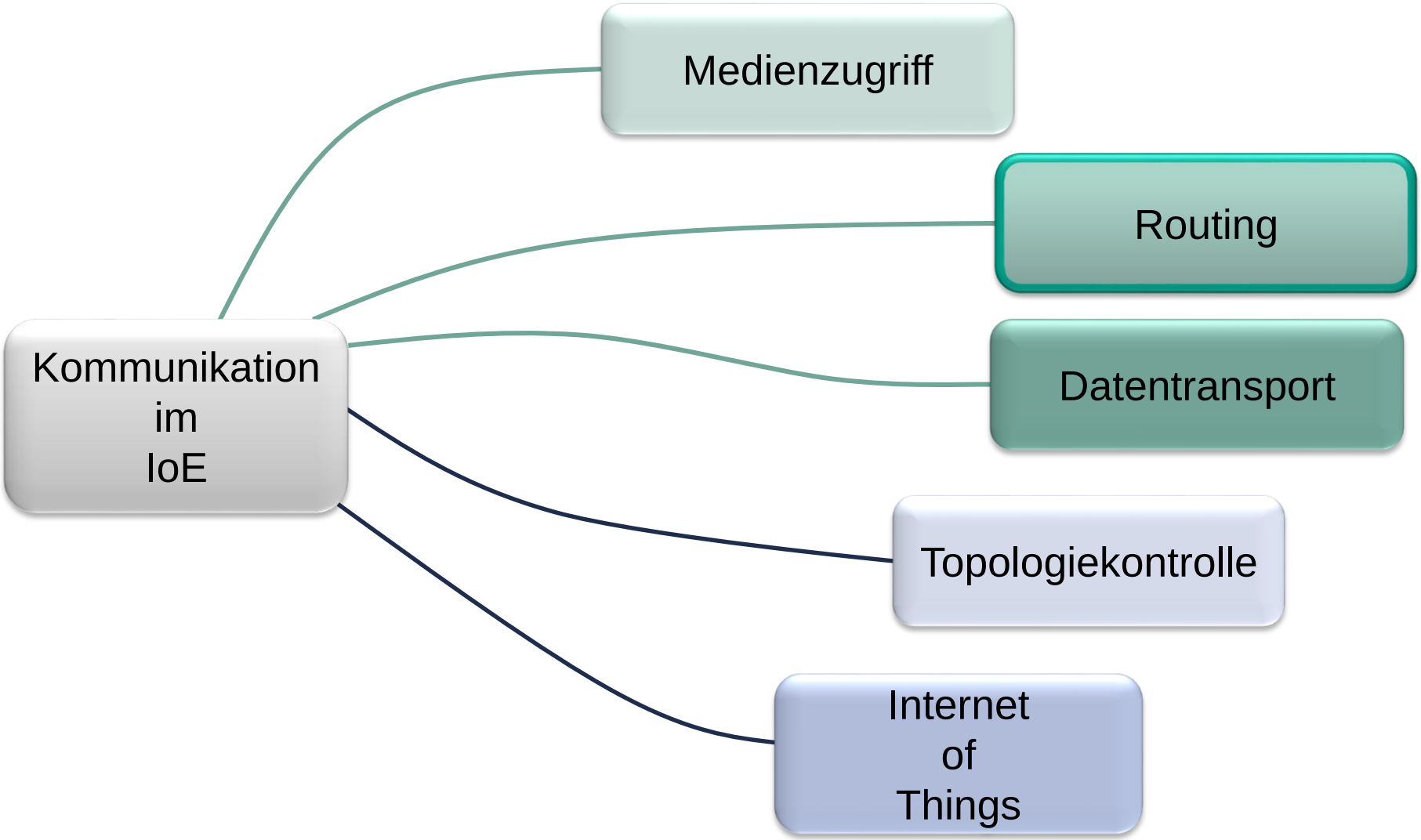
Vorlesung Internet of Everything
Wintersemester 2017/18

Kapitel 4.2 – Routing

Institut für Telematik, Prof. Zitterbart

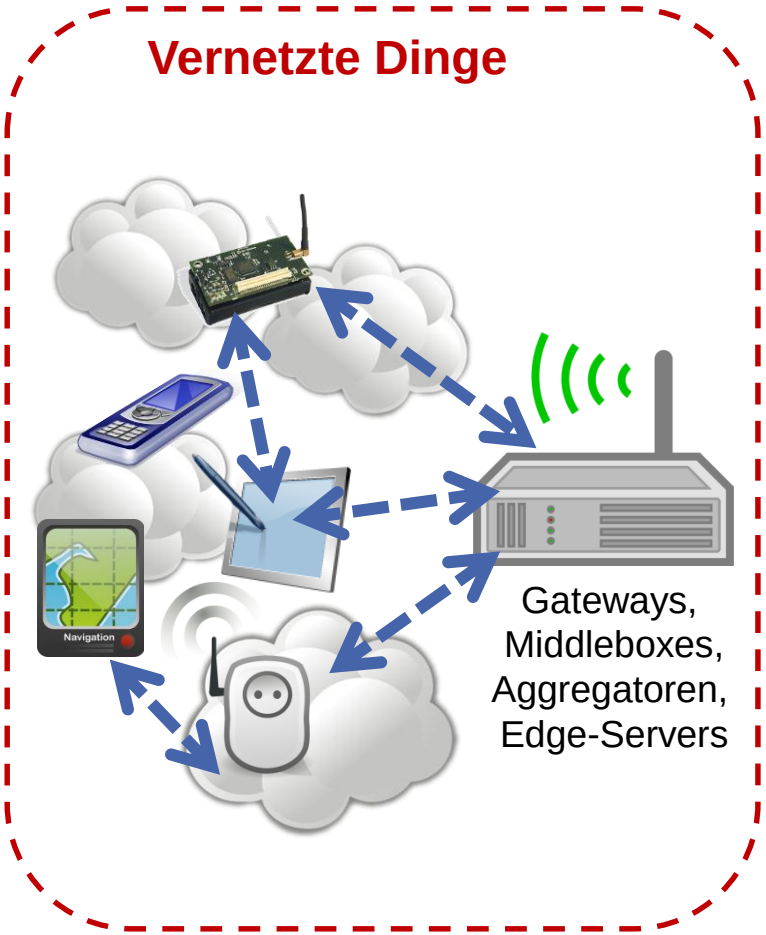

Einführung
Komponenten
Internet
of
Everything
Privatsphäre
Kommunikation
Sicherheit
Gastvorlesungen

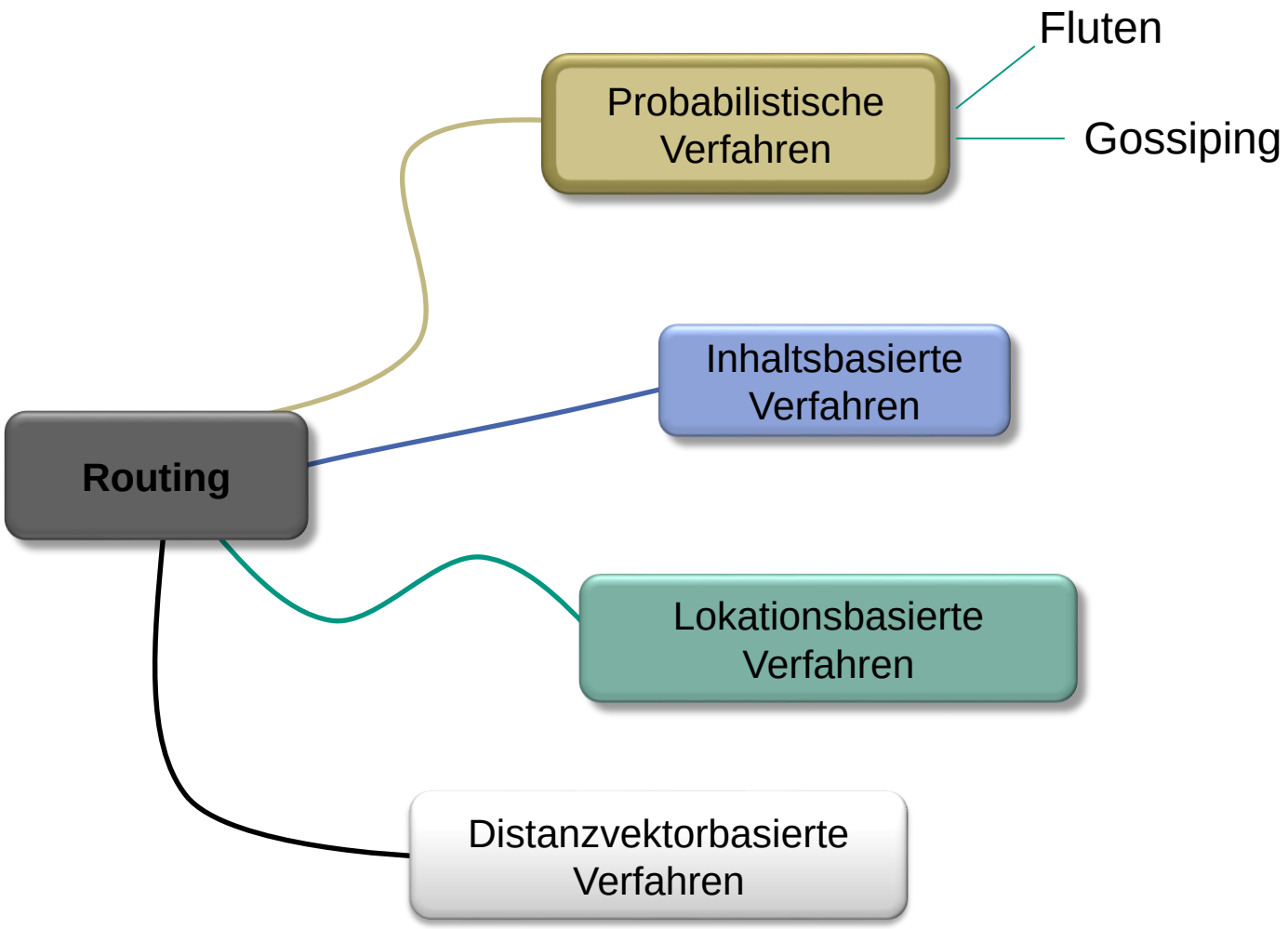
Kapitel 4.2: Routing



Motivation

- Anmerkung
  - Viele Arbeiten zunächst im Kontext drahtloser Sensornetze durchgeführt
    - … „Randbereich“ des IoE
- Annahmen für drahtlose Sensornetze
  - Adressierung anders
    - Globales Adressierungsschema evtl. wenig praktikabel
  - Typischer Datenfluss
    - Concast: Daten von vielen Quellen an eine Senke
    - Dissemination: Daten von einer Quelle an viele/alle Senken
    - Wenig Knoten-zu-Knoten Kommunikation
  - Potenzial für hohe Redundanz
    - Z.B. Temperaturwerte aus kleinem geogr. Bereich sind sehr ähnlich
  - Ressourcen sind stark limitiert
    - Energie, Speicherkapazität …



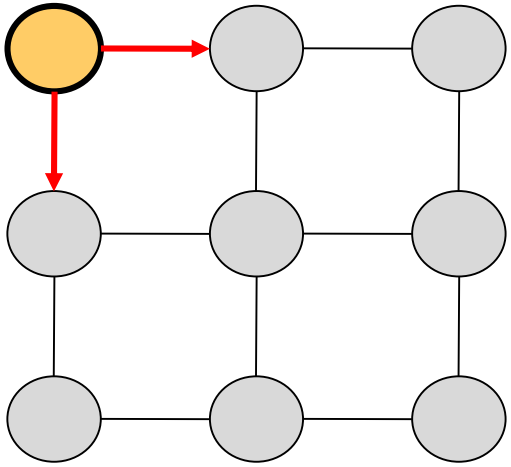

Fluten
Probabilistische Verfahren
Gossiping
Inhaltsbasierte Verfahren
Routing
Lokationsbasierte Verfahren
Distanzvektorbasierte Verfahren

Routingverfahren aus dem akademischen Bereich

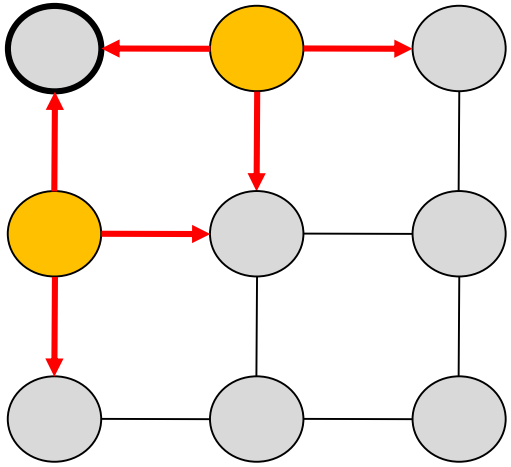
Fluten

- Vorgehensweise
  - Knoten sendet jede Dateneinheit per Broadcast an alle Nachbarn
    - Broadcast-MAC-Adresse
  - Empfangende Knoten leiten Dateneinheit per Broadcast an alle ihre Nachbarn weiter
- Eigenschaften
  - Dezentral und selbstorganisierend
  - Keine Routingtabellen (und deren Wartung) erforderlich
  - Hohe Netzbelastung
- *… durchaus häufig im Einsatz*
- Beispiel

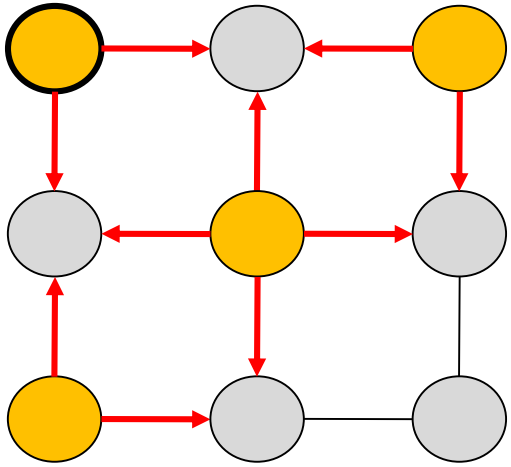
Schritt 1



Schritt 2



Schritt 3



Bewertung

- Vorteile
  - Keine Routenfindung erforderlich
  - Keine Wartung der Topologie erforderlich
  - Keine Routingtabelle, d.h. keine Zustandshaltung und -wartung erforderlich

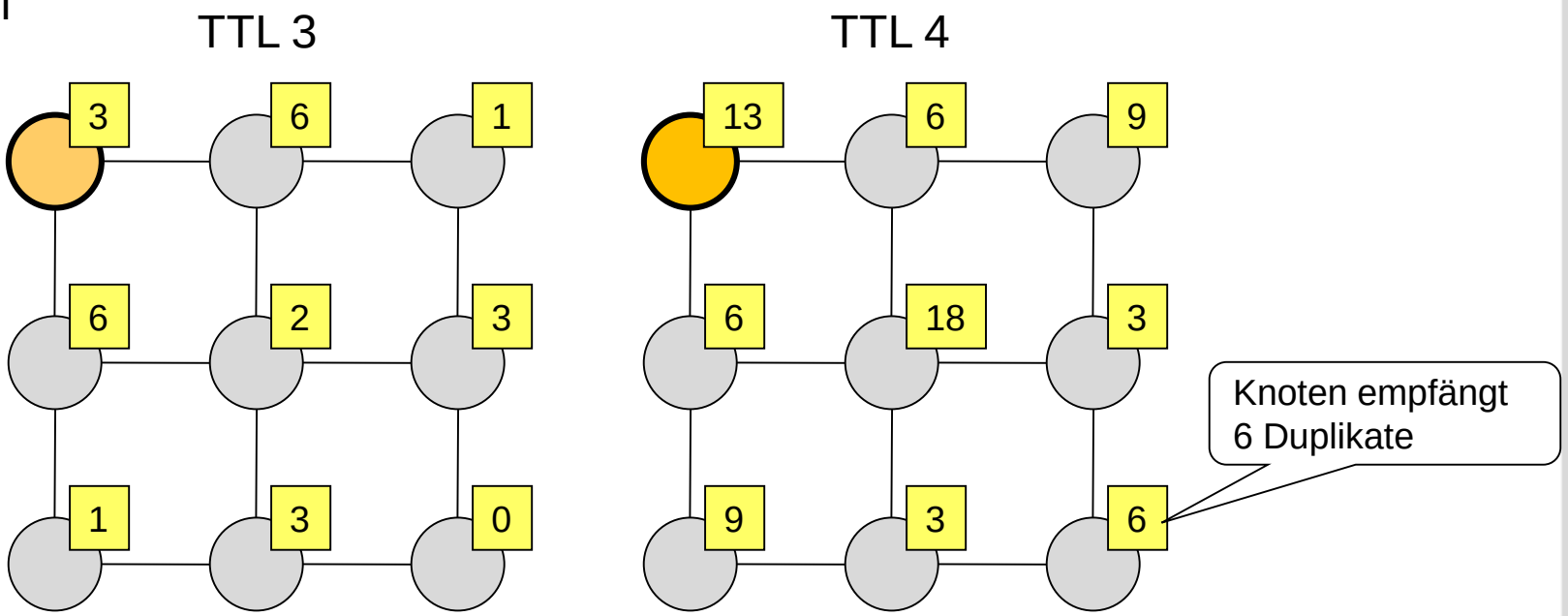
- Nachteile
  - Implosion: versenden duplizierter Dateneinheiten
  - Limitierte Ressourcen nicht berücksichtigt (z.B. Übertragungskapazität, Energie)
    - Hohe Netzbelastung, potenziell viele Kollisionen, potenziell hoher Energieverbrauch
  - Terminiert nicht: Dateneinheiten ‚kreisen' unendlich lange im Netz
  - Keine Zuverlässigkeit

Verbesserung Duplikatvermeidung

- Ziel
  - Sensorknoten leitet neu empfangene Dateneinheit genau einmal weiter
- Anforderungen
  - Alle Sensorknoten müssen Zustandshaltung über bereits weitergeleitete Dateneinheiten betreiben
  - Dateneinheiten müssen eindeutig identifizierbar sein
- Bewertung
  - Anforderungen sind in dezentralen, selbstorganisierenden WSANs nicht immer einfach zu erfüllen
    - Erkennen von duplizierten Dateneinheiten problematisch
    - Beschränkter Speicherplatz
  - Einsatz probabilistischer Verfahren zur Duplikaterkennung

Verbesserung begrenzte Reichweite (TTL)

- Vorgehensweise
  - Begrenzung der Reichweite durch Time-to-Live (TTL)
    - Wenn Dateneinheit maximalen Hopcount erreicht hat, wird sie verworfen
- Vorteile
  - Geringere Netzbelastung, weniger Kollisionen etc.
- Nachteile
  - Keine Zuverlässigkeit
- Beispiel



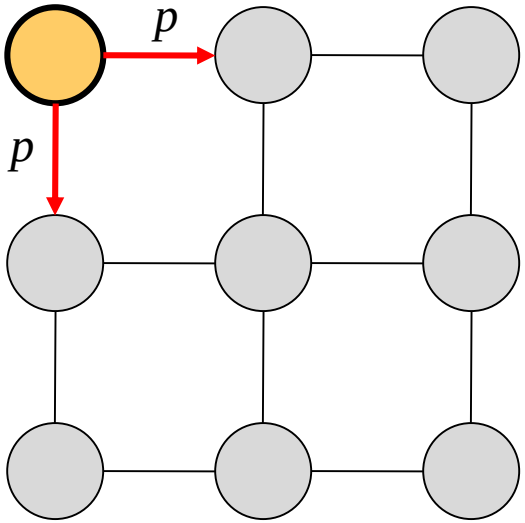
Gossiping

- Grundlegende Idee
  - Rumor Mongering: Ähnlich der Verbreitung von Gerüchten
    - Man erzählt Gerüchte einer gewissen Auswahl/Anzahl von Bekannten
  - Angewandt auf Weiterleitung von Dateneinheiten
    - Leite Dateneinheiten an zufällig ausgewählte Knoten weiter, bis „ausreichend viele“ Knoten die Dateneinheit erhalten haben

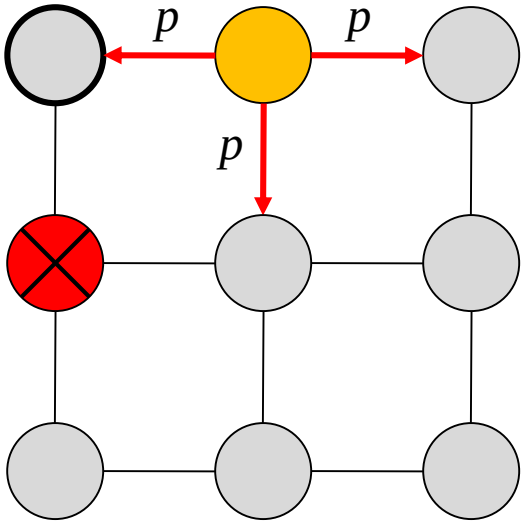
- Vorgehensweise
  - Knoten leitet Dateneinheit mit Wahrscheinlichkeit *p* an Nachbarn weiter
    - Broadcast auf MAC-Ebene erreicht alle 1-Hop-Nachbarn
  - Parameter *p*
    - Typisch: *p* zwischen 65% bis 75%
    - Bei $p < 65\%$ hohe Wahrscheinlichkeit, dass Dateneinheit verworfen wird bevor sie ihr Ziel erreicht

Beispiel

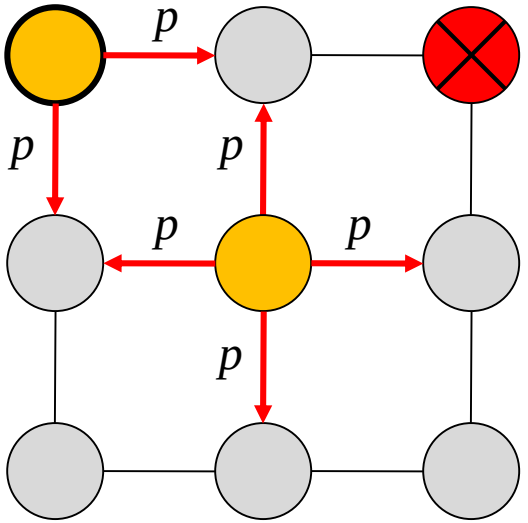
Schritt 1



Schritt 2

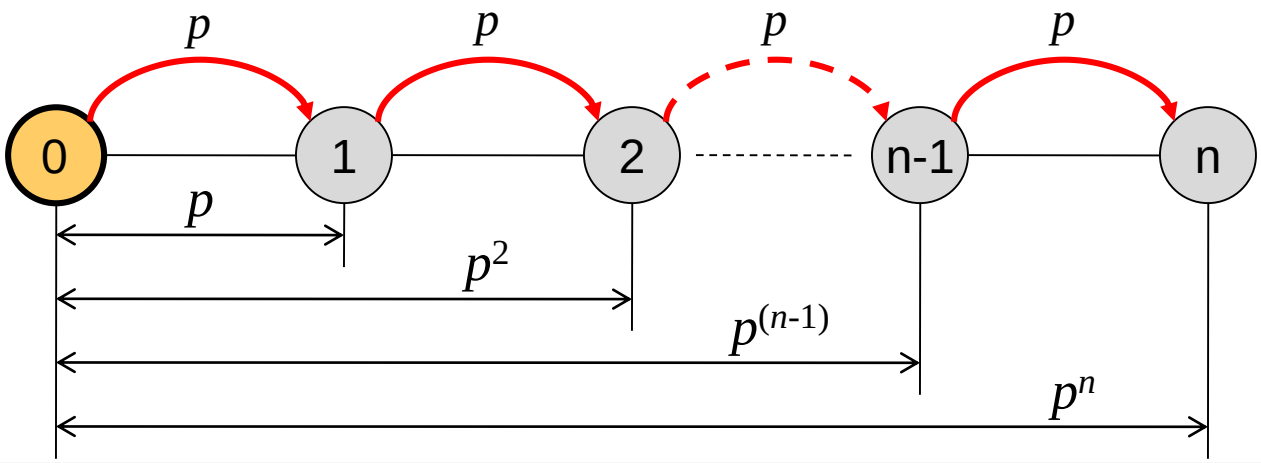


Schritt 3



Bewertung

- Vorteile
  - Keine Implosion von Dateneinheiten
  - Geringerer Overhead als Fluten
- Nachteile
  - Evtl. lange Übertragungszeit
    - Nachricht kann sehr „unsinnigen“ Pfad zwischen zwei Knoten nehmen
  - Keine Zuverlässigkeit
    - Es erhalten nicht alle Knoten im Netz die Dateneinheit
      - *… it is a “feature” not a bug*



Kombination aus Fluten und Gossiping

- Idee
  - Auf den *k* ersten Hops fluten, dann Gossiping mit Wahrscheinlichkeit *p*
  - Als GOSSIP(*p*, *k*) bezeichnet
- Parametrisierung
  - GOSSIP(1, *k*) → Fluten
  - GOSSIP(*p*, 0) → „Normales“ Gossiping
- Erfahrungswerte
  - Annahmen
    - Reguläre Netzwerktopologie, mittlere Größe (ca. 1000 Knoten)
    - Perfektes MAC-Protokoll (keine Datenverluste, keine Kollisionen)
  - Ergebnisse
    - Parameter *p*=0,72 und *k*=4
    - Nahezu alle Knoten erhalten die Dateneinheiten

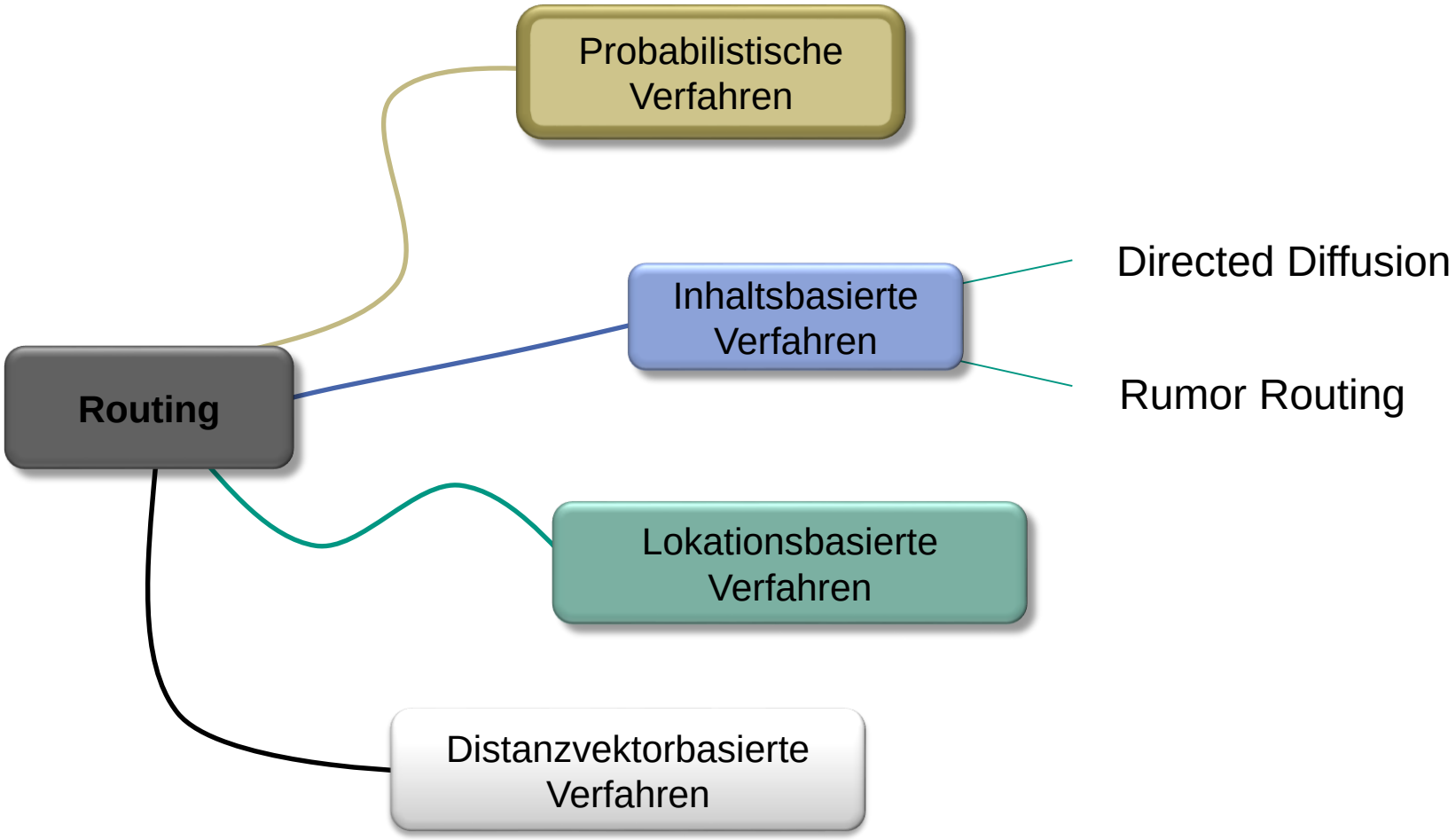
[HaHL02]

Verbesserung: variierendes *p*

- Idee
  - Wahrscheinlichkeit *p* für Weiterleitung steigt, je näher eine Dateneinheit ihrem Ziel kommt
- Voraussetzung
  - Jeder Knoten kennt Distanz zum Ziel
    - Ziel kann z.B. erfolgreich empfangene Dateneinheiten quittieren und damit die Distanz zum Ziel bekannt machen
- Parameter
  - p_{R_i}: Gossiping Wahrscheinlichkeit für eine Dateneinheit auf dem *i*-ten System R_i in Richtung Ziel
  - *k*: Faktor, der den Verlauf der Wahrscheinlichkeit bestimmt
  - Berechnung von p_{Ri}

$$p_{Ri} = \begin{cases} (1+k)\,p_{Ri-1} & \text{näher zum Ziel} \\ (1-k)\,p_{Ri-1} & \text{weiter weg vom Ziel} \\ p_{Ri-1} & \text{gleichblei bend oder unbestimmt} \end{cases}$$

[SelfOrg]


Probabilistische Verfahren
Inhaltsbasierte Verfahren
Directed Diffusion
Rumor Routing
Routing
Lokationsbasierte Verfahren
Distanzvektorbasierte Verfahren

Directed Diffusion

- Datenzentrisches Verfahren
  - Klassische Adressen spielen keine Rolle, Daten stehen im Mittelpunkt
  - Basiert auf Fluten, ist selbstorganisierend
  - Findet Pfade von (Ereignis-)Quellen zu einer Senke (Interessent)
  - Unterstützt Aggregation von Daten

- Grundlegende Idee
  - Daten werden durch Attribut-Wert-Paar benannt
  - Beobachtungsaufgabe wird als Interesse im Netz verbreitet („Diffusion“)
    - Gradient wird hierdurch in den Knoten im Netz etabliert
  - Gemessene Ereignisse folgen mehreren Pfaden zum Interessenten (Senke) („directed“)
    - Verstärkung einiger weniger Pfade

[Inta00]

- Vorgestellt von Intanagonwiwat, Govindan, Estrin, Heidemann und Silva, University of California, Los Angeles, 2000

Directed Diffusion

- Besonders geeignete Szenarien
  - Eine Datenanfrage – mehrfache/regelmäßige Antworten
    - Z.B. regelmäßige Temperaturmessungen
  - Etablierung einer “Infrastruktur” zahlt sich eventuell aus
    - Interessen (Queries) „teuer“, Ereignisse „billig“ hinsichtlich Kommunikationsaufwand
- Sensorknoten, die eine Datenanfrage beantworten können sind unbekannt
- Mehrere Sensorknoten können dieselben Daten anfragen

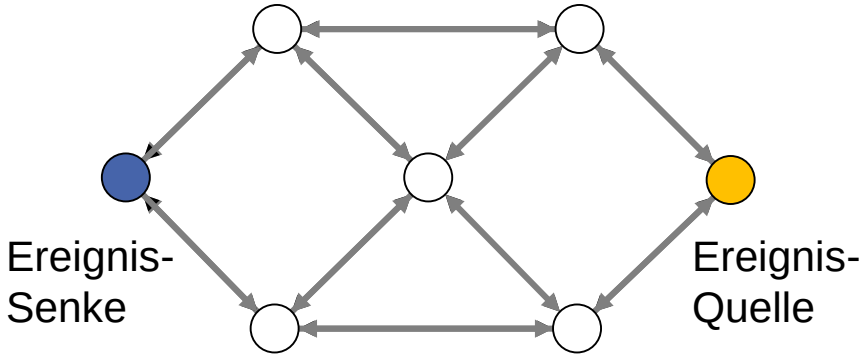
Beschreibung der Daten

- Interesse wird geäußert (Query), z.B.
  - ```
    type = temperature                  // sammle Temperaturwerte
    interval = 20 s                     // ein Messwert alle 20 Sekunden
    duration = 1 h                      // sende Werte für eine Stunde
    location = first floor              // nur Messwerte aus 1. Stock
    precision = 0.5                     // minimale Genauigkeit
    ```
- Übertragung von Daten
  - ```
    type = temperature                  // Wert ist vom Typ temperature
    location = first floor, room101     // Lokation des Sensors
    value = 20C                         // gemessener Wert
    precision = 0.7                     // Genauigkeit des Sensorwerts
    timestamp = 03:20:45                // Zeit der Messung
    ```

Basisverfahren (1)

- Senken äußern Interesse an Daten (Query)
  - Interesse wird periodisch erneuert. Soft-state und selbstorganisierend
    - Zeitstempel wird monoton erhöht
  - Interesse wird im Netz geflutet
    - Weitere Regeln können angewandt werden, z.B.
      - Unterdrücken, wenn ähnliche Interessensbekundungen kürzlich gesehen wurden

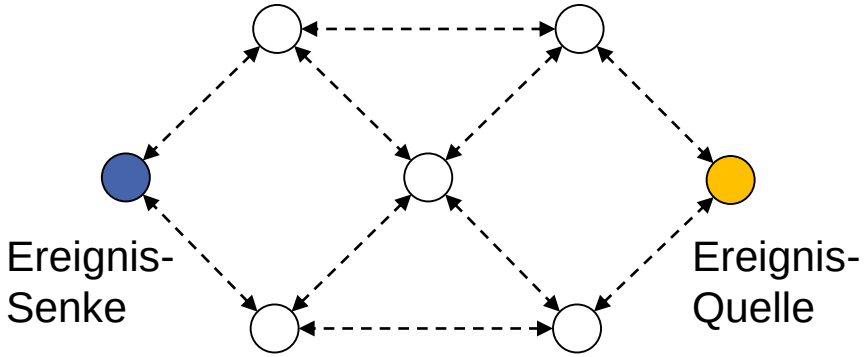
- Beispiel



Basisverfahren (2)

- Aufbau von Pfaden zur Datenquelle
  - Speichern der Richtung aus der Interesse kam
    - Als Gradient bezeichnet
      - Enthält auch noch Information über Datenrate
    - Enthält keinen Bezug zu Quelle oder Senke
    - Konsequenz aus Fluten: Bidirektionale Gradienten (aber nicht symmetrisch!)
  - Zu einer Query können mehrere Gradienten existieren

- Beispiel

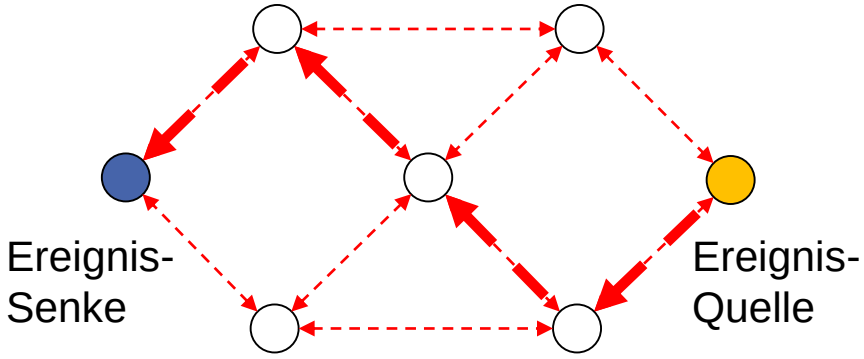


Basisverfahren (3)

- Daten werden auf Rückwärtspfaden gesendet
  - Also von der Ereignis-Quelle entlang der Gradienten Richtung Senke
  - Auftretende Ereignisse (z.B. Messung eines Temperaturwerts)

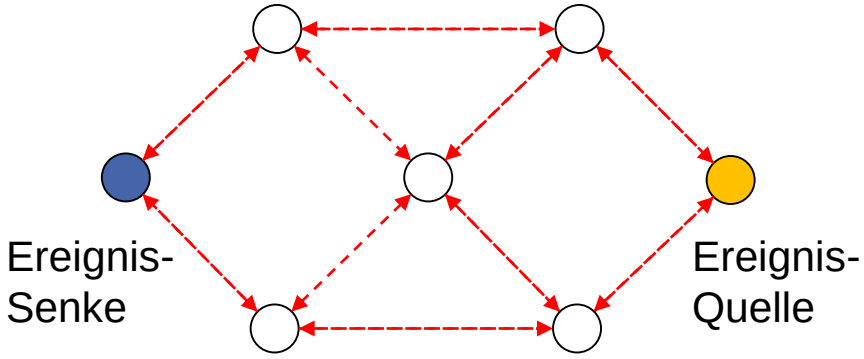
  - Weiterleitung entspricht zunächst Fluten
  - → Verbesserung: Etablierung eines bzw. einiger guten Pfade

- Beispiel



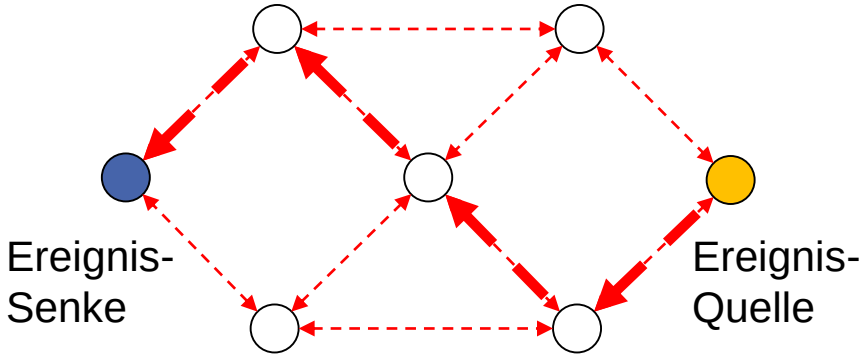
Gradientenverstärkung (1)

- Suche nach Ereignisquellen
  - Anfangs werden Gradienten mit niedriger Datenrate etabliert
    - Niedrige “Stärke”
    - Mehrere Gradienten pro Knoten und Interesse möglich
  - Weiterleitung der Ereignisse erfolgt an alle Gradienten
    - Weiterleitung auf mehreren Pfaden, mehrfaches Senden von Daten



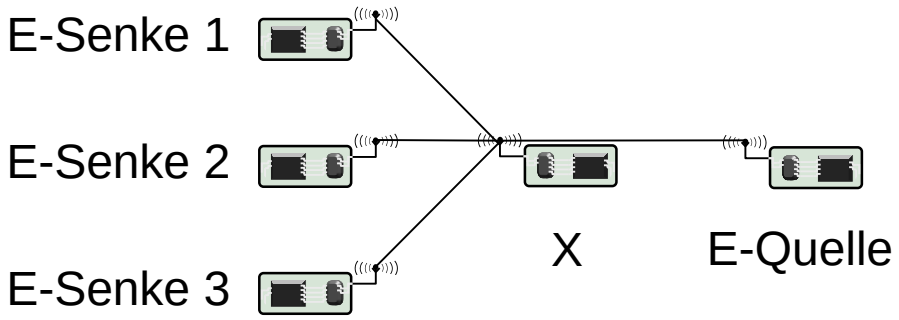
Gradientenverstärkung (2)

- Verstärkung, falls Ereignis-Quelle gefunden
  - Senke (Interessent) startet Reinforcement-Phase
    - Datenrate zu diesem Interesse wird erhöht, d.h. “Stärke” des Gradienten steigt
    - Ist angeforderte „Stärke“ höher als derzeit verfügbare
      - Weiterleiten zu einem Nachbarn in Richtung Ereignis-Quelle
  - → Etablierung eines bzw. einiger Pfade zwischen Ereignis-Quelle und Senke
    - Weiterleitung erfolgt nur noch auf diesen Pfaden
    - Pfad mit geringer Verzögerung wird gewählt
  - → Anpassung an Änderungen der Datenquellen, Topologie und Senken möglich
    - Andere Metriken möglich, um etwa Stabilität zu erhöhen, z.B.
      - Nachbar, von dem die meisten Ereignisse empfangen wurden
      - Nachbar, von dem konstant Ereignisse empfangen werden

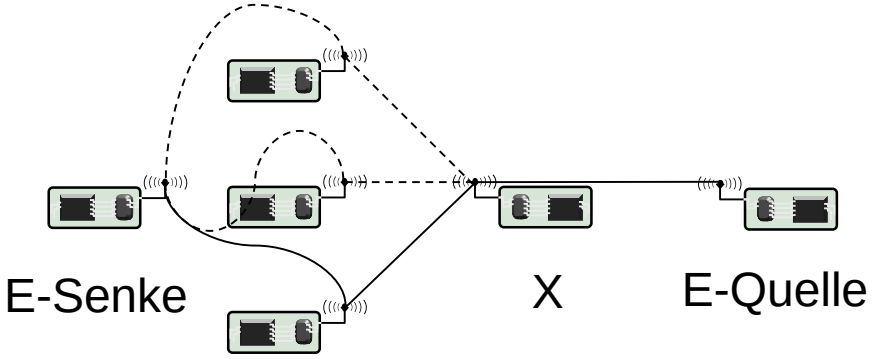


Gradientenverstärkung (3)

- Gradientenverstärkung löst folgendes Problem
  - System X kennt drei Ereignissenken
  - Aufbau von einem oder drei Verteilbäumen?



- Lösung
  - Gradientenverstärkung
    - Oben: Drei Senken verstärken drei Gradienten
    - Unten: Senke verstärkt genau einen Gradienten



Directed Diffusion vs. „traditionelles“ Routing

- Directed Diffusion
  - Reaktives Routing
    - Pfad wird als Reaktion auf Interessensbekundung etabliert
  - Pfadaufbau in zwei Phasen
    - Phase 1: redundantes Senden von Daten
    - Phase 2: Reduktion der Pfade durch Gradientenverstärkung

    → Datentransfer in beiden Phasen
  - Datencache
    - Duplikaterkennung → Vermeidung von Schleifen
    - Anpassung der Datenrate (ggf. Interpolation, Aggregation)
  - Nur lokale Kommunikation
    - Keinerlei Ende-zu-Ende Kommunikation
- „Traditionelles“ Routing
  - Aufbau eines schleifenfreien Pfads, dann erst Datentransfer
  - Ende-zu-Ende Kommunikation

Evaluierung (1)

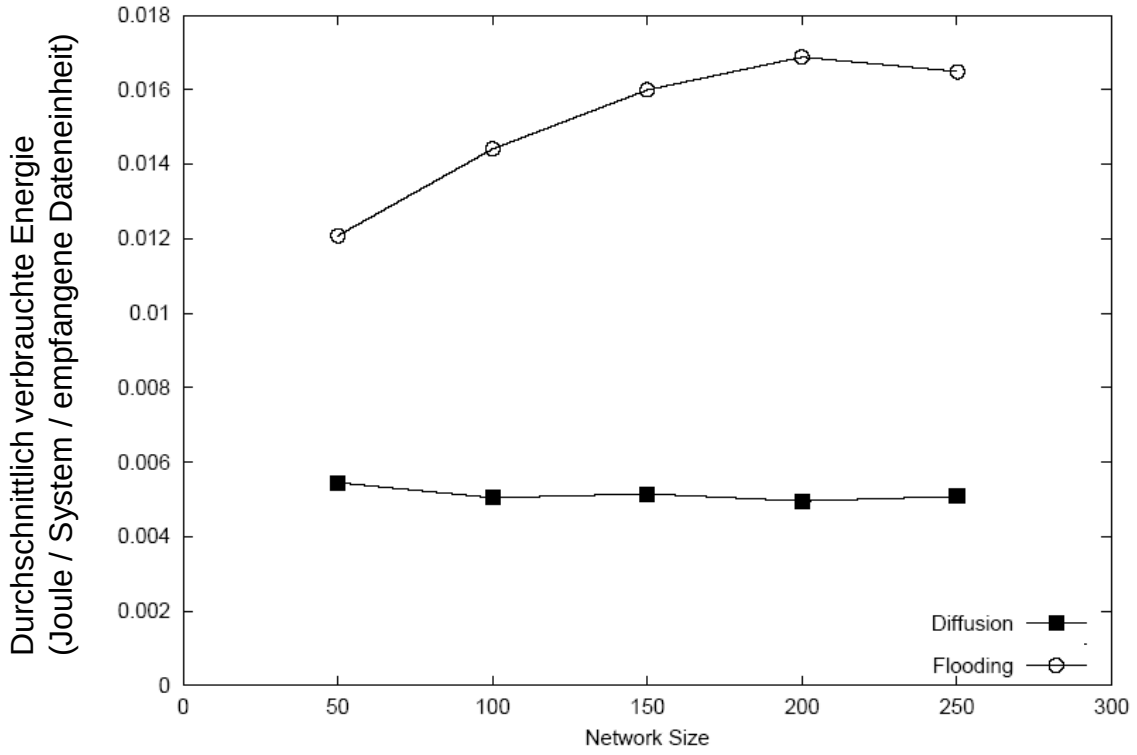
- Vergleich mit
  - Fluten
    - Repräsentiert „untere Schranke“ für die Leistungsfähigkeit
- Simulative Untersuchungen
  - Sensornetz war nicht ausgelastet
  - Sensorknoten zufällig auf quadratischem Feld platziert
    - 5 Ereignis-Quellen und 5 Senken
  - Knotenausfälle
    - feste Anzahl an Sensorknoten wurde für 30 Sekunden ausgeschaltet


[Inta00]


[InGE03]

Evaluierung (2)

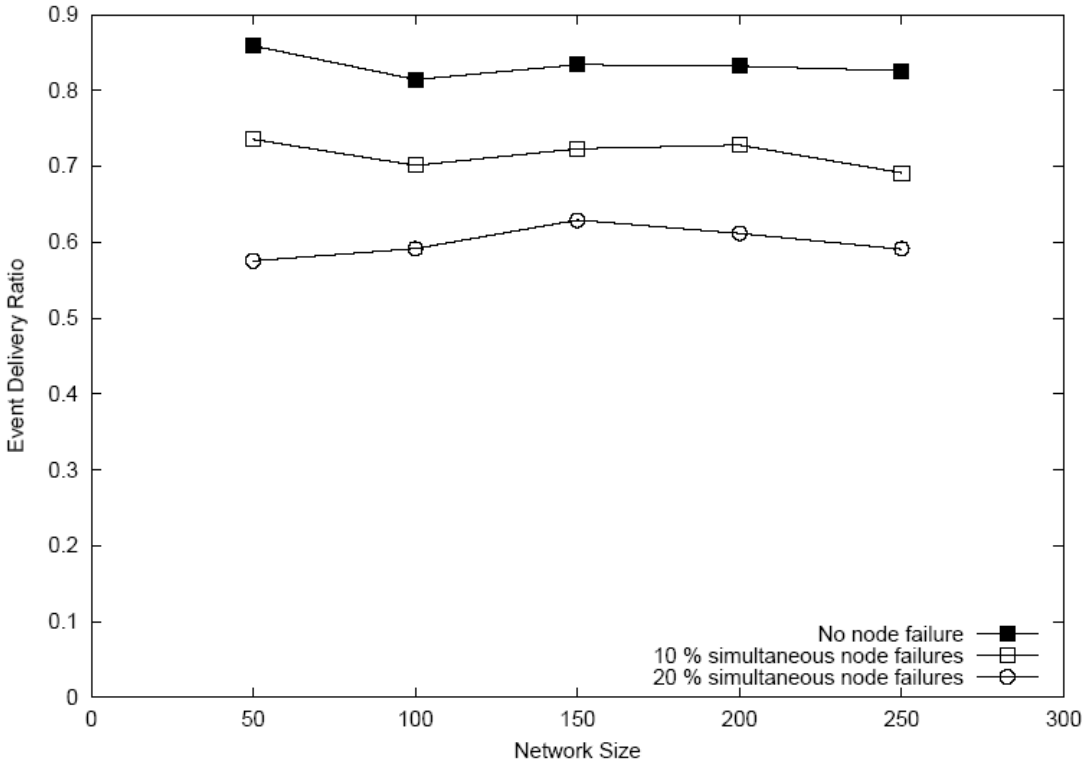
- Mittlere verbrauchte Energie
  - Verbrauchte Energie pro Knoten / Anzahl verschiedener Ereignisse bei der Senke
  - Directed Diffusion deutlich besser als Fluten



[Inta00]

Evaluierung (3)

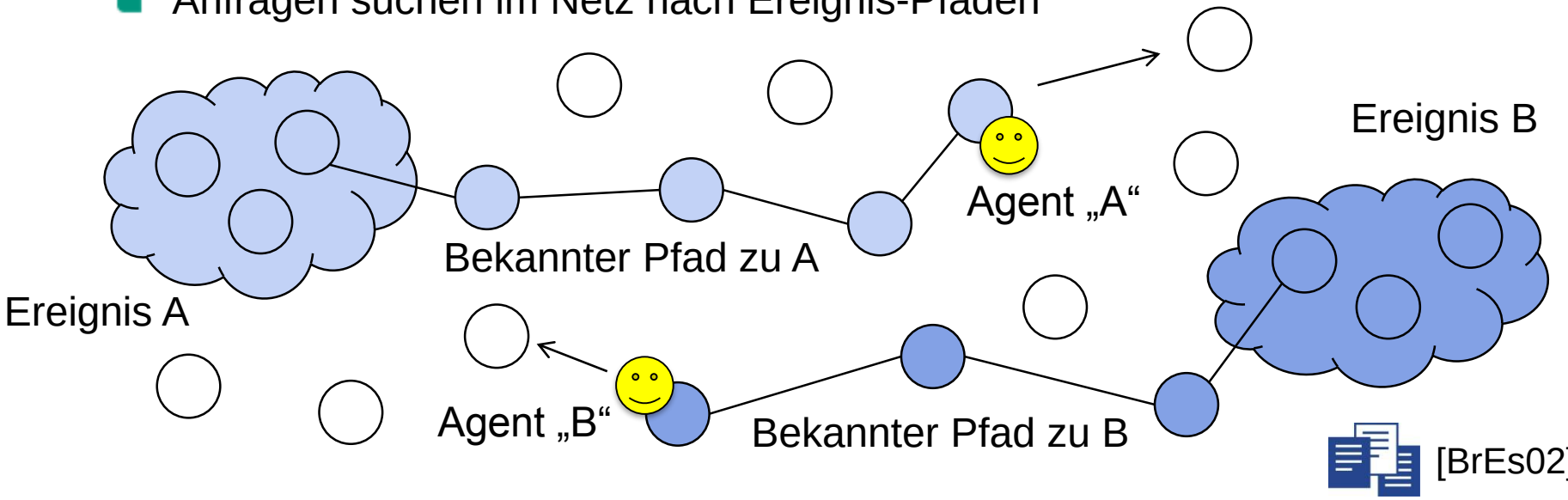
- Auslieferungsrate bei Knotenausfällen
  - Auslieferungsrate := $\frac{\text{Anzahl unterschiedlicher empfangener Ereignisse}}{\text{Anzahl ursprünglich versendeter Ereignisse}}$
  - Zu jeder Zeit waren 10% oder 20% der Knoten nicht operabel
  - Auslieferungsrate wird verringert, bleibt aber relativ stabil



[InGE03]

Rumor Routing

- Datenzentrischer Ansatz, wie bei Directed Diffusion
  - Initiative geht von Ereignis-Quelle *und* von Ereignis-Senke aus
  - Kompromiss zwischen Fluten von Anfragen und Fluten von Ereignissen
- Agentenbasierter Ansatz
  - Ereignis-Agenten etablieren Pfade zu Ereignissen
    - Ereignis ist Abstraktion, identifiziert Menge von Sensorwerten etc., ist lokales Phänomen (z.B. Temperatur im Raum)
  - Anfragen suchen im Netz nach Ereignis-Pfaden



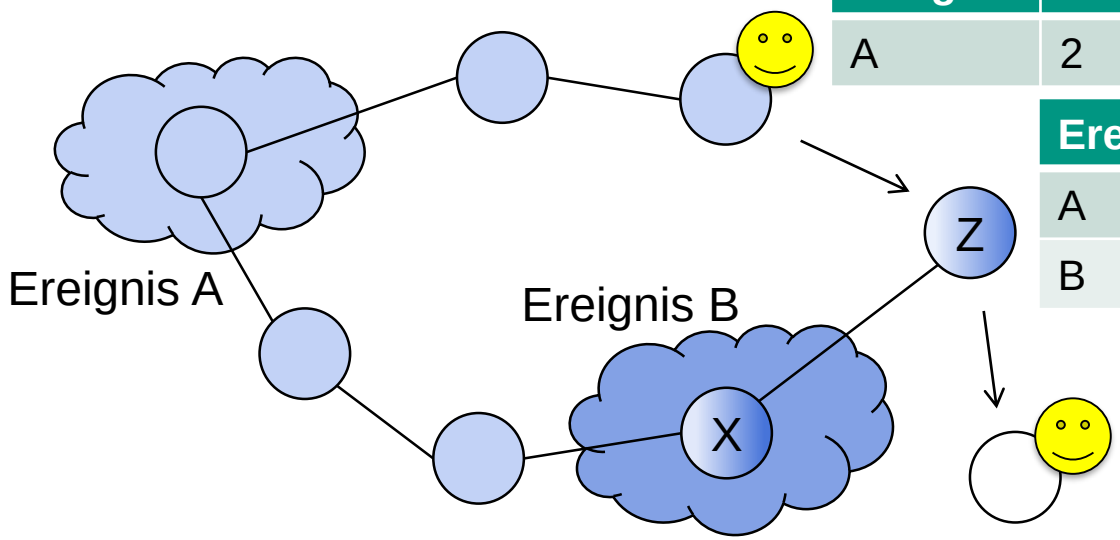
[BrEs02]

- Vorgestellt 2002 von Braginsky & Estrin, University of California, Los Angeles

Ereignis-Agenten

- Besitzen Information über Ereignisse
- Werden bei neuem Ereignis mit gew. Wahrscheinlichkeit erzeugt
- Wandern (als langlebige Dateneinheit) durch das Netz
  - Können auf ihrem Weg Informationen lernen
  - Besitzen max. Lebendauer (= Anzahl von Hops)
- Hinterlassen Pfadinformation in den Knoten

- *In Anlehnung an biologische Systeme → Ameisen*



Ereignis	Distanz
A	2

Ereignis	Distanz	Richtung
A	4 → 3	Y
B	1	X

Ereignis	Distanz
A	4
B	2

Beteiligte Sensorknoten

- Benötigen Informationen über Nachbarn und Ereignisse
  - Nachbarschaftsinformation
    - Z.B. durch periodische Hello-Dateneinheiten
  - Ereignistabelle
    - Weiterleitungsinformation zu bekannten Ereignissen
    - Updates durch ankommende Ereignis-Agenten
    - Einträge sollten limitierte Lebensdauer haben

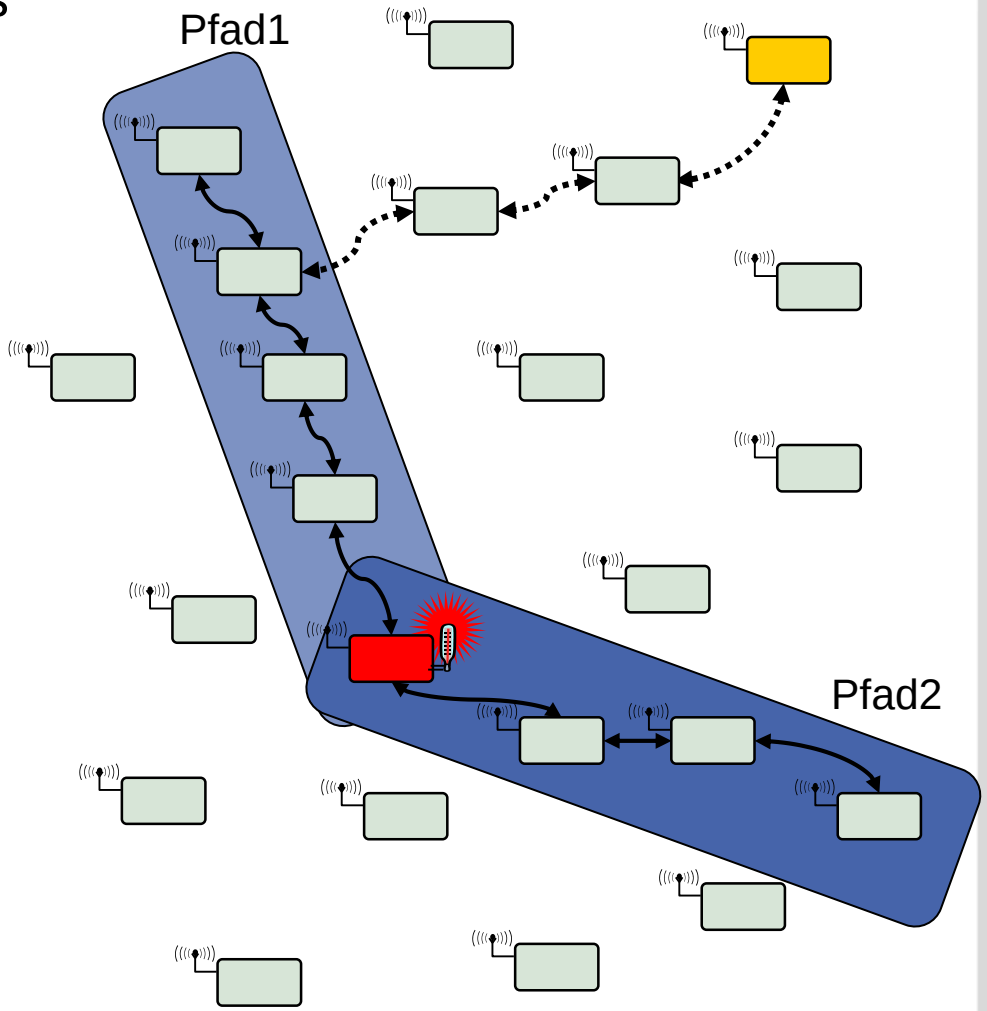
Nutzung von Random Walk

- Einfachste Form von Random Walk
  - Leite Dateneinheit an zufällig gewählten Nachbarn per Unicast weiter
    - Unicast-MAC-Adressen
  - Alternative Form
    - Leite mehrere Kopien der Dateneinheit parallel an zufällige Nachbarn weiter
  - *… analog zur Verbreitung von Gerüchten*

- Random Walk sowohl für Anfragen (Such-Agenten) als auch für Ereignis-Agenten
  - Trifft Anfrage auf Ereignis-Pfad, dann gezieltes Weiterleiten
    - Mehrere Pfade möglich

Beispiel

- Roter Knoten beobachtet Ereignis
  - Sendet zwei Ereignis-Agenten
  - Agenten installieren Routing-information über Ereignis
    - Ereignis-Pfad1 und Ereignis-Pfad2
- Gelber Knoten fragt nach einem Ereignis
  - Sendet Such-Agenten aus
  - Agent kreuzt bestehenden Ereignis-Pfad
    - Ereignis-Pfad1 im Beispiel
  - Pfad zum Ereignis ist ab jetzt bekannt
    - Ereignis-Quelle – erster Knoten auf Ereignis-Pfad 1
    - Vom „Kreuzungspunkt“ zur Ereignis-Quelle



Nochmal zum Prinzip

- Funktioniert, weil
  - Wahrscheinlichkeit hoch, dass sich in einem Rechteck zwei zufällige Pfade schneiden
    - Wahrscheinlichkeit von 69%, dass sich zwei Pfade schneiden
    - Wahrscheinlichkeit von 99,7% einen von 5 Pfaden zu schneiden

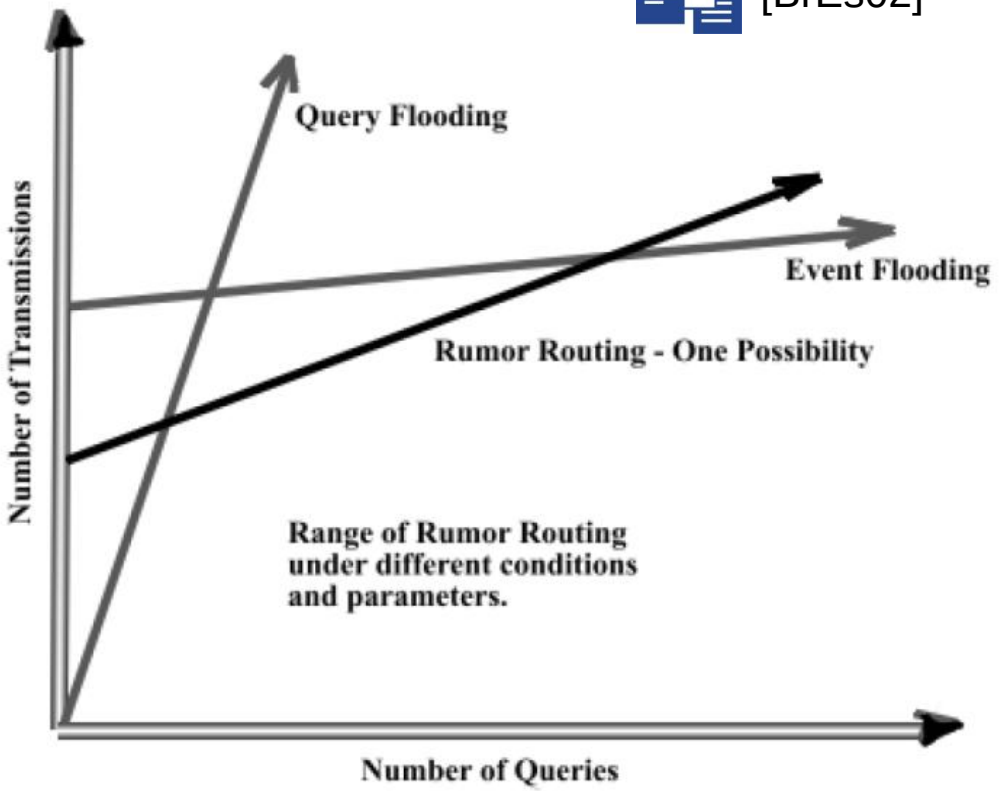
  → Trade-off zwischen Erfolgswahrscheinlichkeit und Energieverbrauch möglich
    - Etablierung mehrerer Pfade benötigt mehr Energie

- Voraussetzungen
  - Nachbarschaften müssen bekannt sein
    - Z.B. periodische Hello-Dateneinheiten
  - Ereignis-Information muss auf Pfad gespeichert werden
    - Ereignis-Information sollte mit Timeout versehen werden
      - Soft-State und selbstorganisierend

Leistungsfähigkeit

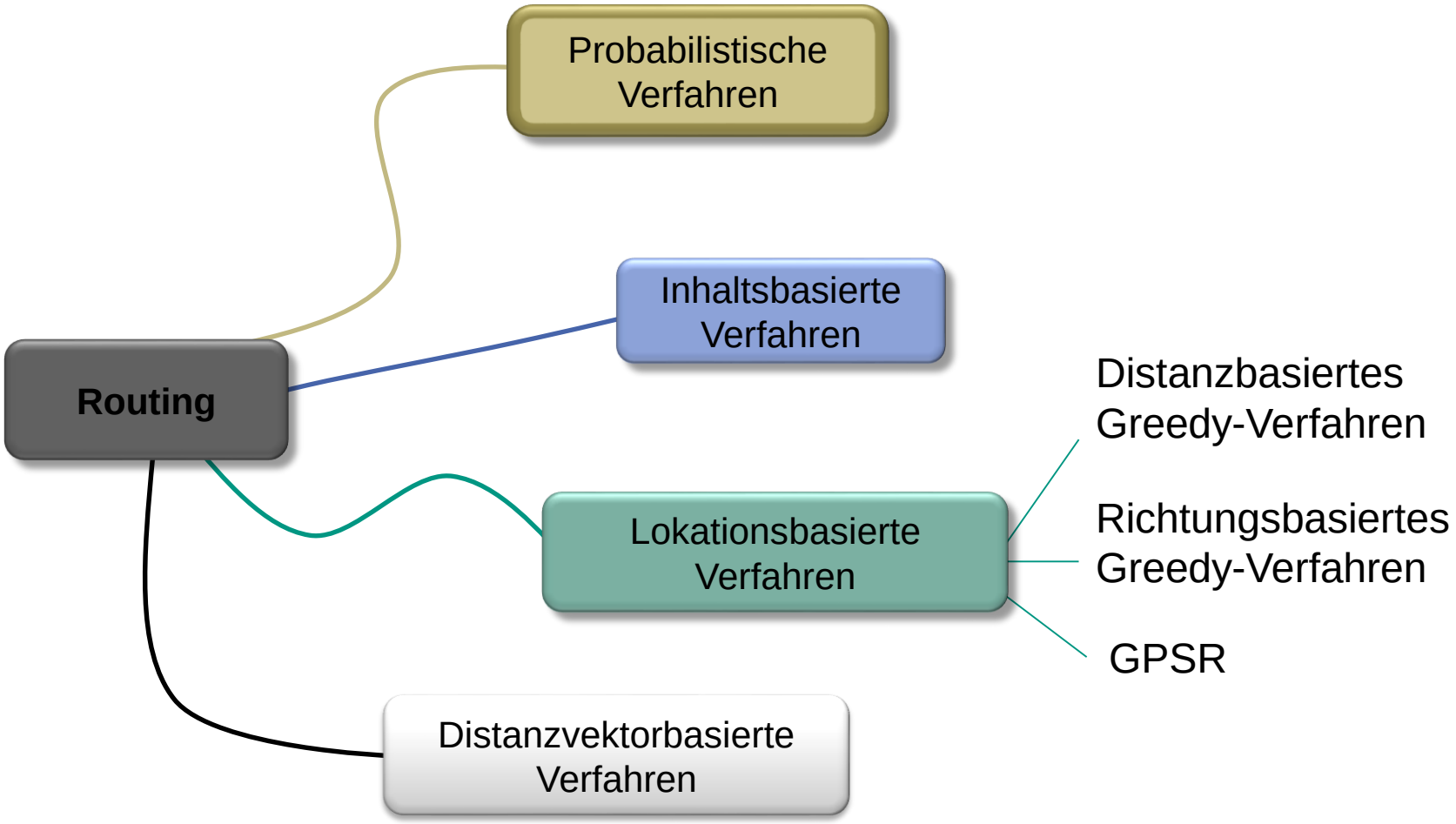
- Was ist besser: Fluten oder Rumor Routing?
  - Metrik: Anzahl der Sendevorgänge als Maß für Energieverbrauch
- Abwägung zwischen
  - Finde kürzesten Pfad durch Fluten
  - Finde etwas längeren Pfad durch Rumor Routing
- Wann lohnt sich Rumor Routing?
  - Anfragen fluten
    - Wenn viele Events, wenige Anfragen
  - Events fluten
    - Gradienten zur Eventquelle aufbauen
    - Anfragen mit geringen Kosten möglich
  - Rumor Routing
    - Für die Fälle dazwischen

[BrEs02]



Daten- vs. Adressorientiertes Routing

	Adressorientiert	Datenorientiert
Routingansatz	Pfad wird anhand Zieladresse bestimmt	Ziel wird anhand des Inhalts einer Dateneinheit bestimmt
Voraussetzung	Global eindeutige Adressen	Vordefinierte Formate inkl. Semantik
Routing-verfahren	Proaktives oder reaktives Routing	(probabilistisches) Fluten oder Interesse-basiertes Reverse Routing
Vorteile	Typischerweise geringe Verzögerung beim Verbindungsaufbau und bei Datenverteilung	Keine Adressinformation erforderlich Redundanz
Nachteile	Global einheitliche Adressen erforderlich	Aufwand für eine einzelne Übertragung erhöht sich


Probabilistische Verfahren
Inhaltsbasierte Verfahren
Routing
Lokationsbasierte Verfahren
Distanzbasiertes Greedy-Verfahren
Richtungsbasiertes Greedy-Verfahren
GPSR
Distanzvektorbasierte Verfahren

Überblick Lokationsbasierte Verfahren

- Motivation
  - In vielen Anwendungen sollen Orte/Regionen adressiert werden, z.B.
    - Irgendein System innerhalb einer Region
    - Das System mit dem geringsten Abstand zu einem Punkt
    - Systeme, die weiter als X Meter entfernt sind
  - Informationen über Position der Quell-, Ziel-, Zwischensysteme kann Routing unterstützen
    - Evtl. Abbildung zwischen Systemname und Position notwendig
    - Evtl. kann auf Routingtabelle verzichtet werden
    - Positionsangaben können Routingalgorithmen vereinfachen
      - Richtung ist vorhanden
- Vorteil
  - Falls Position äquivalent zu Adressen gesehen werden
    - Implizite Ordnung auf Adressen
    - „Geeignet“ für Greedy Strategien

Distanzbasiertes Greedy Verfahren

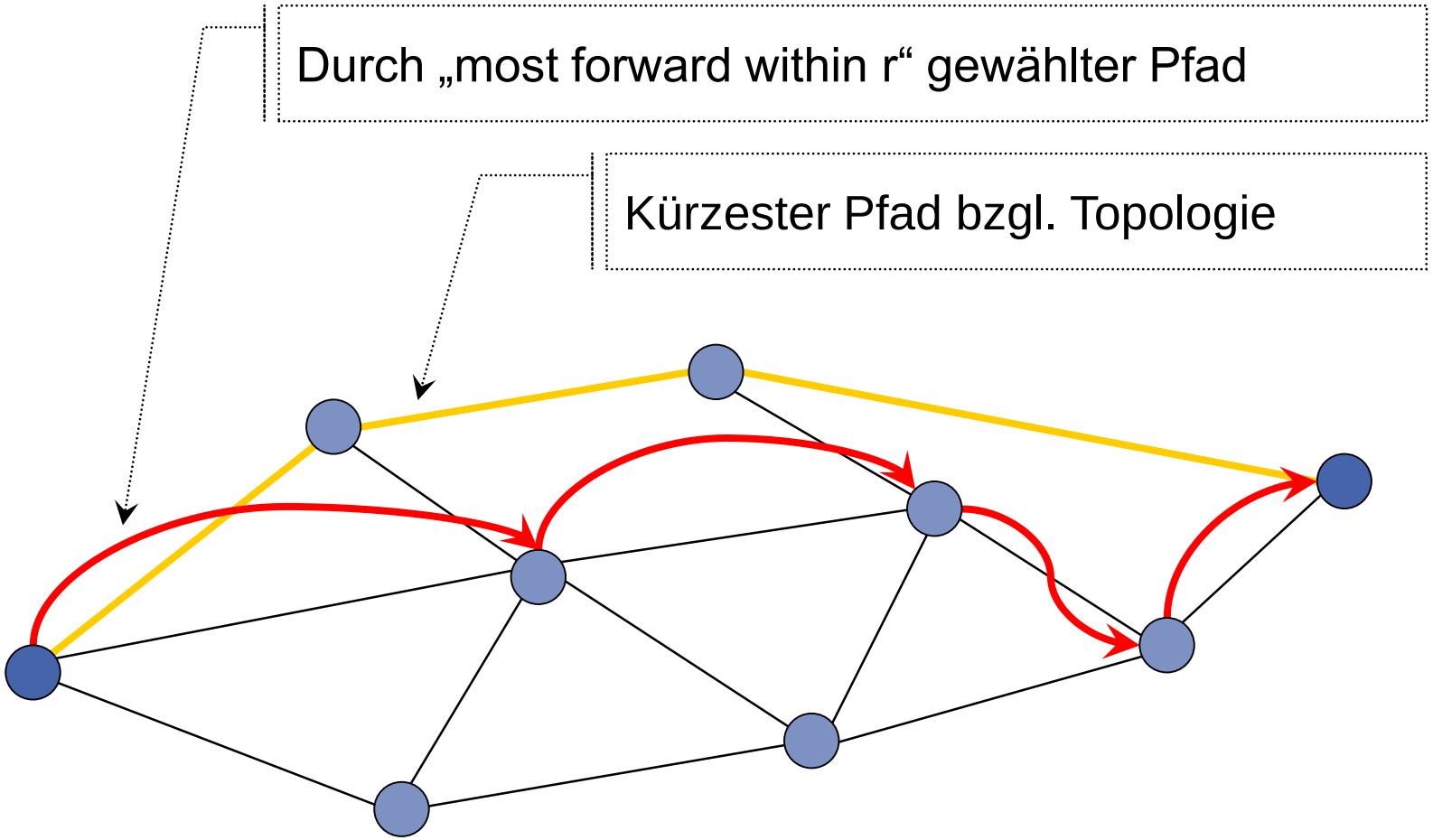
- Ziel
  - Dateneinheit zu einer bekannten Position übertragen
- Annahme
  - Jedes System kennt die eigene Position

- Strategie: „most forward within r“
  - r bezeichnet den Kommunikationsradius → Nachbarschaft
  - Greedy-Verfahren
    - Leite Dateneinheit an Nachbar-System weiter, das dem Ziel am nächsten ist
      - $nexthop(v) = \arg min_{u \in Nachbarn(v)}\{dist(u, Ziel)\}$

- Bewertung
  - \+ Strategie ist garantiert frei von Schleifen
  - − Topologie bleibt unbeachtet
    - Es wird nicht immer der topologisch kürzeste Pfad gefunden
  - − Systeme am Rand der Übertragungsreichweite bevorzugt
    - Pfade sind instabil, da Kommunikationswahrscheinlichkeit mit der Distanz abnimmt

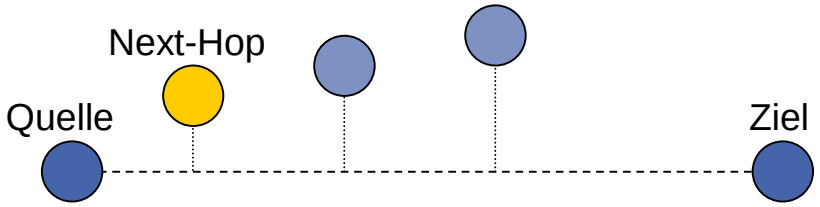
[TaKl84]

Beispiel

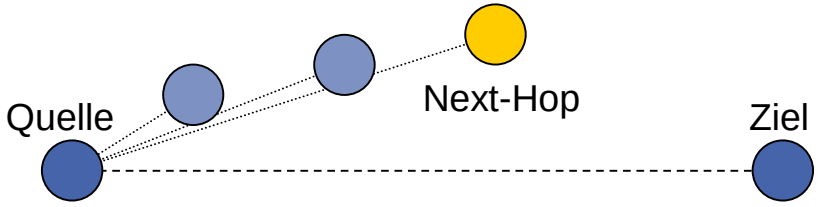

Durch „most forward within r“ gewählter Pfad
Kürzester Pfad bzgl. Topologie

Richtungsbasiertes Greedy Verfahren

- Idee: Wähle Next-Hop System *möglichst nahe* an der idealen Richtung zum Ziel
  - Ideale Richtung = Verbindungsgerade
- Zwei sinnvolle Metriken
  - Minimaler Abstand zur Verbindungsgerade



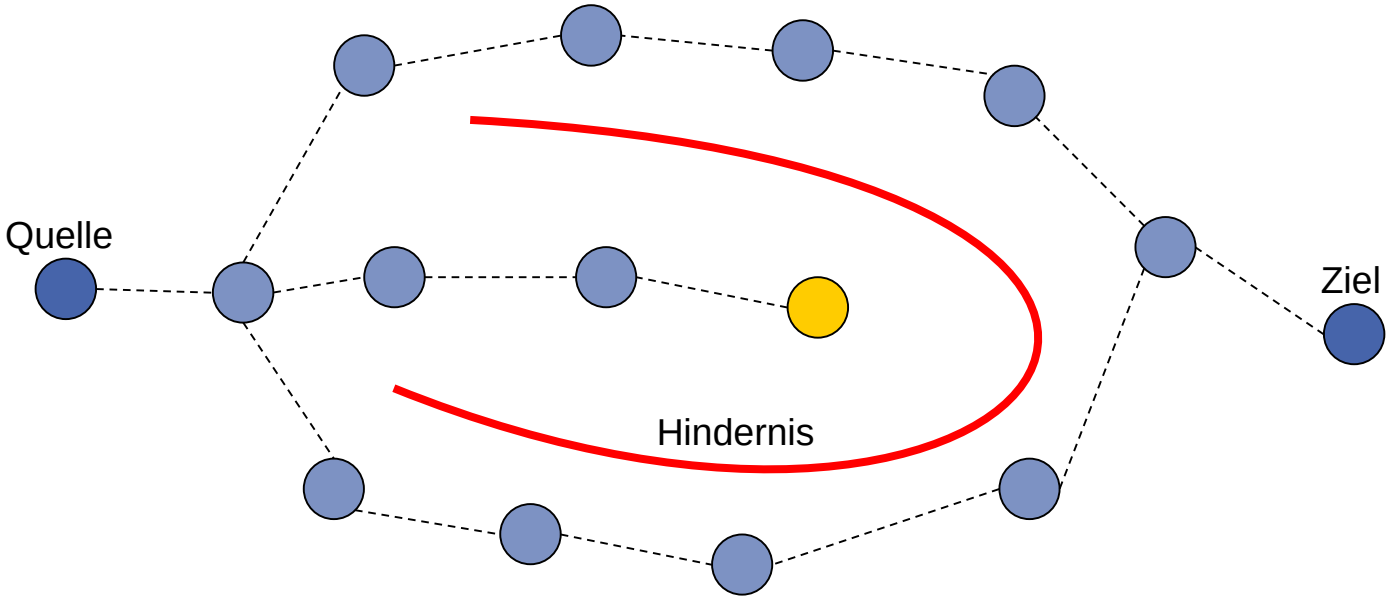
  - Minimaler Winkel zur Verbindungsgerade



- Problem
  - Nicht garantiert frei von Schleifen

Problem der Sackgassen

- Problem
  - Distanz- und Richtungsbasierte Greedy Strategien bleiben in Sackgassen stecken (=lokales Extremum)

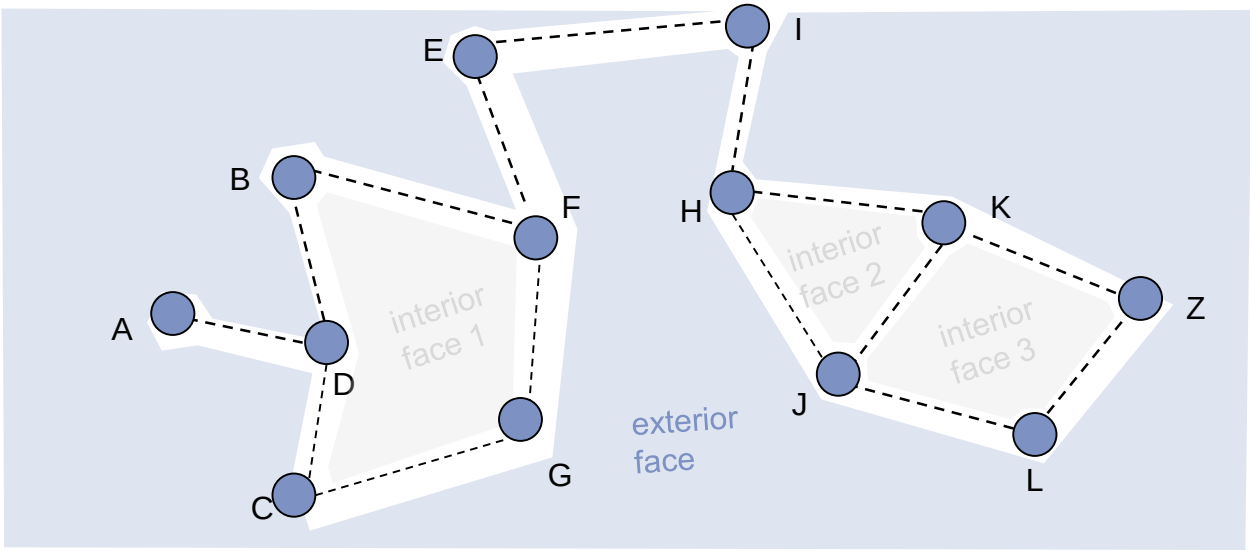


Greedy Perimeter Stateless Routing (GPSR)

[KaKu00]

- Allgemeines
  - 2000 von B. Karp & H. T. Kung vorgestellt
- Idee
  - Rechte Hand Regel erlaubt aus einer Sackgasse (sogar aus einem Labyrinth) zu entkommen
- Vorgehensweise: Weiterleitung in zwei Modi
  - Greedy-Modus
    - Weiterleitung durch distanzbasiertes Greedy-Routing
    - Bringt Dateneinheit „schnell“ dem Ziel näher
  - Perimeter-Modus
    - Weiterleitung durch Perimeter-Routing
    - Ermöglicht das Entkommen aus Sackgassen
    - Nutze „Rechte Hand Regel“ um die Dateneinheit um das Hindernis herum zu leiten
  - GPSR wechselt zwischen Greedy- und Perimeter-Modus

Definitionen: Face & Perimeter



- Face := Größtmögliche Fläche einer *Ebene*, welche *nicht* von Kanten zwischen benachbarten Systemen geschnitten wird
  - Face kann vollständig innerhalb des Netzes liegen (*interior Face*) oder das Netz umschließen (*exterior Face*)
  - Notation: Folge der Knoten, welche beim Umlaufen der Face gegen den Uhrzeigersinn besucht werden. (analog zur Notation in [BETT99])
- Perimeter := Menge der Systeme, die eine Face definieren
  - Im Beispiel
    - Interior Face 1 wird definiert durch die Systeme D, C, G, F, B
    - Interior Face 2 wird definiert durch die Systeme H, J, K
    - Interior Face 3 wird definiert durch die Systeme J, L, Z, K
    - Exterior Face wird definiert durch die Systeme A, D, B, F, E, I, H, K, Z, L, J, H, I, E, F, G, C, D


[BETT99]

Ablauf GPSR

- Initial: Greedy-Modus

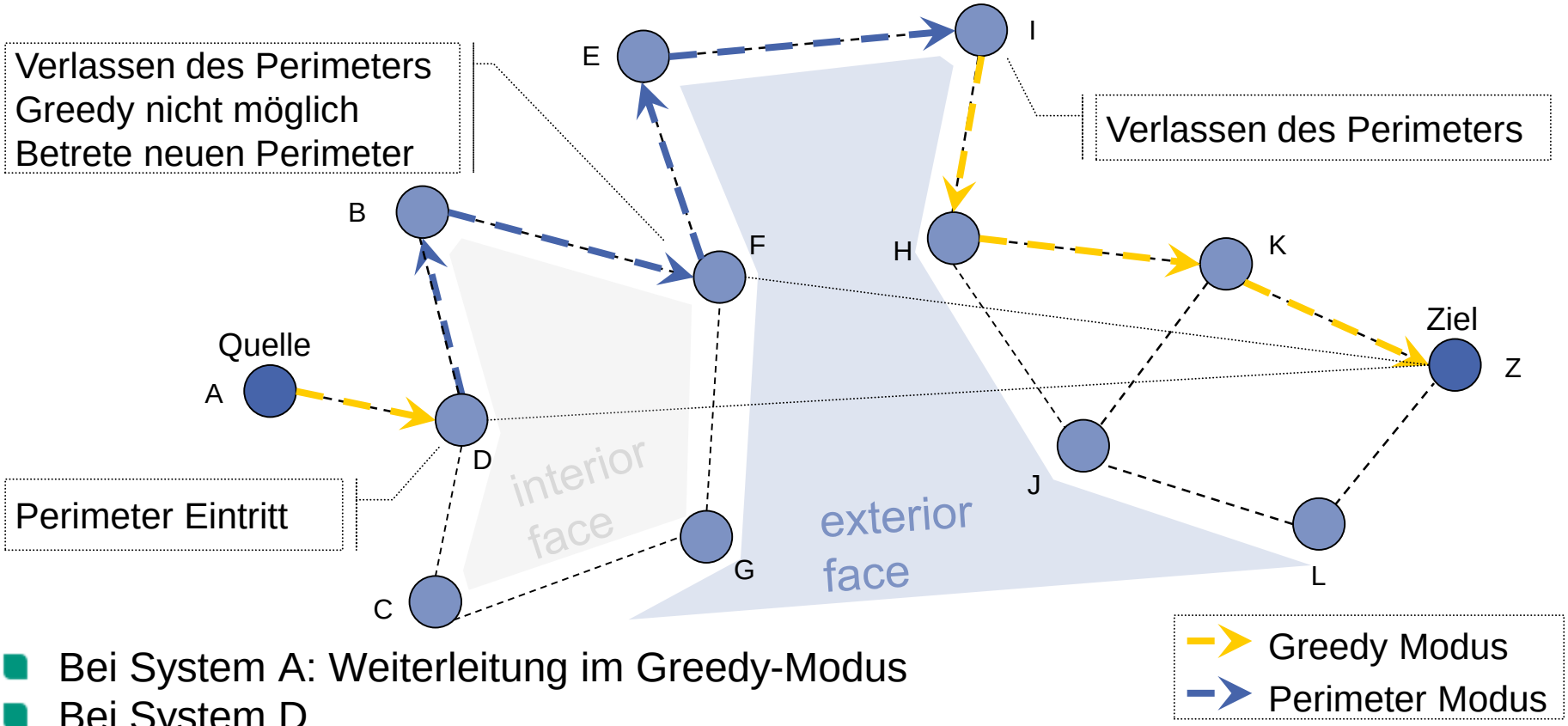
- Greedy-Modus
  - Weiterleitung durch distanzbasiertes Greedy-Verfahren
  - Weiterleitung erfolgt im Greedy-Modus, solange dies möglich ist (also keine Sackgasse vorliegt)
  - Wechsel in den Perimeter-Modus, falls Sackgasse vorliegt

- Perimeter-Modus
  - Position des Eintrittspunktes in Perimeter-Modus wird in der Dateneinheit vermerkt
  - Weiterleitung um Face, welche von der Verbindungsgeraden zwischen aktuellem System und Zielposition geschnitten wird
  - Leite Dateneinheit gemäß der „Rechte Hand Regel“ entlang des entsprechenden Perimeters weiter (= um die Face-Fläche herum)
  - Wechsel in den Greedy-Modus, sobald weiterleitendes System dem Ziel näher ist als der Eintrittspunkt in den Perimeter-Modus

- Anmerkung
  - Varianten verwenden andere Bedingungen zum Modus-Wechsel

[KuWZ03]

Beispiel GPSR



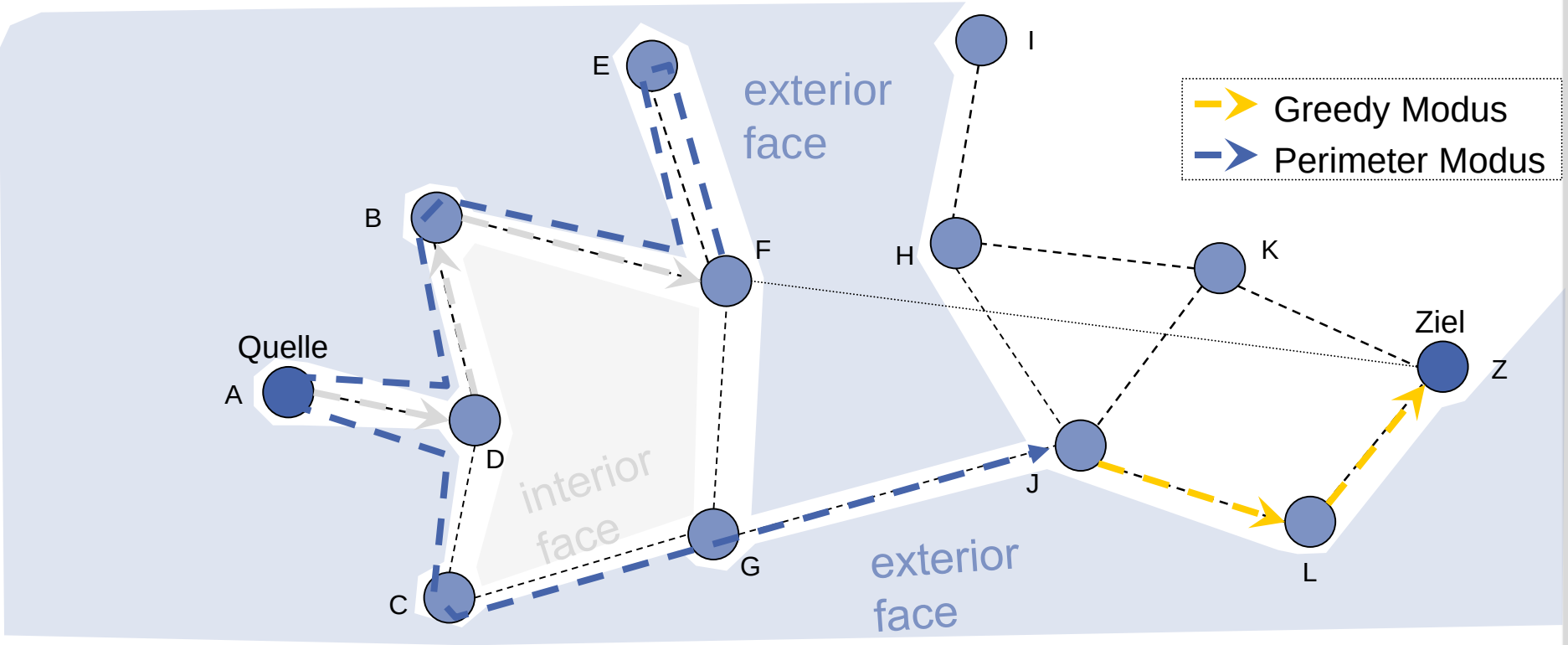
- Bei System A: Weiterleitung im Greedy-Modus
- Bei System D
  - Greedy Weiterleitung nicht möglich, da Nachbarn B & C jeweils weiter als D von Z entfernt sind
  - Wechsel in Perimeter-Modus, bestimme nächsten Perimeter
    - Verbindung DZ schneidet Perimeter (DCGFB)
    - Weiterleitung entlang Perimeter durch Rechte Hand Regel: Systeme D, B, F.

Beispiel GPSR Fortsetzung

- Bei System F
  - Dateneinheit erreicht F im Perimeter-Modus
  - F ist Ziel Z näher als Perimeter-Einstiegspunkt D, daher Wechsel in den Greedy-Modus
  - Greedy Weiterleitung bei F nicht möglich, daher erneuter Wechsel in den Perimeter-Modus
    - Vermerke Position des Systems F als Perimeter-Eintrittspunkt
    - Bestimmung der nächsten Face
      - Verbindung FZ schneidet exterior Face …GFEIHJ…
    - Weiterleitung entlang dieser Face zu Systemen E, I

- Bei System I
  - System I ist Ziel Z näher als Perimeter-Eintrittspunkt F
  - Verlasse Perimeter-Modus, Wechsel in den Greedy-Modus
  - Greedy Weiterleitung entlang der Systeme H, K, Z

Was passiert, falls Kante E-I fehlt?

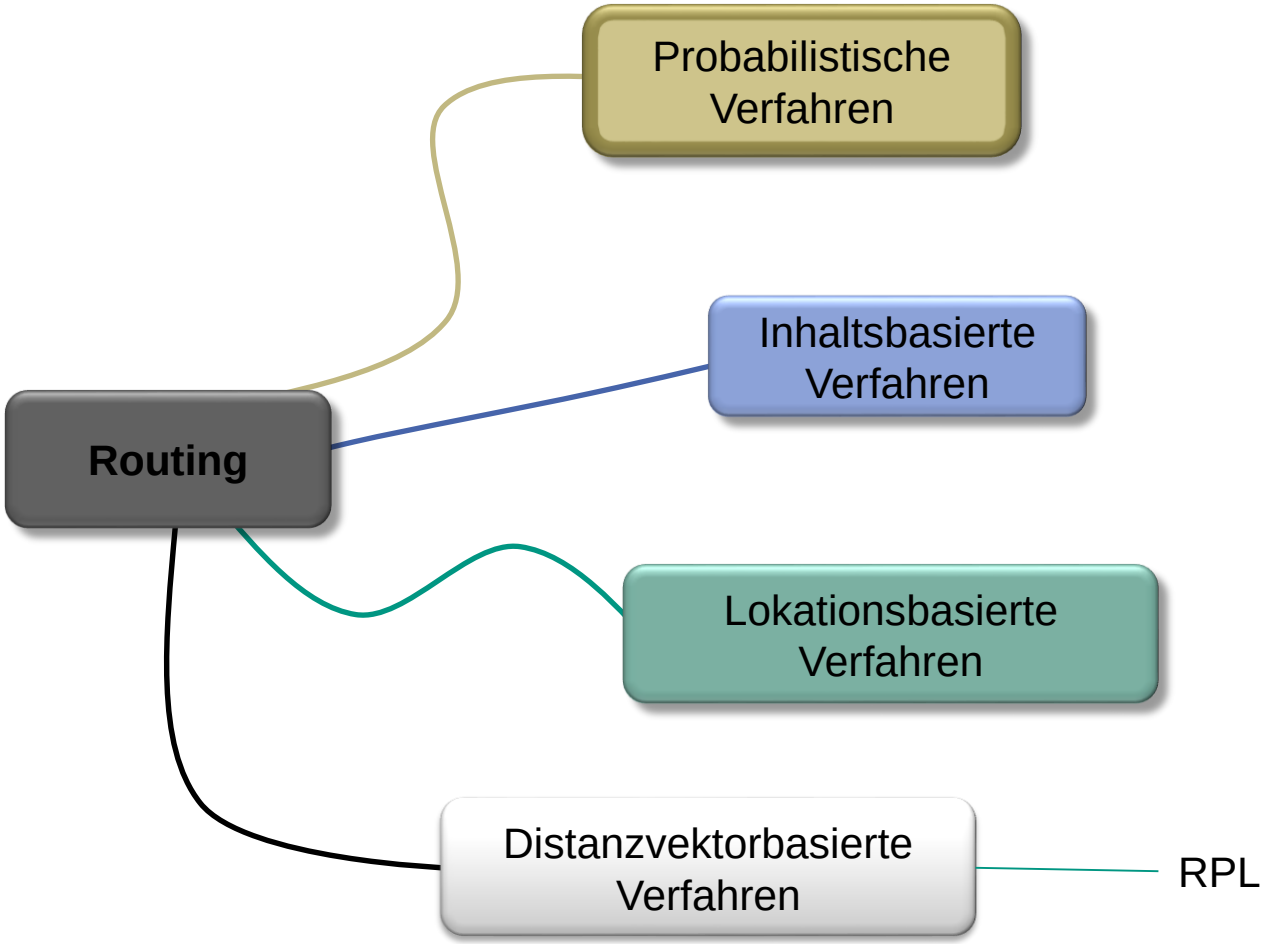


- Annahme: Es existiere eine neue Kante GJ (sonst ist das Netz partitioniert und System Z nicht erreichbar)
  - Bis System F wie zuvor
  - Weiterleitung entlang Perimeter um exterior Face durch Rechte Hand Regel
- Bis System J erreicht wurde mit geringerem Abstand zu Z als Perimeter-Eintrittspunkt F
  - Greedy Weiterleitung möglich

Zusammenfassung & Anmerkungen

- Zusammenfassung
  - Wechsel zwischen zwei Modi
    - Greedy Modus: Distanzbasierte Greedy Strategie
    - Perimeter Modus: Rechte Hand Regel in Sackgassen

- Anmerkung
  - Perimeter Modus erfordert *planaren* Graph
  - Verbindungsgraphen eines drahtlosen Netzes sind i.A. nicht planar
    - Verbindungsgraph muss zunächst zu planaren Graphen ausgedünnt werden
      - Z.B. Gabriel Graph
      - Z.B. Relative Neighborhood Graph
  - Greedy Modus auf vollständigem Graph


Probabilistische Verfahren
Inhaltsbasierte Verfahren
Routing
Lokationsbasierte Verfahren
Distanzvektorbasierte Verfahren
RPL

Anforderungen

- … die „üblichen“ Anforderungen
  - Geringer Energieverbrauch, geringe Datenraten, hohe Fehlerraten, geringe Ressourcen hinsichtlich Speicher und Verarbeitung
  - Betrieb oberhalb eines „Low-Power“ Medienzugriffsprotokolls
    - Unabhängigkeit vom MAC-Protokoll
    - Soll u.a. über 802.14.5 arbeiten können

- Reaktion auf Änderungen
  - In „traditionellen“ Netzen
    - Schnelle Reparatur bei Fehlern, finden alternativer Pfade
    - Beispiel: Fast Reroute (OSPF, IS-IS)
  - In „low power and lossy“ Netzen
    - Überreaktionen vermeiden
    - Kann zu Instabilitäten und nicht akzeptablem Overhead führen

Anforderungen (aus RFC 6550) - Auswahl

- Adaptivität
  - Pfade dynamisch bestimmen/anpassen bei geänderten Umgebungsbedingungen
  - Geeignete Metriken unterstützen, die solche Änderungen erfassen
- Constraint-based Routing
  - Bestimmte Links aufgrund von Randbedingungen („constraints“) ausschließen
    - z.B. niedriger Batteriestand, geringe Linkqualität
- Verschiedene Verkehrsmuster
  - Multipoint-to-Point (Concast), Point-to-Multipoint (Multicast), Point-to-Point (Unicast)
  - Parallele Pfade
- Konfiguration und Management
  - Maxime: *so wenig wie möglich!*

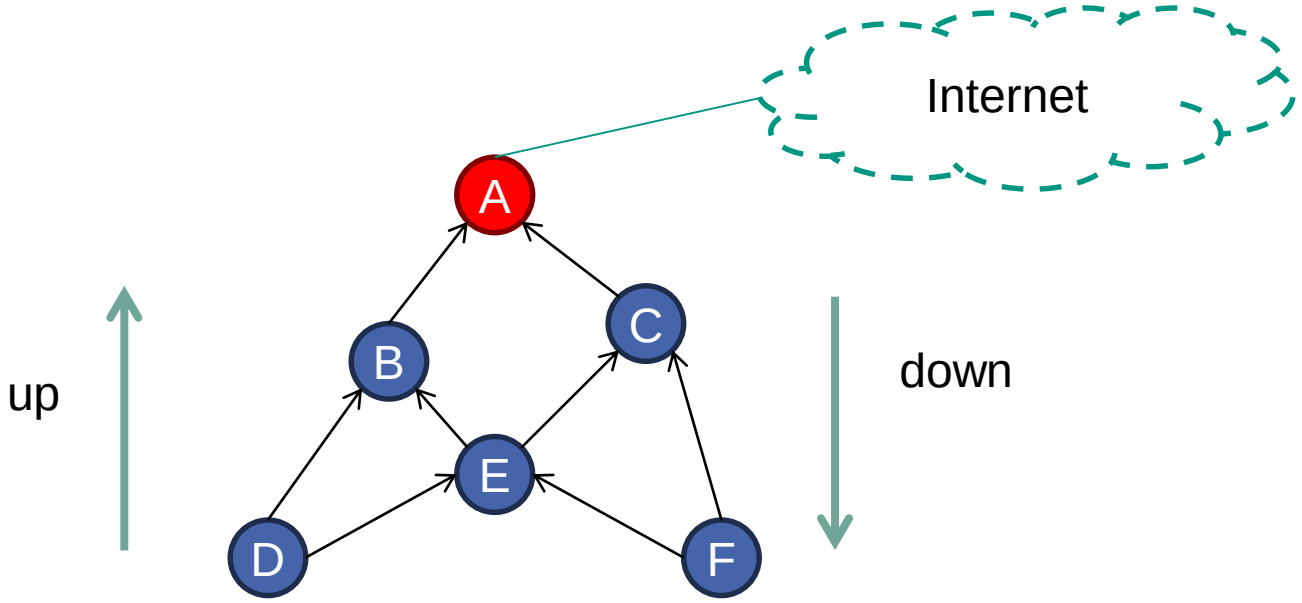
Standardisierung in der IETF

- IETF erkennt Notwendigkeit, Routingprotokoll für „low power lossy networks“ zu entwerfen
  - Arbeitsgruppe „ROLL: Routing Over Low power and Lossy networks“

- Ergebnis: Routingprotokoll RPL („Ripple“), RFC 6550
  - Distanz-Vektor-basiert
  - Aufbau von DODAGs (Destination-oriented Distributed Acyclic Graphs)
    - Nutzung einer Objective Function
      - Übergeordnete Regeln, z.B. Anzahl von Elternknoten, Nutzung von Load-Balancing …)
    - Basiert auf Metriken und Randbedingungen (Constraints)
  - Beispiele
    - Finde Pfade mit besten Übertragungseigenschaften (Metrik) und vermeide unverschlüsselte Übertragungsabschnitte (Constraint)
    - Finde Pfad mit der geringsten Latenz (Metrik) wobei batteriebetriebene Knoten vermieden werden sollen (Constraint)

Grundlegendes Prinzip

- Verbindung von Sensor-basierten Zugangsnetzen zum Internet über wenige dedizierte Knoten (Wurzeln)
  - „up“
    - Verkehr in Richtung Internet: z.B. Sammlung von gemessenen Daten zur Auswertung in der Cloud
  - „down“
    - Verkehr in Richtung der Knoten: z.B. Code-Updates



Grundlegendes Prinzip

- Gerichtete Azyklische Graphen (DAGs, directed acyclic graph)
  - Wurzel ist dedizierter Knoten
    - „up“ in Richtung DAG-Wurzel (Concast)
    - „down“ in entgegengesetzter Richtung (Multicast)
    - Point-to-Point
      - Benutzt beide Richtungen Zunächst „up“ zu gemeinsamem Vorfahren, dann „down“ zum Ziel

- Beispiel
  - Sensorknoten B bis F, Verbindung zum Internet über Knoten A
    - A: Wurzel oder LowPAN Border Router (LBR)
      - Vom Systemadministrator konfiguriert

Grundlegendes Prinzip

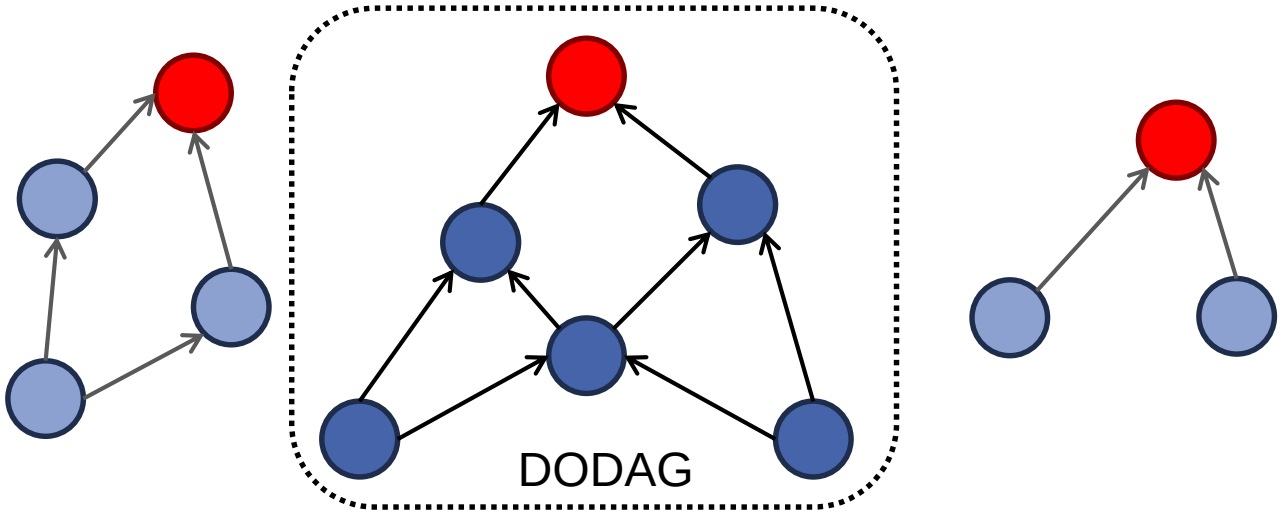
- Konstruktion des DODAG im Überblick
  - Startet von der Wurzel
  - Knoten in der Nachbarschaft empfangen Nachricht, entscheiden ob Beitritt zum Graphen oder nicht
    - … bei einem Beitritt
      - Knoten besitzt nun Route zur Wurzel
      - Wurzel ist Elternknoten
    - Falls Knoten als Router arbeitet
      - Verbreiten der Routinginformation in seiner Nachbarschaft
      - …
    - Falls Knoten nur als Blattknoten operiert
      - Keine weiteren Aktivitäten

- … repräsentiert Concast-Datenfluss (multipoint to point)

Destination Oriented DAG

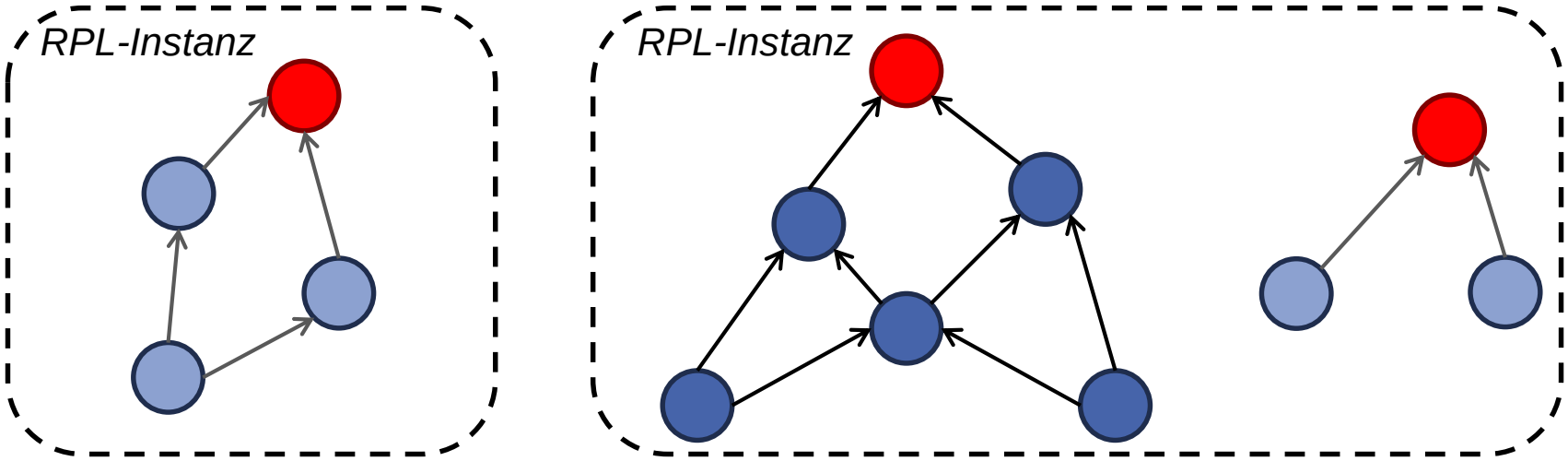
- Eigenschaften
  - Definiert einen DAG, der Pfade zu genau einer Wurzel formt
  - Gültigkeit: innerhalb einer RPL-Instanz
  - Durch DODAG ID innerhalb der Instanz eindeutig identifiziert

- Pro RPL-Instanz kann ein Knoten genau einem DODAG beitreten



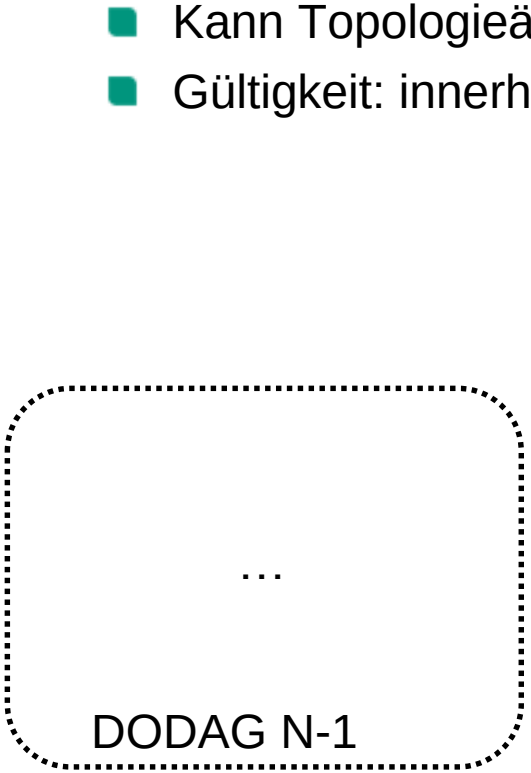
Varianten von RPL-Instanzen

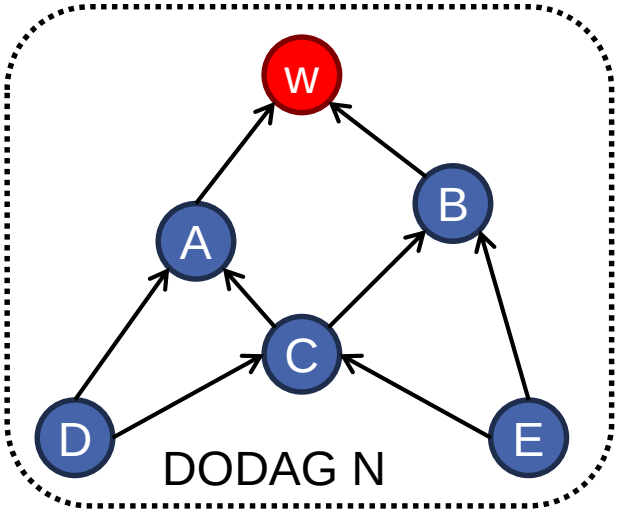
- Einzelner DODAG mit genau einer Wurzel
  - Z.B. einzelner Lichtkontroller in einer Home Automation Anwendung
- Mehrere unkoordinierte DODAGs mit unabhängigen Wurzeln
  - Z.B. mehrere Datensammelpunkt bei Umgebungsbeobachtung
- Ein DODAG mit einer virtuellen Wurzel, die durch mehrere physikalische Wurzelknoten vertreten ist
  - Z.B. mehrere Router mit einem zuverlässigen Transitlink

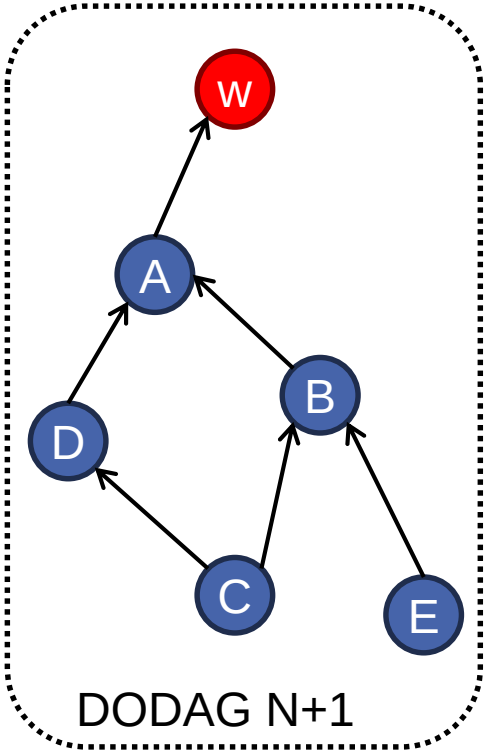


Dynamische Anpassung von DODAGs

- DODAGs können durch Änderungen im Netz nicht mehr optimal sein
  - Dynamische Anpassungen
  - Resultieren in unterschiedlichen DODAG-Versionen
  - Anpassung kontrolliert durch die Wurzel (w)
  - Kann Topologieänderung bewirken (wie in N→N+1)
  - Gültigkeit: innerhalb eines DODAGs

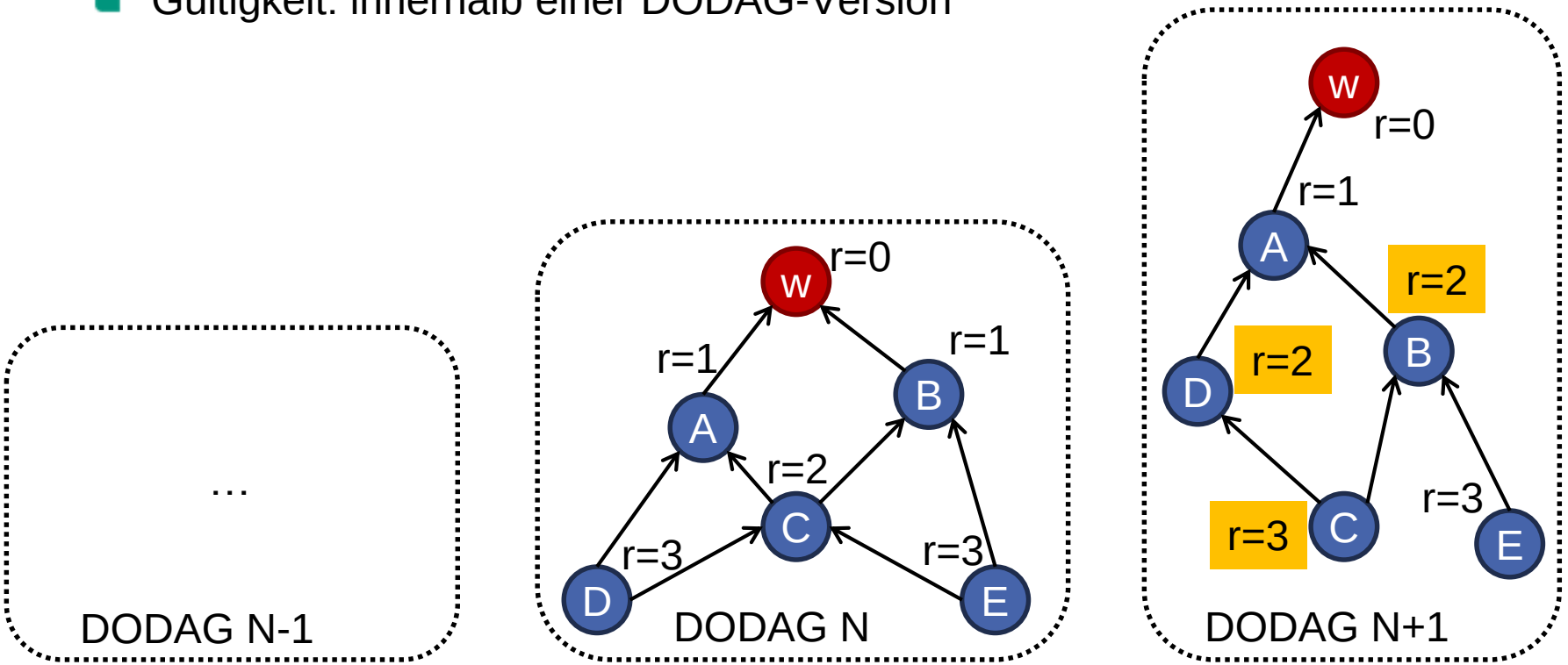






Node Rank

- Skalarer Wert der relative Position eines Knotens im DODAG angibt
  - Relative Position zur Wurzel
  - Wächst streng monoton „nach unten“
    - Wird beim Aufbau des DODAG zur Schleifenvermeidung verwendet
  - Gültigkeit: innerhalb einer DODAG-Version



Objective Function

- Abstrakt formuliertes Pfadkosten-Kriterium
  - Kann Metriken und Constraints enthalten
    - z.B. „schließe Pfade über Batteriebetriebene Knoten aus“
- Definiert, wie die zu verwendenden Metriken und Constraints in den Node Rank umgewandelt werden
  - …und damit welche Knoten als Eltern gewählt werden können
  - Auch, ob genau ein Elternknoten oder mehrere gleichzeitig benutzte Elternknoten gewählt werden sollen
- Jeder DODAG entspricht der Ausprägung einer Objective Function

[Thu11], [VaBr11]

RPL Kontrollnachrichten

- RPL definiert neue ICMPv6 Nachricht mit drei möglichen Typen
  - DAG Information Object (DIO)
    - Enthält Informationen, die es Knoten ermöglichen, RPL Instanzen zu entdecken, Konfigurationsparameter zu erlernen und DODAG Eltern zu wählen
    - Insbesondere enthalten DIOs den NodeRank des Senders
  - DAG Information Solicitation (DIS)
    - Gezielte Anforderung eines DIOs von einem Knoten
  - Destination Advertisement Object (DAO)
    - Um Informationen über Routingziele (Erreichbarkeit) entlang eines DODAGs zu kommunizieren

Genaue Informationen in  [VaBr11] (RPL RFC 6550)

Storing und non-storing nodes

- Ziel: Etablierung von Routinginformation für Down-Richtung
  - Blattknoten senden DAO-Nachrichten in Richtung Wurzel
- Problem: Speicheraufwand in den Knoten
  - Non-storing nodes
    - Halten keine Routinginformation
    - Wurzel fügt Source-Routing-Information in Dateneinheiten ein
  - Storing nodes
    - Knoten halten Routinginformation
  - … Mischbetrieb ist nicht zulässig

RPL und Sicherheit

- Grundlegend
  - Wichtig, aber auf sehr ressourcenarmen Geräten oftmals problematisch
- RPL-Optionen
  - Unsecured
    - Keinerlei zusätzliche Sicherheitsmaßnahmen für Kontrollnachrichten
  - Pre-installed
    - Vorinstallierte gemeinsame Geheimnisse auf den Knoten
    - Gesicherte RPL-Nachrichten
  - Authentifiziert
    - Beitritt als Blattknoten mit vorinstalliertem Schlüssel
    - Als Router: Schlüssel muss von Authentication Authority erworben werden
- Schutzziele
  - Integrität
  - Vertraulichkeit
  - Schutz vor Wiedereinspielen
  - Schutz vor Verzögerungen

DODAG Konstruktion

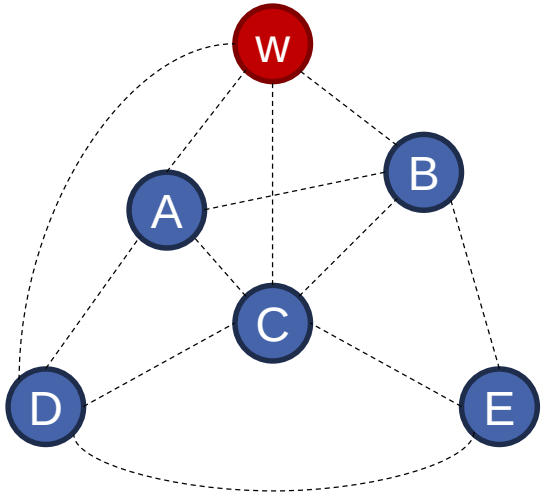
- Knoten schicken periodisch link-lokale Multicast DIO-Nachrichten
  - Stabilität des DODAGs oder etwaige Inkonsistenzen beeinflussen Periodendauer
- Knoten hören auf DIO-Nachrichten und verwenden die Information darin, um einem DODAG beizutreten oder einen DODAG zu verwalten
  - Knoten können DIS Nachrichten benutzen, um DIOs anzufordern
- Elternwahl erfolgt aufgrund von Informationen in DIO-Nachrichten (z.B. Node Rank)

➔ Ergebnis: Upward Routen in Richtung der DODAG Wurzel
  - Wurzel selbst muss nicht bekannt sein
    - Abstand (NodeRanks) reicht aus
    - Abstand über lokale DIOs bekannt – diese enthalten NodeRank des Senders
  - OF bestimmt aus Metrik den NodeRank, aufgrund dessen Eltern gewählt werden

  → somit automatisch Routen in „richtige Richtung“ wegen strenger Monotonie

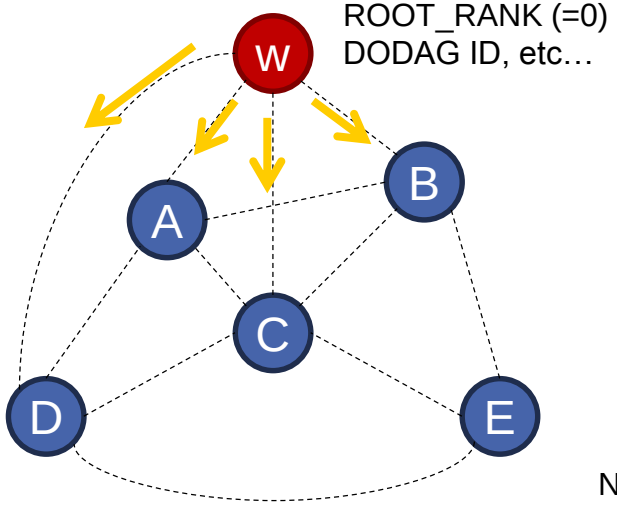
DODAG-Konstruktion Schritt für Schritt – Vorbemerkungen

- Im folgenden Beispiel gelte
  - NodeRank
    - Ganzzahl-Anteil der (aktuell) minimalen Pfadkosten zur Wurzel
  - Pfadkosten (immer zur Wurzel)
    - Summe der Linkkosten entlang des Pfades
  - Linkkosten
    - Symmetrisch
    - Während der Konstruktion konstant
- Elternwahl
  - Wähle zwei Knoten
    - Minimiere dabei Pfadkosten
  - Knoten dürfen ihren NodeRank nur verkleinern, nicht vergrößern
    - Dient der Vermeidung von bestimmten Fehlern, dazu mehr nach dem Beispiel…
- DIOs
  - Enthalten insbesondere
    - Eigenen NodeRank
    - Eigene Pfadkosten
      - Pfadkosten über besten Elternknoten
  - Werden gesendet
    - Initial von der Wurzel
    - Wenn sich eigener NodeRank oder Pfadkosten geändert haben

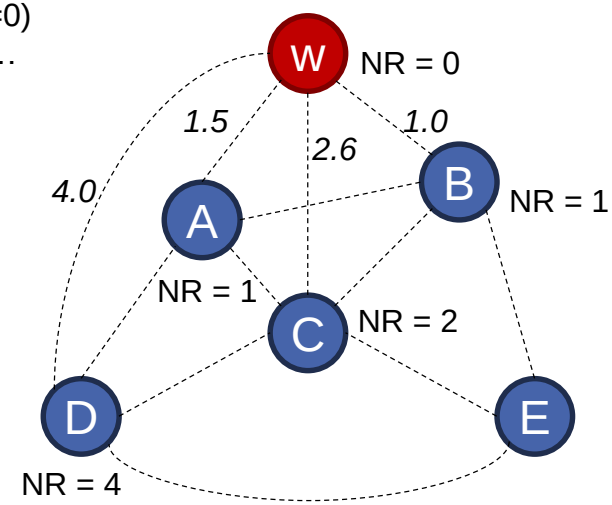


DODAG-Konstruktion Schritt für Schritt

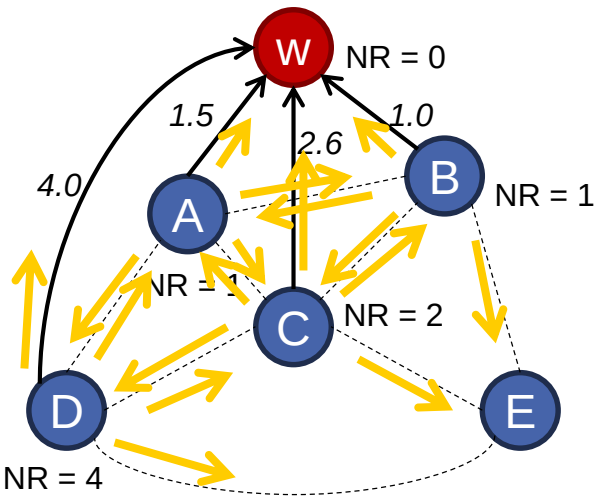
1.) Initialisierung durch Wurzel



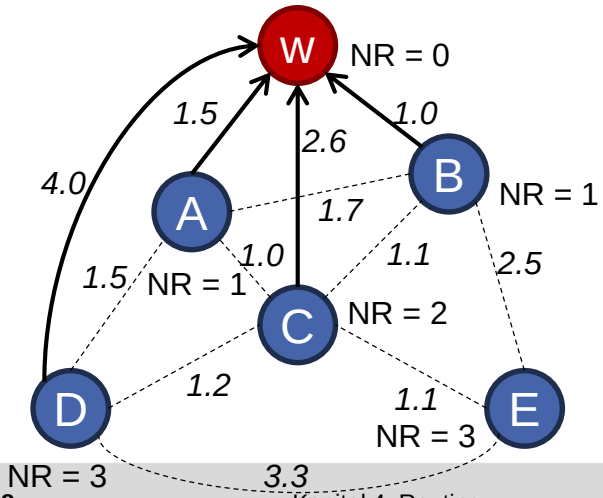
2.) Link-Kosten u. NodeRanks



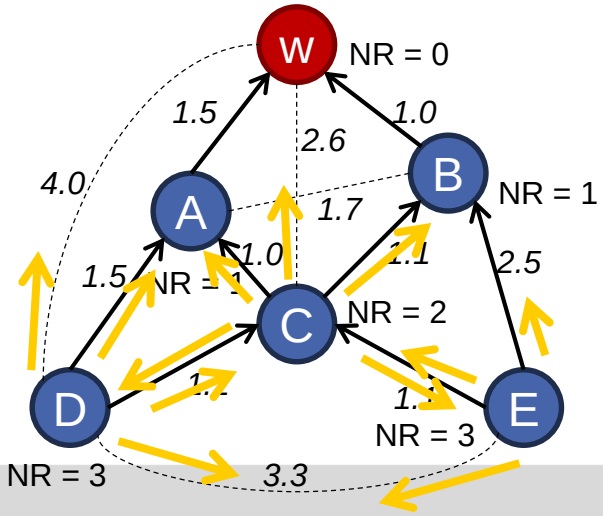
3.) Elternwahl u. DIO Multicast



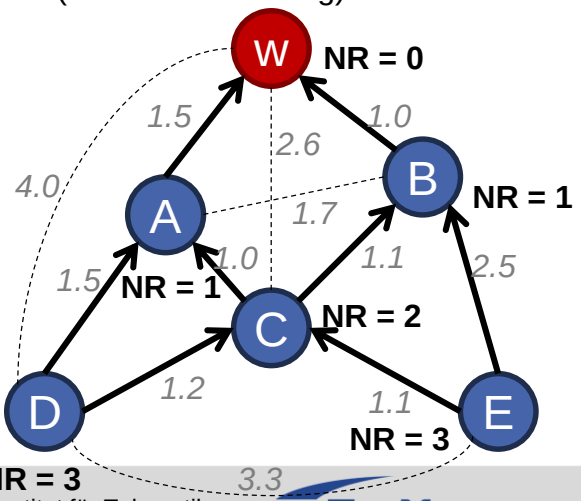
4.) Link-Kosten u. NodeRanks



5.) Elternwahl u. DIO



6.) Link-Kosten u. NodeRanks (keine Veränderung)

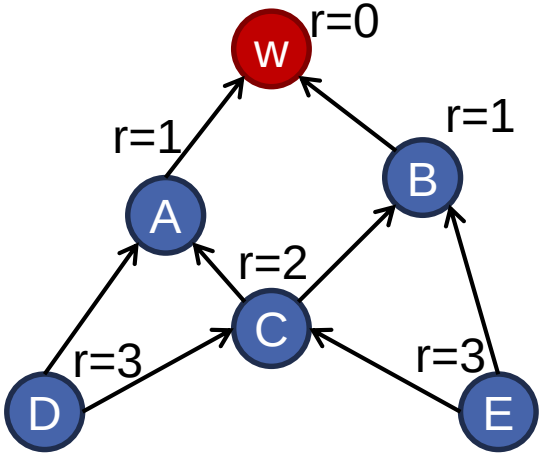


DODAG-Konstruktion Schritt für Schritt – Nachbemerkungen

- Schritt 1
  - Wurzel gibt ROOT_RANK bekannt
  - Weitere (logische) Wurzeln würden denselben Wert benutzen
- Schritt 2
  - Zufällige Wartezeit auch möglich
  - Dann hätte Knoten D möglicherweise schon von A und C DIOs bekommen und w nicht als Elter gewählt
- Schritt 4
  - Knoten D
    - Pfadkosten(w) = 4.0
    - Pfadkosten(A) = 3.0
    - Pfadkosten(C) = 3.8
    - ➔ min = 3.0 → NodeRank = 3
  - Knoten C
    - Pfadkosten(w) = 2.6
    - Pfadkosten(A) = 2.5
    - Pfadkosten(B) = 2.1
- Schritt 5
  - Knoten C, D, E haben geänderte Pfadkosten
  - Knoten D hat geänderten Rank
  - Knoten E
    - Pfadkosten(B) = 3.5
    - Pfadkosten(C) = 3.7
    - → eigene Pfadkosten: 3.5
  - ➔ erneuter DIO multicast
  - B und A können keinen zweiten Elternknoten bestimmen
    - Kein Knoten hat einen passenden NodeRank (kleiner als 1)
- Schritt 6
  - Knoten E muss streng genommen nochmal DIO senden
    - Pfadkosten(C) = 3.2
    - → eigene Pfadkosten = 3.2
  - Schritt im Bsp ausgelassen weil keine Topologieänderungen mehr

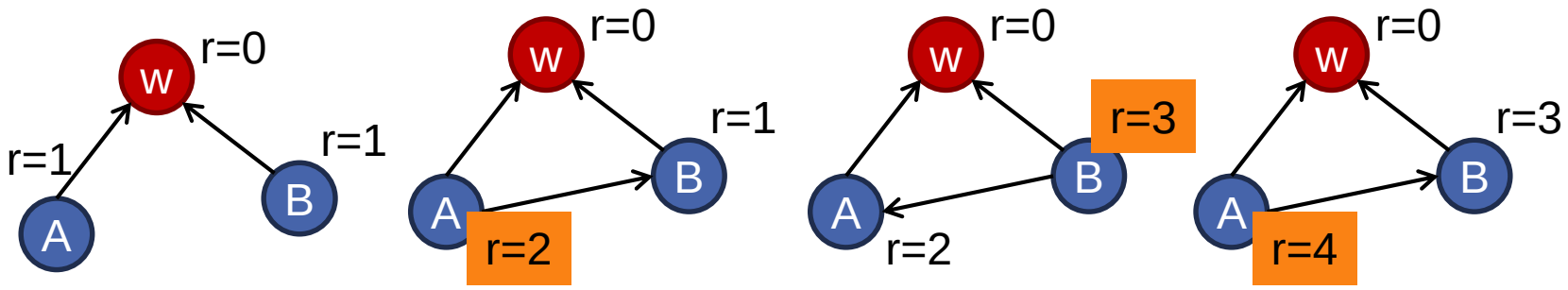
DODAG Konstruktion Distanzvektorbasiert

- Distanzvektor-Protokoll
  - Es werden Pfadkosten für günstigsten Pfad zur Wurzel bekanntgegeben
  - Elternknoten werden solche, die die Pfadkosten minimieren
    - Es wird eine Elternmenge und ein bevorzugter Elternknoten gewählt
    - Je nach OF können auch mehrere bevorzugte Eltern gewählt werden (s. Knoten C, D, E im Beispiel)
- Aber Vorsicht geboten im Hinblick auf
  - Schleifen
  - Count-to-infinity



NodeRank – Greediness und Count-to-infinity

- Es kann günstig sein, einen hohen Node Rank zu haben
  - Größere Menge an potentiellen Elternknoten steht zur Verfügung
    - Bessere Robustheit bei Knoten-/Linkausfällen
  - Weniger Knoten können selbst diesen Knoten als Elternknoten wählen (Schleifenvermeidung) → Energieersparnis, weniger Weiterleitungen
- Problem der Greediness am Beispiel
  - 1: Knoten A verlässt DODAG und tritt mit höherem Rank wieder bei
  - 2: Knoten B tut dasselbe, gehe zu 1:



[VaBr11]

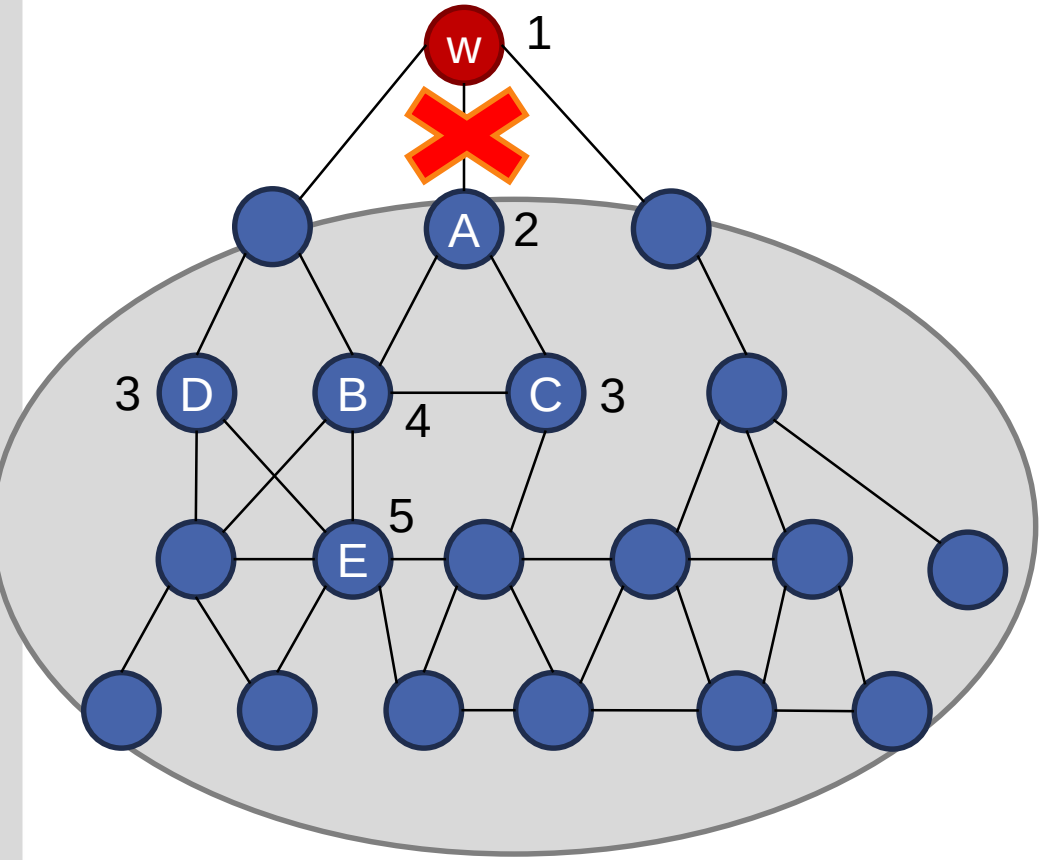
Schleifen – max_depth rule

- Schleifenbildung unvermeidbar
  - Links bzw. Linkkosten können sich überall ständig ändern
  - 100% Schleifenvermeidung würde großen Aufwand bedeuten
- Die „max_depth rule“ schränkt Elternwahl ein
  - Erlaubt sind alle Knoten, deren Node rank
    - Kleiner ist als der eigene, bzw.
    - Den eigenen Node rank nicht mehr als einen bestimmten Wert überschreiten
      - $NodeRank(Parent) < NodeRank(self) + DAGMaxRankIncrease$

➔ max_depth rule verhindert nicht das Entstehen von Schleifen

  - schränkt aber deren Größe ein
  - DAGMaxRankIncrease kann auch 0 sein
    - einstellbar durch Administrator, Bekanntgabe durch DODAG Wurzel
- Es ist ausdrücklich erlaubt, Geschwister- und Kinder bzw. Nachkommen als Eltern zu wählen.
  - Rank muss angepasst werden (strenge Monotonie!)

max_depth rule Beispiel



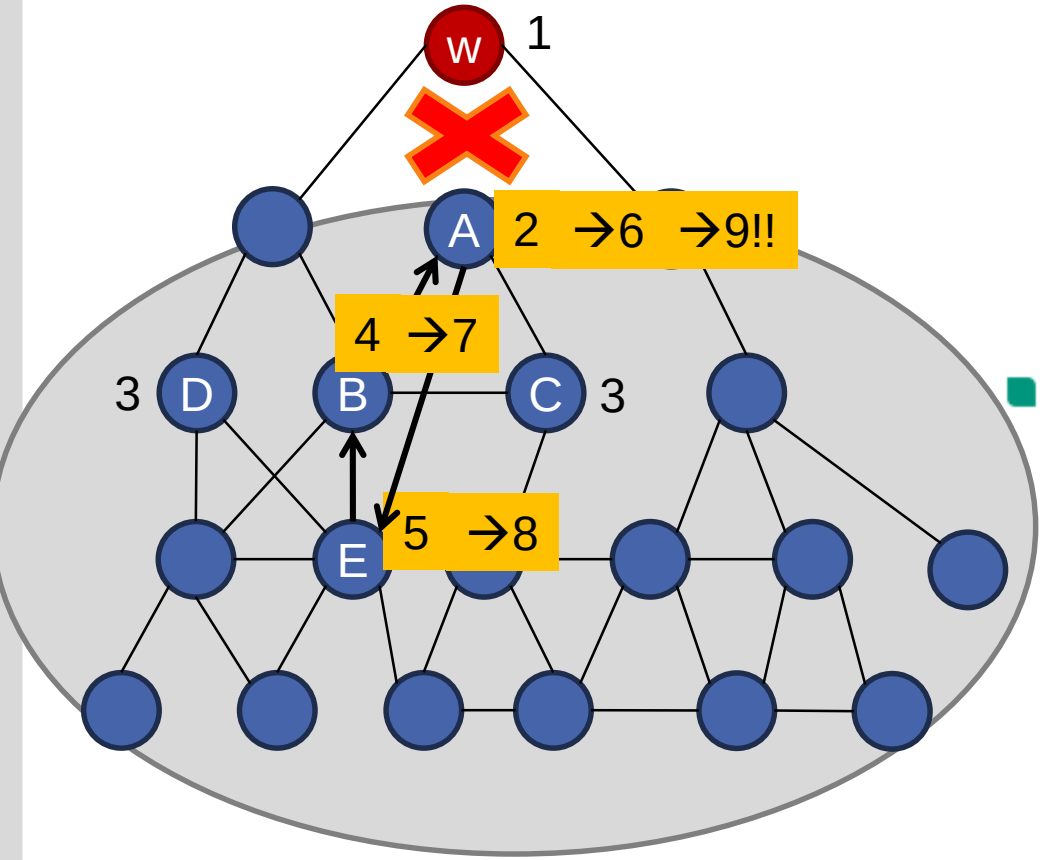
- Szenario
  - Link zwischen Wurzel und A fällt aus
  - DAGMaxRankIncrease = 5
  - NodeRank(A) = 2

➔ A darf sich einen neuen Elternknoten mit maximalem NodeRank 7 wählen

  - Potentiell alle Knoten im Netz
  - Angenommen, A wählt E als neuen bevorzugten Elternknoten

Nach [VaBr10]

Schleifenerkennung Beispiel



- A hat E als neuen Elternknoten gewählt
  - A muss nun seinen neuen Rank bekanntgeben (6)
  - B muss Rank anpassen (7)
  - E muss Rank anpassen (8)
  - … woraufhin wiederum A seinen Rank anpassen muss (9)
- Schleife wird an dieser Stelle erkannt
  - 9 ist größer als erlaubt
    - $NewRank \leq oldRank + DAGMaxRankIncrease$
  - Knoten A löst sich aus dem DODAG und sucht sich einen anderen Elternknoten
    - z.B. Knoten D

Schleifenerkennung bei Datenübertragung

- Bisher besprochen: Schleifenentstehung bei Signalisierung
  - Zur Vermeidung von Schleifen Verwendung von
    - Einschränkungen bei Bekanntgabe und Annahme des NodeRank
    - Einschränkungen bei der Elternwahl
  - Immer noch möglich: Schleifen im Datenpfad
    - Bspw. wenn Links vorübergehend ausfallen
    - Keine unmittelbare Reaktion durch das Routingprotokoll
      - Zu teuer, da hoch dynamische Umgebung (und oft wenig Datenaufkommen!)

- Ansatz bei RPL: Piggybacking von Routing-Kontrollinformation in Dateneinheiten
  - Setzen von Feldern im Kopf der Dateneinheit
    - Möglichkeit: IPv6 extension header
  - Verfahren auch „datapath validation“ genannt

Data Path Validation - Schleifenerkennung

- Benötigte Information
  - NodeRank des Absenders
  - Richtung („up“ oder „down“) der Dateneinheit

- Beispiel für „verirrte Dateneinheit“
  - Dateneinheit markiert als in „up“ Richtung bewegend
  - Sender hat geringeren NodeRank als Empfänger
    - ➔ Empfänger weiter von der Wurzel entfernt

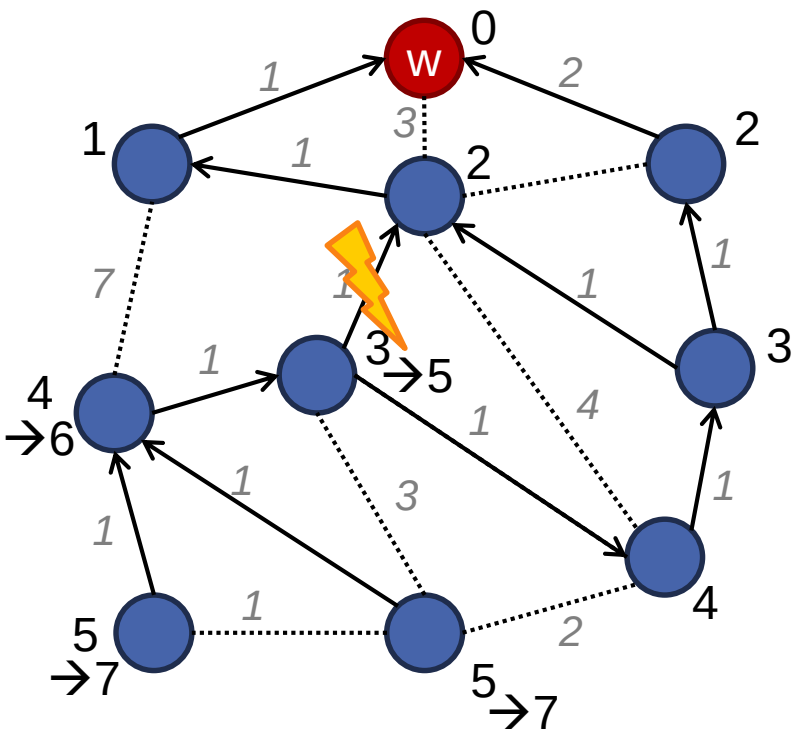
  ➔ Schlussfolgerung: Dateneinheit hat sich nicht in angezeigter Richtung („up“) bewegt

  ➔ Inkonsistenter DODAG

- Reaktion nach Feststellung: Reparatur des DODAG
  - s. nachfolgende Folien

DODAG Reparatur – lokale Strategien

- Wahl eines anderen Elternknotens



DODAG Routing Link

Nicht genutzter Link

0 Node Rank

→6 Neuer Node Rank

2 Link-Kosten

Link-Ausfall oder Inkonsistenz erkannt

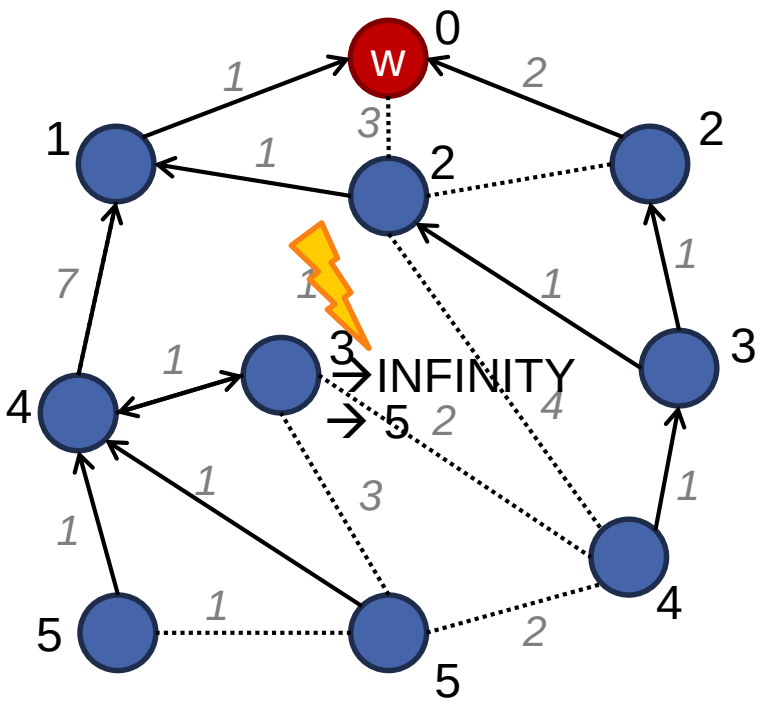
Wahl eines neuen Elternknotens

Anpassung des eigenen Node Ranks (mit anschließender Bekanntgabe)

Anpassungen im sub-DODAG

DODAG Reparatur – lokale Strategien

- Poison-and-Wait (falls kein neuer Elter gewählt werden kann)



DODAG Routing Link
Nicht genutzter Link
0 Node Rank
2 Link-Kosten

Link-Ausfall oder Inkonsistenz erkannt
Poisoning des Sub-DODAGs
Löschen aus allen Eltern-Sets aller tatsächlichen oder potenziellen Kinder
Neue Elternwahl wo nötig (falls nicht möglich, poison and wait)
Rejoin nach gewisser Wartezeit

DODAG Reparatur – globale Strategie

- Neue Version des DODAG erzeugen
  - Durch die Wurzel initialisiert
    - Standard spezifiziert nicht, wann oder unter welchen Voraussetzungen
  - Neuberechnung des DODAG
    - Ohne Einschränkungen bei der Bekanntgabe von NodeRanks oder Elternwahl
    - Aufgrund aktueller Link-Metriken und Constraints

➔ Kompletter „Neustart“ der Topologie, inklusive erneuter Optimierung anhand der Objective Function

- Im Vergleich deutlich teurer als lokale Reparatur
- Aber hinterher bessere Pfade!

Diskussion I

- Annahmen über Verkehrsmuster
  - MP2P stellt den „Löwenanteil des Verkehrs“
    - Hier findet RPL redundante und optimierte Pfade
  - P2MP ist selten
    - Kommunikation mit Aktoren wird nicht beachtet
    - Aktives Auslesen (s. Directed Diffusion) wird als sekundär betrachtet
  - P2P ist ein Kuriosum, was so gut wie nirgends auftritt
    - Pfade sind nicht optimal, im non-storing Mode wird *immer* über die Wurzel kommuniziert!
- Gelten diese Annahmen für alle Szenarien?
  - Für alle Anwendungen?
- Generierung von „down“-Routen mittels DAOs erzeugt umfangreichen Signalisierungsoverhead

[ClHePh11]

Diskussion II

- Fragmentierung auf Vermittlungsschicht
  - IEEE 802.15.4 beschränkt die L2 Rahmengröße auf 127 Bytes
  - RPL entworfen für IPv6
    - Abweichend von der spezifizierten L3 MTU von mind. 1280 Bytes
    - Header von 40 Oktetten komprimierbar auf 2 Oktette mittels bestimmten Adaptionsschichten
  - Also 127 Oktette minus:
    - 25 Oktette L2 Frame Overhead
    - 21 Oktette Link Layer Security
    - 2 Oktette komprimierter IPv6 header
  - Ergibt 79 Oktette L3 Nutzdaten
- Reicht doch, oder?
  - RPL Kontrollverkehr wird über ICMPv6 gesendet (Header 4 Oktette)
  - Und DIOs werden nicht selten über 80 Oktette lang
    - ➔ Reicht also nicht, *daher wird Fragmentierung benötigt*
  - Weiterhin vergrößert sich bei Source-Routing der L3-Header in O(#hops).
- Aber: Verlust eines Fragmentes bedeutet Verlust des Pakets!
  - Und das Netz ist per Definition hoch verlustbehaftet („lossy“!)

Diskussion III

- Aggregation von Adressen
  - Im storing mode muss ein RPL-Router Routingtabellen-Einträge für seinen gesamten sub-DODAG speichern
    - Realisiert über Präfixe → Voraussetzung ist, dass Adressen aggregierbar sind
  - Eltern-Wechsel ist erwartetes Verhalten
- Einsetzbarkeit von RPL ohne Aggregation von L3-Adressen fraglich
  - Dann können Routingtabellen „sehr groß“ werden
    - Im Vergleich zum Internet immer noch klein
    - Aber Speicher ist eine knappe Ressource in Sensornetzen!
  - Vor allem in der Nähe der Wurzel, wo sehr viele Knoten in „down“-Richtung erreichbar sind
- I.A. nur noch Baumtopologie, kein allgemeiner DODAG mehr
  - Eltern können nur noch redundant gewählt werden, wenn ein gemeinsames Präfix vorhanden ist

Die von uns zur Erstellung der Folien genutzte

LITERATUR

Literatur I

[BETT99] Battista G. D., Eades P., Tamassia R., Tollis T.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999

[BMSU99] Bose P., Morin I., Stojmenovic I., Urrutia J.: Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications. S. 48-55. Seattle, WA, USA. 1999.

[BrEs02] Braginsky D., Estrin D.: Rumor Routing Algorthim for Sensor Networks. In: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02). S. 22-31. Atlanta, GA, USA, 2002.

[ClHePh11] T. Clausen, U. Herberg, M. Philipp; *A Critical Evaluation of the „IPv6 Routing Protocol for Low Power and Lossy Networks" (RPL)*, May 2011. Rapport de recherche, INRIA.

[CrKa04] Crescenzi P., Kann V.: A Compendium of NP Optimization Problems. http://www.nada.kth.se/~viggo/wwwcompendium/wwwcompendium.html. Februar 2004.

[GGHZ01] Gao J., Guibas L.J., Hershberger J., Zhang L., Zhu A.: Geometric Spanners for Routing in Mobile Networks. In: Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Long Beach, CA, USA. 2001.

[GuKh98] Guha S., Khuller S.: Approximation Algorithms for Connected Dominating Set. Algorithmica, 20. S. 374-387. 1998.

Literatur II

[HaHL02] Haas Z. J., Halpern J. Y., Li L.: Gossip-Based Ad Hoc Routing. In: Proceedings of the IEEE Infocom. Vol 3, S.1707-1712. New York, NY, USA, Juni 2002

[HCB00] Heinzelman W.R., Chandrakasan A., Balakrishnan H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences. S. 174-185. Hawaii, HI, USA. Januar 2000.

[InGE03] Intagonwiwat C., Govindan R., Estrin D.: Directed Diffusion for Wireless Sensor Networking. IEEE/ACM Transactions on Networking, Vol. 11, No. 1, Februar 2003

[Inta00] C. Intagonwiwat, R. Govindan, D. Estrin; Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks; Proceedings ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Boston, MA, USA, August 2000

[KaKu00] Karp B. & Kung H. T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom '00), ACM Press, 2000, 243-254

[KaWi05] H. Karl, A. Willig: Protocols and Architectures for Wireless Sensor Networks, Wiley & Sons, ISBN 0470095105, 2005

[KuWZ03] Kuhn F., Wattenhofer R., Zollinger A.: Worst-Case optimal and average-case efficient geometric ad-hoc routing Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, ACM, 2003, S. 267-278

Literatur III

[RaRo00] Ramanathan R., Rosales-Hain R.: Topology Control of Multihop Wireless Networks using Transmit Power Adjustments. In: Proceedings of IEEE Infocom, Tel-Aviv, Israel. S. 404-413. März 2000.

[SelfOrg] F. Dressler, Vorlesung Selbstorganisation, http://www.ccs-labs.org/~dressler/teaching/selbstorganisation-ss10/ Buch: Self-Organization in Sensor and Actor Networks. Wiley & Sons 2007, ISBN 978-0-470-02820-9

[TaKl84] Takagi H. & Kleinrock L.: Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. IEEE Transactions on Communications, 1984, 32, 246-257

[Thu11] B. Thubert, Ed.; RPL Objective Function 0 draft-ietf-roll-of0-20, Internet-Draft, September 2011. Expires 08. March 2012. Work in Progress.

[Tous80] Toussaint G.: The Relative Neighborhood Graph of a Finite Planar Set. In: Pattern Recognition 12: S. 261-268. 1980.

[VaBr11] J.-P. Vasseur et al.; RPL: IPv6 Routing Protocol for Low power and Lossy Networks, RFC 6550

[VaDu10] J.-P. Vasseur, A. Dunkels; Interconnecting Smart Objects with IP – the next internet, Kapitel 17. Morgan Kaufmann 2010, ISBN 978-0-12-375165-2

[WuLi00] Wu J., Li H.: On Calculating Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In: Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications. Boston, MA. August 2000.