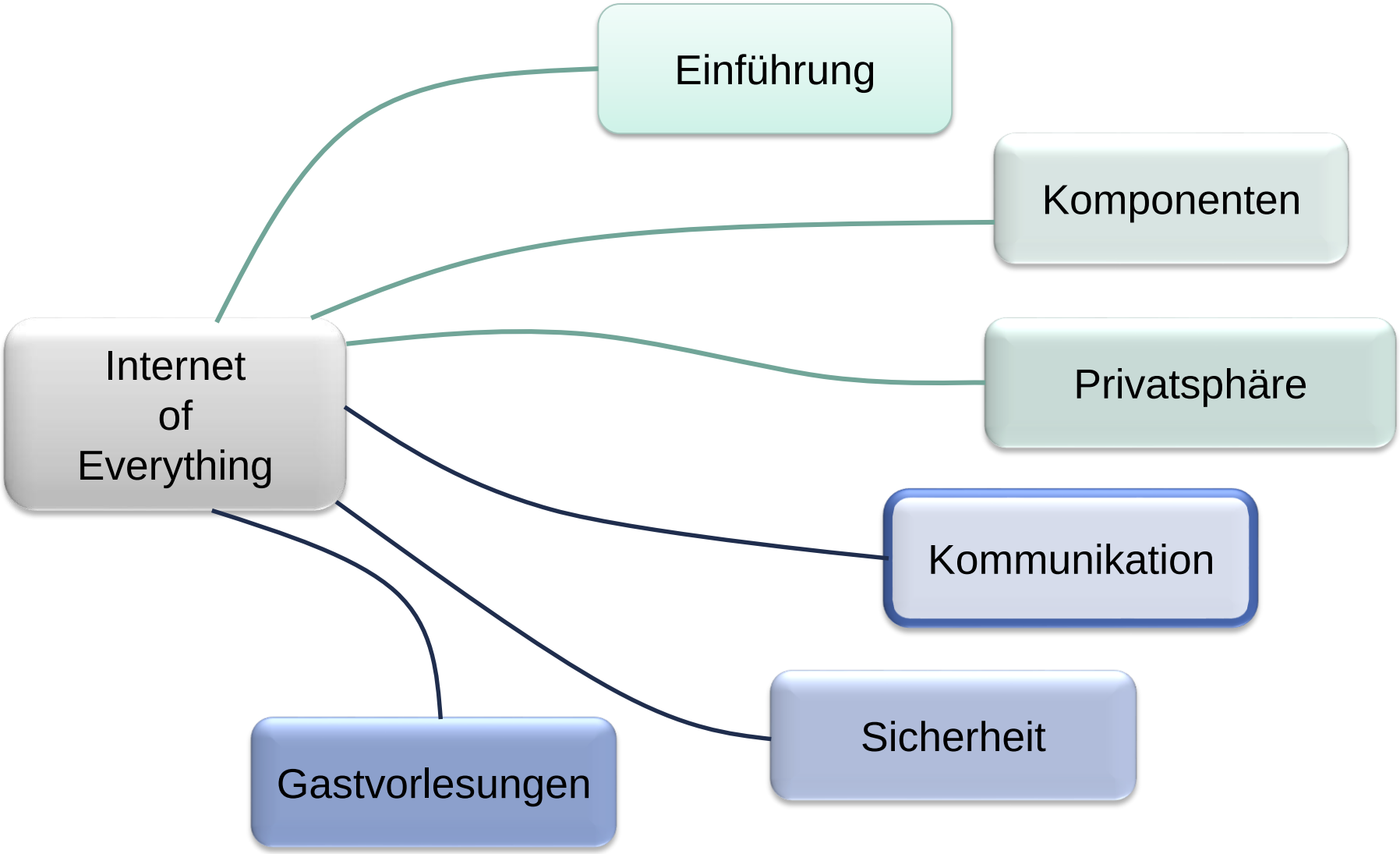
Karlsruher Institut für Technologie

Vorlesung Internet of Everything
Kapitel 4.4 – Datentransport

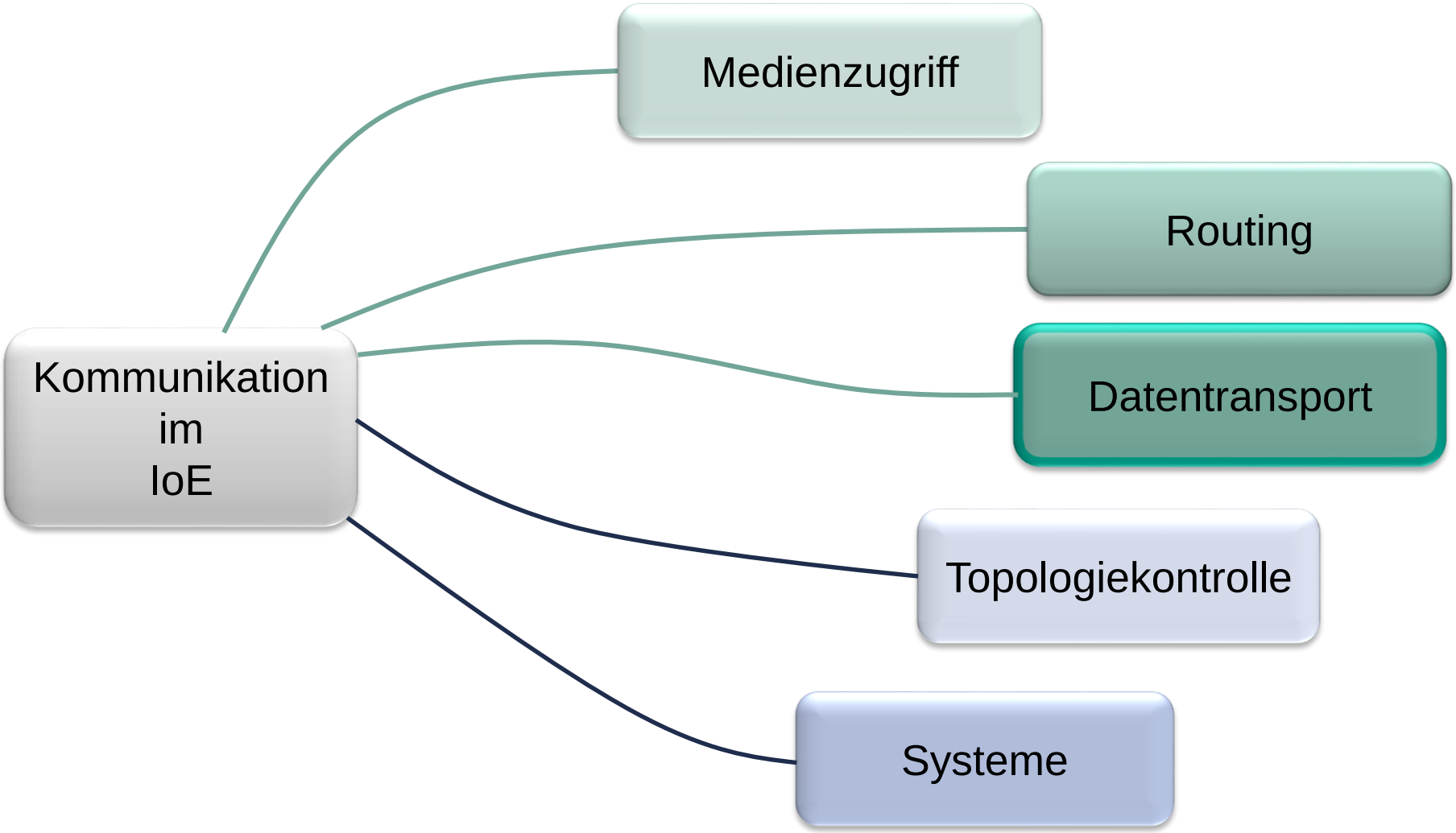
Prof. Dr. Martina Zitterbart, Martin Florian, Markus Jung
[zitterbart, florian, m.jung]@kit.edu

Institut für Telematik, Prof. Zitterbart

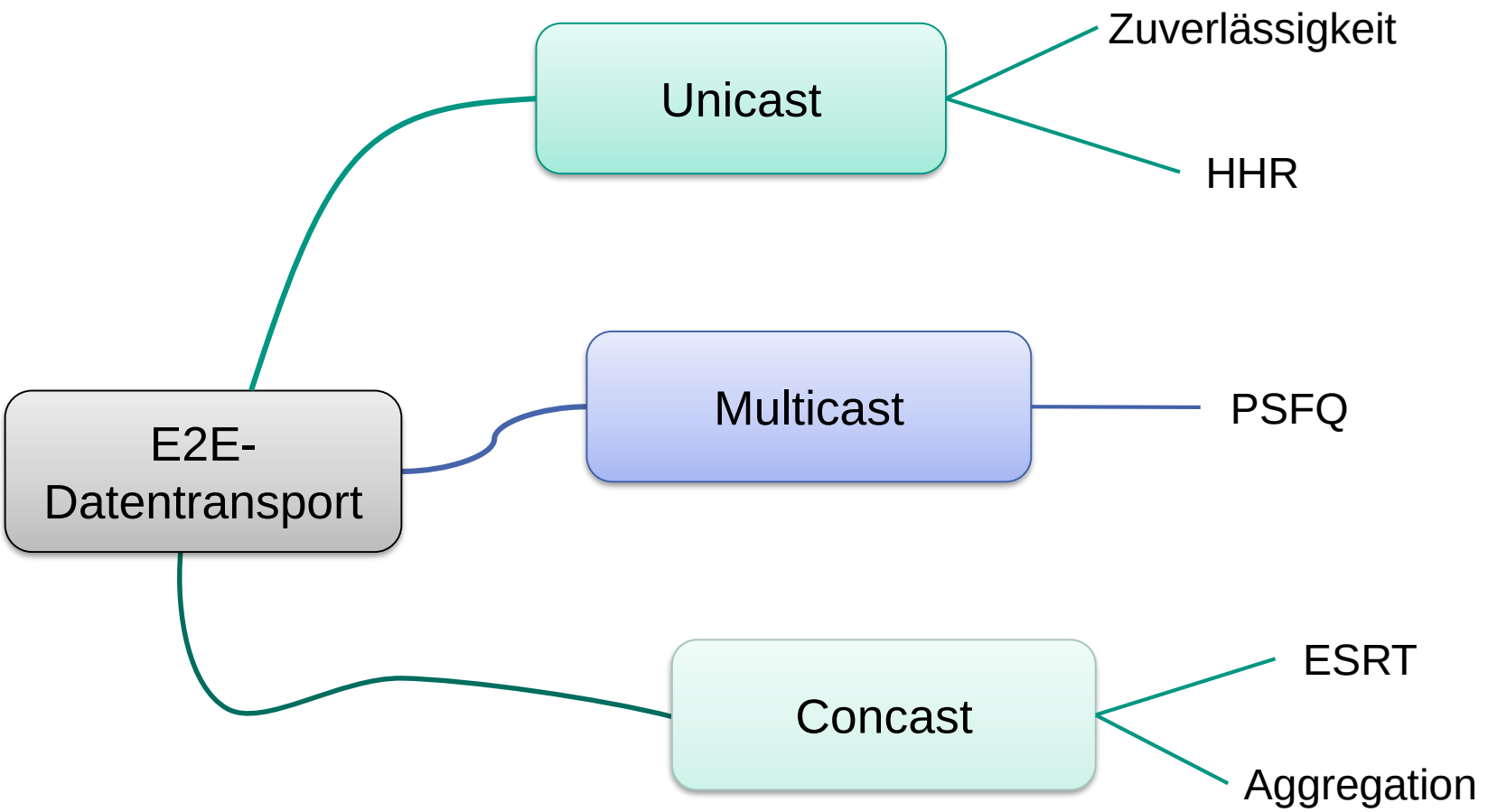
Inhalte der Vorlesung



Kapitel 4.4: Datentransport



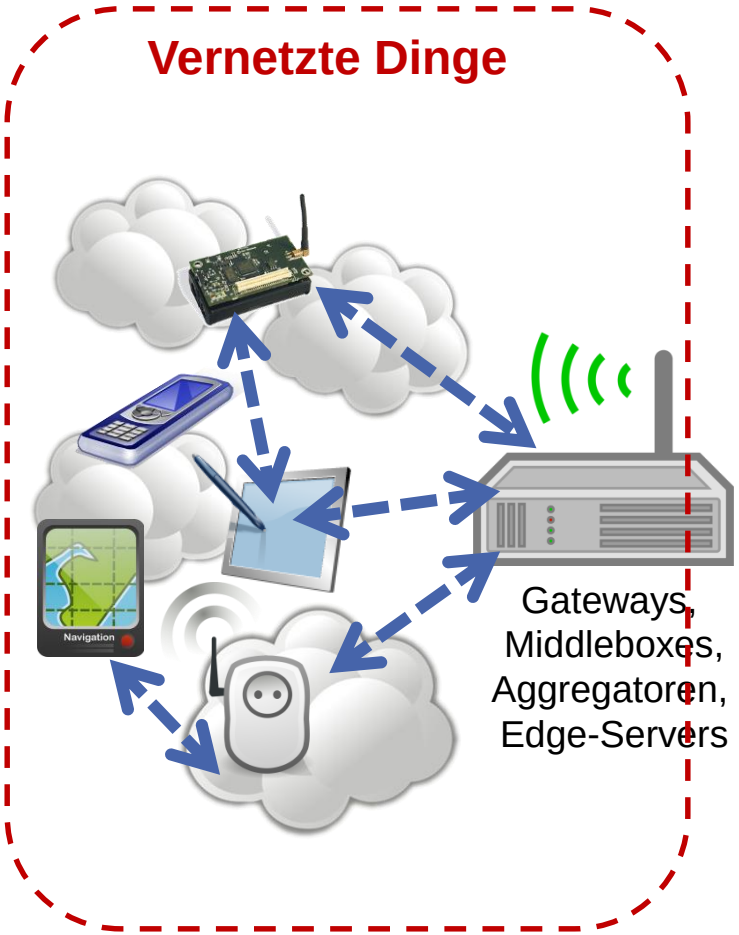
Schwerpunkte des Kapitels im Überblick



Einordnung

- Hier betrachtet: der drahtlose Zugangsbereich
  - i.d.R. kein klassischer Ende-zu-Ende-Transport
- Außerdem
  - Datentransport oft mit Routing verknüpft, z.B.
    - Directed Diffusion
    - Rumor Routing

  → Klassisches Schichtenmodell greift nicht unbedingt



Besonderheiten, die zu berücksichtigen sind

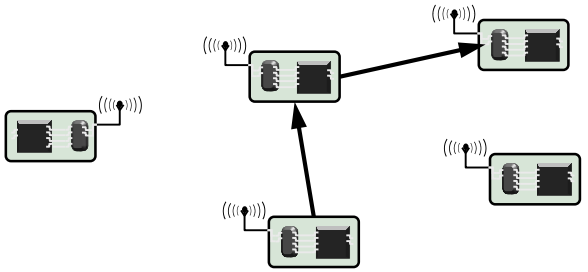
- Zuverlässigkeit
  - Drahtloses Medium
    - Paketfehlerraten bis zu 50% je nach Umgebung
    - Sendewiederholungen, FEC → Redundanz
  - „Unzuverlässige“ Systeme
    - Schlafzustände, einfache Hardware, …
- Beschränkte Ressourcen
  - Energie
    - So wenig wie möglich senden, Wiederholungen vermeiden → ARQ potentiell ineffizient
    - „Kurze“ Dateneinheiten → FEC potentiell ineffizient
  - CPU
    - Nur einfache (FEC-)Verfahren möglich
  - Speicher
    - Möglichst wenig Daten sollten zwischengespeichert werden

Besonderheiten, die zu berücksichtigen sind

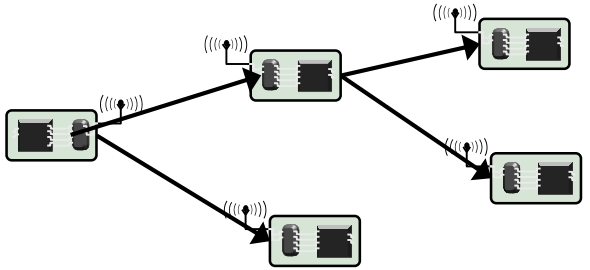
- Zeitliche Anforderungen
  - Daten potentiell schnell veraltet, oft Realzeitanforderungen
    - Sendewiederholungen evtl. zu langsam?
- Netztopologie und Kommunikationsmuster
  - Keine klassische Client/Server-Struktur, viele Sensoren/Aktoren, eine/wenige Senken:
    - Concast: Viele Sensoren schicken Messdaten an eine Senke
    - Multicast: Senke sendet Anfragen an viele Sensoren/Aktoren
  - Daten(vor)verarbeitung im Netz (In-Network-Processing)
    - Keine klassische Ende-zu-Ende Sichtweise

Typische Kommunikationsformen

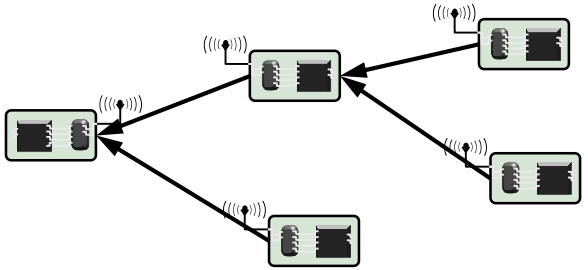
- Sensoren → Aktoren
  - Unicast
  - Herausforderung: Zuverlässigkeit



- Senke → Sensoren/Aktoren
  - Typische Daten
    - Kontroll- oder Steuerungsinformationen
    - Code-Updates
  - Multicast/Broadcast
  - Herausforderung: Zuverlässigkeit

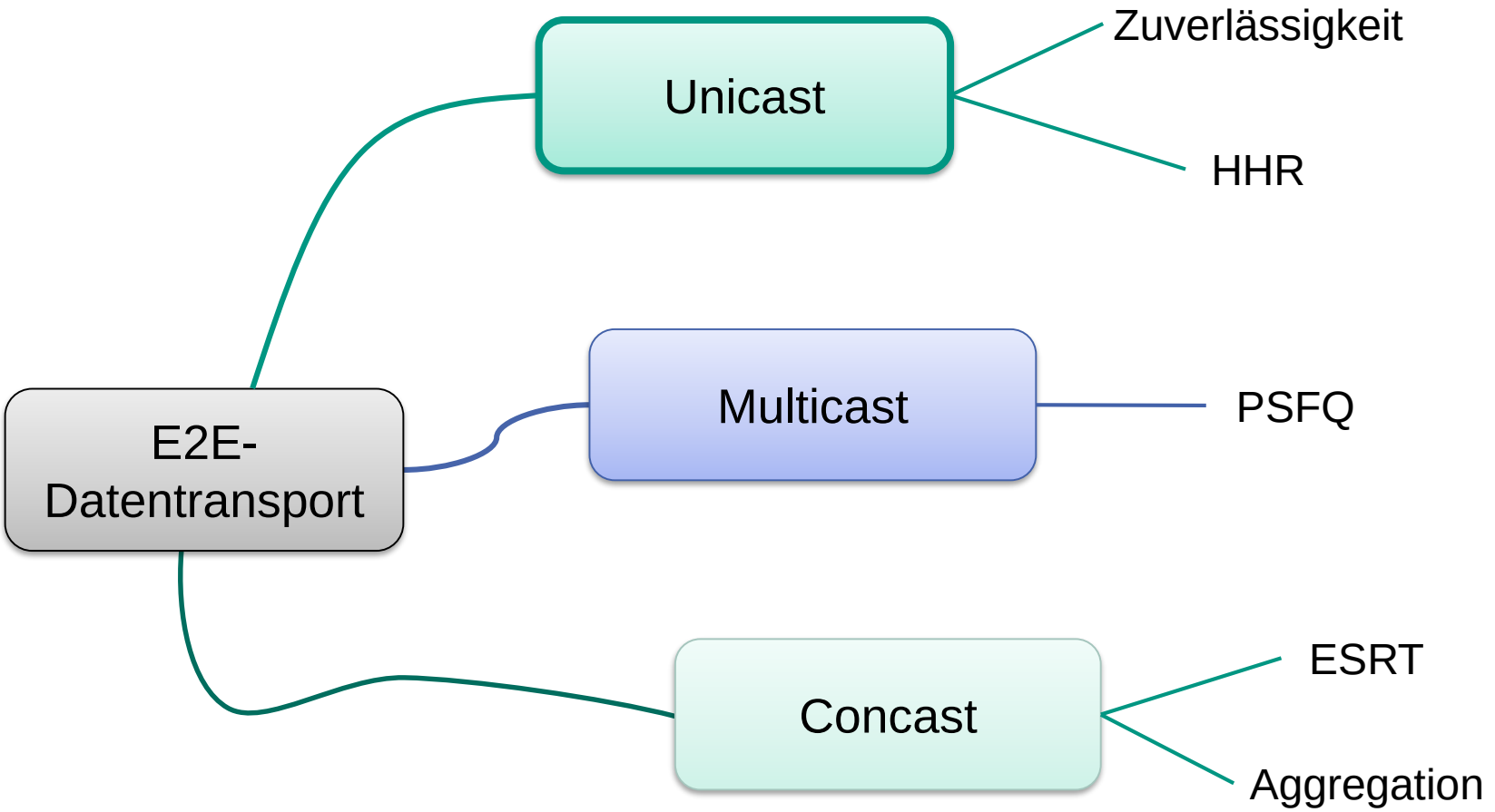


- Sensoren → Senke
  - Typische Daten
    - Sensormessdaten
  - Concast
  - Herausforderung: Zuverlässigkeit + Stau



Unicast

- Anwendungsbeispiele
  - Hausautomation: Lichtschalter steuert Lichtquelle
  - Industrie: Überwachungssensor stoppt Produktionsanlage
  - Autowäsche: Abstandssensor verhindert Beschädigungen am Auto
- Generell
  - Meistens Sensor → Aktor
- Herausforderungen
  - Hohe Zuverlässigkeit
  - Geringe Verzögerung


E2E-
Datentransport
Unicast
Zuverlässigkeit
HHR
Multicast
PSFQ
Concast
ESRT
Aggregation

Zuverlässigkeit - traditionell

- Zuverlässiger Transportdienst (z.B. durch TCP)!
  - Alle Dateneinheiten kommen an
  - Dateneinheiten kommen unverändert an
  - Dateneinheiten kommen reihenfolgetreu an
  - Dateneinheiten kommen duplikatfrei an
  - Keine Phantom-Dateneinheiten

- Protokollmechanismen für zuverlässige Transportdienste
  - Zeitgeber
  - Quittungen
  - Sequenznummern
  - Sendewiederholungen (Automatic-Repeat-Request, ARQ)
    - Z.B. Stop-and-Wait
  - Vorwärtsfehlerkorrektur (Foward Error Correction, FEC)
    - Redundante Übertragung von Daten

Warum nicht einfach TCP?

- TCP erkennt Ursache für Paketverlust nicht
  - Nimmt an, dass Stausituation existiert
    - Führt zu schlechter Performance
- TCP liefert 100% Zuverlässigkeit
  - Viele Wiederholungen → Energiebedarf hoch
  - Head-of-Line Blocking … ggfs. hohe Latenzen
    - …unnötig/hinderlich für viele Anwendungen des IoE
  - Daten werden beim Transport im Netz nicht verändert
    - … kein In-Network-Processing möglich
- TCP ist verbindungsorientiert
  - 3-Wege-Handshake … hohe Latenzen beim Verbindungsaufbau
    - … für spontane Übertragung von Ereignissen eher nicht geeignet
- TCP nur für Unicast ausgelegt

Zuverlässigkeit?

- Wie viel Zuverlässigkeit?
  - 100% Zuverlässigkeit realisierbar / notwendig?
- Welche Protokollmechanismen?
  - ACK oder NACK, FEC, …?
- Welcher Kontext?
  - Zuverlässigkeit Hop-by-Hop oder Ende-zu-Ende?

- Nicht alle Messwerte sind für Senke(n) wichtig
  - Z.B. durchschnittliche Temperatur im Raum

  → Berechnung des Durchschnittswertes im Netz → geringeres Datenaufkommen im Sensornetz → Energieersparnis?!
    - Auch als In-Network Processing bezeichnet

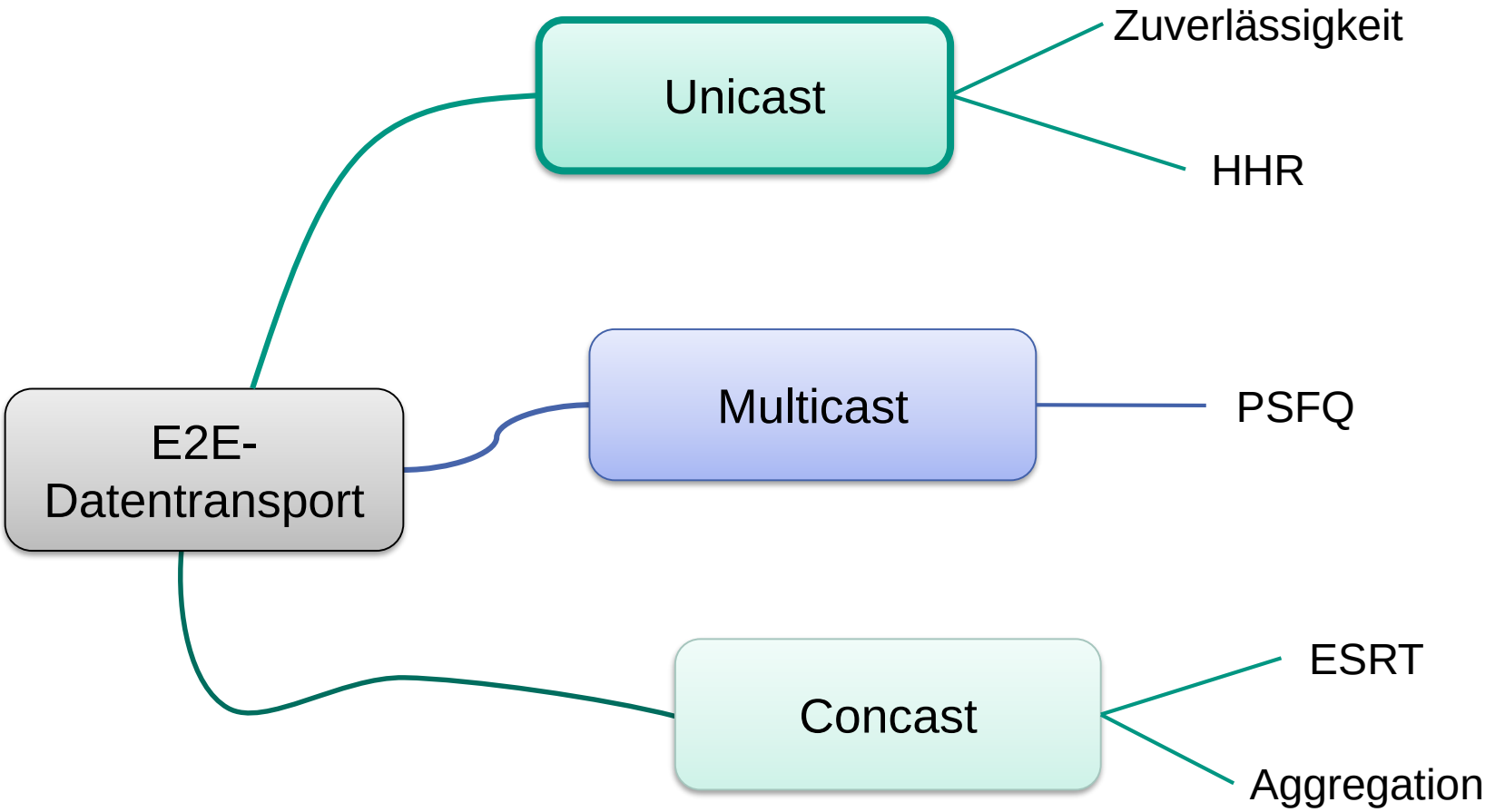
  → Keine klassische Ende-zu-Ende Sichtweise mehr!

- Ungenauigkeiten können (anwendungsabhängig) in gewissem Rahmen toleriert werden
  - Z.B. Schwankungen bei der durchschnittlichen Temperatur von 0,1 Grad Celsius
  - Andere Formen von Zuverlässigkeit möglich, z.B. probabilistisch

Probabilistische Zuverlässigkeit

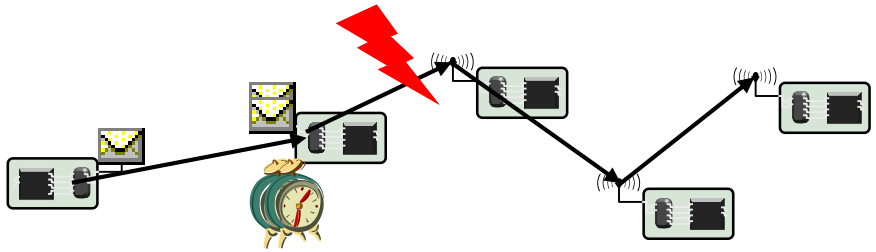
- Nur ein Teil (%) aller Dateneinheiten erreicht Senke
  - Einzelne Dateneinheiten erreichen Senke mit Wahrscheinlichkeit p
    - Aufgrund schlechter Funkverbindungen, Systemausfall etc.
    - Verbraucht weniger Energie, als alle Dateneinheiten zu transportieren
  - Teil der Dateneinheiten ausreichend bei redundanter Information
    - z.B. bei periodischem Messen

- Genauigkeit
  - Information, die Senke erreicht, stimmt nicht exakt mit der erfassten Information überein
    - Z.B. unscharf/ungenau
      - Teil der Information durch Paketverlust verloren
      - Information im Netz aggregiert (z.B. Durchschnitt berechnet
  - Information, die Senke erreicht ist veraltet
    - lange Verzögerung auf Grund schlechter Funkverbindungen

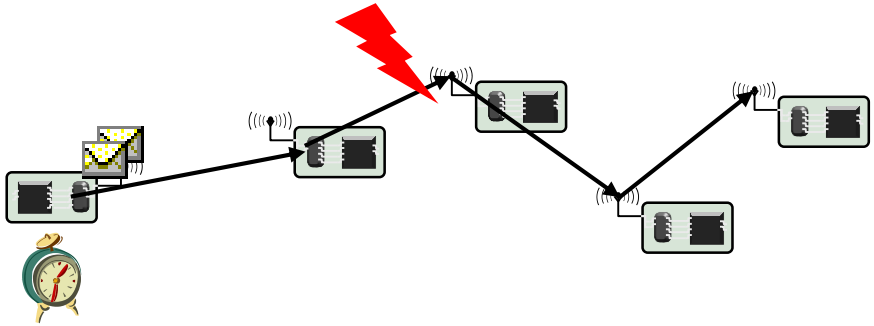

E2E-
Datentransport
Unicast
Zuverlässigkeit
HHR
Multicast
PSFQ
Concast
ESRT
Aggregation

Zuverlässigkeit: Pro Hop vs. Ende-zu-Ende

- Bei Quittungen pro Hop („Link Layer“)
  - Jeweiliges Nachbarsystem (Vorgänger) wiederholt Dateneinheit



- Bei „Ende-zu-Ende“ Quittungen
  - Nur ursprünglicher Sender wiederholt Dateneinheit



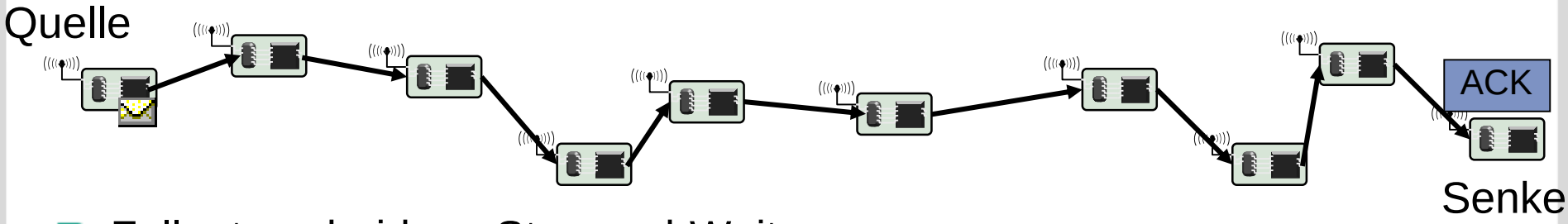
Zuverlässigkeit: Pro Hop vs. Ende-zu-Ende

- ACK vs. NACK, bei Verlust einer einzelnen Dateneinheit
  - Nur „positive“ Quittungen (ACK) sinnvoll
  - Negative Quittungen (NACK) schwierig
    - Empfänger weiß nicht, dass Dateneinheit verschickt wurde
    - Aber: bei periodischem Senden sinnvoll einsetzbar

- Aber welches System sendet Quittung?
  - Jedes benachbarte System auf Multi-Hop-Pfad („Link Layer“)
    - ⊕ Wiederholungen auf MAC-Schicht möglich → geringere Latenz
    - ⊖ Zustandshaltung auf Zwischensystemen nötig
    - ⊖ Viele Quittungen zwischen Nachbarn nötig, teuer
  - Nur das Empfängersystem („Ende-zu-Ende“)
    - ⊕ Kontrolle auf Transportschicht immer nötig bei zuverlässigem Dienst
    - ⊖ Bestimmen des Timers?
    - ⊖ Dateneinheit immer über gesamten Pfad wiederholt

Vergleichsszenario: Pro Hop vs. Ende-zu-Ende

- Abstraktes Szenario
  (ohne Betrachtung von Kollisionen, Medienzugriffen etc.)
  - Parameter: n=10 Systeme, 1 Quelle, 1 Senke, 8 Zwischensysteme
  - Quelle sendet 1 Datum über Zwischensysteme an Senke
  - Stop-and-Wait: Senke sendet ACK bei Empfang, Quelle sendet nach Empfang von ACK nächste Dateneinheit
    - Quelle wiederholt eigene Übertragung (Timeout), wiederholt unendlich oft



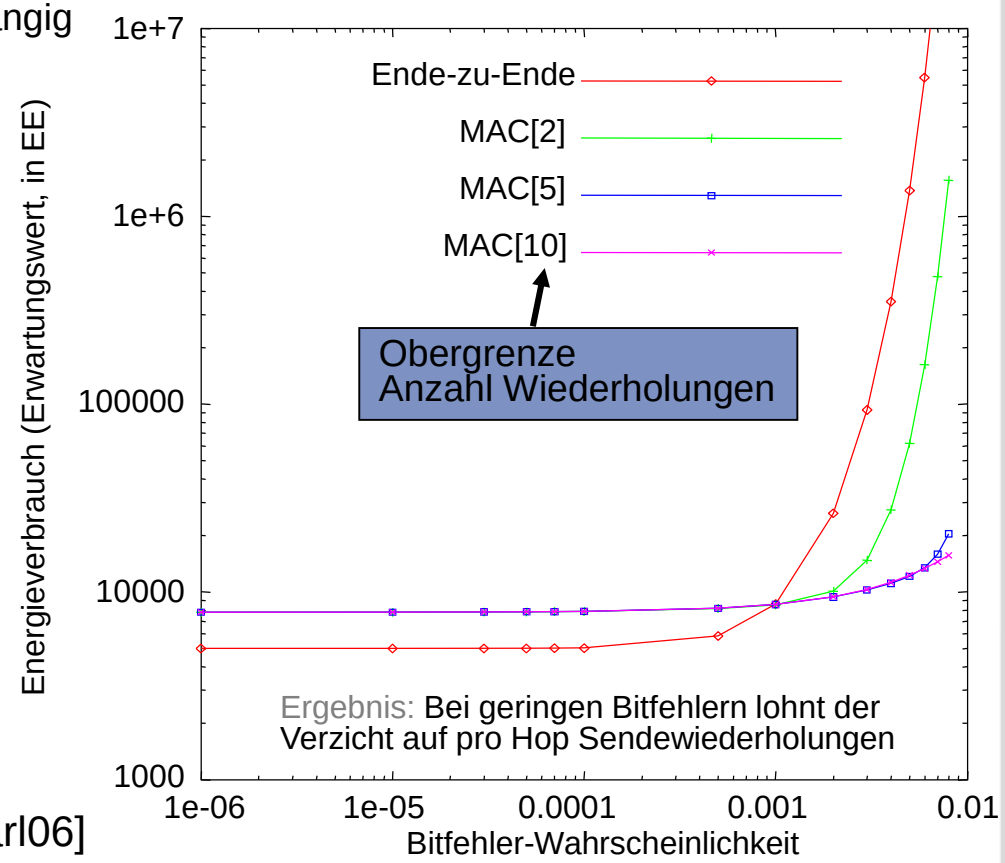
- Fallunterscheidung Stop-and-Wait
  - Nur Ende-zu-Ende ACKs/Wiederholungen
  - Ende-zu-Ende und ACKs/Wiederholungen pro Hop
    - „MAC[x]“: jeweils bis zu x Sendewiederholungen pro Hop

Energieverbrauch: Pro Hop vs. Ende-zu-Ende

- Drahtlose Kommunikation zwischen je 2 Systemen fehlerbehaftet
  - Modelliert durch „Binary Symmetric Channel“ (BSC)
    - Es gilt: Jedes Bit wurde mit gleicher Wahrscheinlichkeit p falsch übertragen
    - Alle Bits sind voneinander unabhängig falsch übertragen worden
    - Stochastisches Modell
      → Erwartungswert!

- Energieberechnung
  - Energiebedarf je Bit
    $e_t = e_r = 1\ EE$
  - Energiebedarf je Übertragung
    fixed energy = $50\ EE$
  - Länge der Dateneinheit
    $l = 100$ Bit
  - Acknowledgements
    ack(Senke) = 50 Bit
    ack(MAC) = 20 Bit

[Karl06]



Pro Hop vs. Ende-zu-Ende

- Ende-zu-Ende Quittungen immer nötig, aber
  - Pro Hop Sendewiederholungen sinnvoll
    - Gerade bei hohen Bitfehlerraten / „langen“ Routen
  - Je größer x bei MAC[x], desto später beginnt der exponentielle Anstieg im Energieverbrauch
  - Man kann zeigen
    - Verfahren mit MAC[∞] haben linearen Energieverbrauch bei steigender Anzahl Hops

- Weitere Vorteile von pro Hop Sendewiederholungen
  - Energieverbrauch wird im Netz verteilt
  - Sender nicht in jede Sendewiederholung involviert

Hop-by-Hop Reliability (HHR)

- Zielsetzung
  - Erhöhe Zuverlässigkeit ohne Quittungen

- Ansatz
  - Erhöhe die Wahrscheinlichkeit für erfolgreichen Empfang
  - Ohne Quittungen und Sendewiederholungen
    - Statt dessen, sende Dateneinheit immer sofort k mal
  - Wähle k so, dass Dateneinheit mit Wahrscheinlichkeit r von Senke korrekt empfangen wird
    - $r_i = 1 - {p_i}^{k_i}$
  - Berechnung von k_i
    - mit Kenntnis der Paketfehlerrate p_i eines Systems i zum Nachbarsystem

[DBN03]

HHR mit Acknowledgements (HHRA)

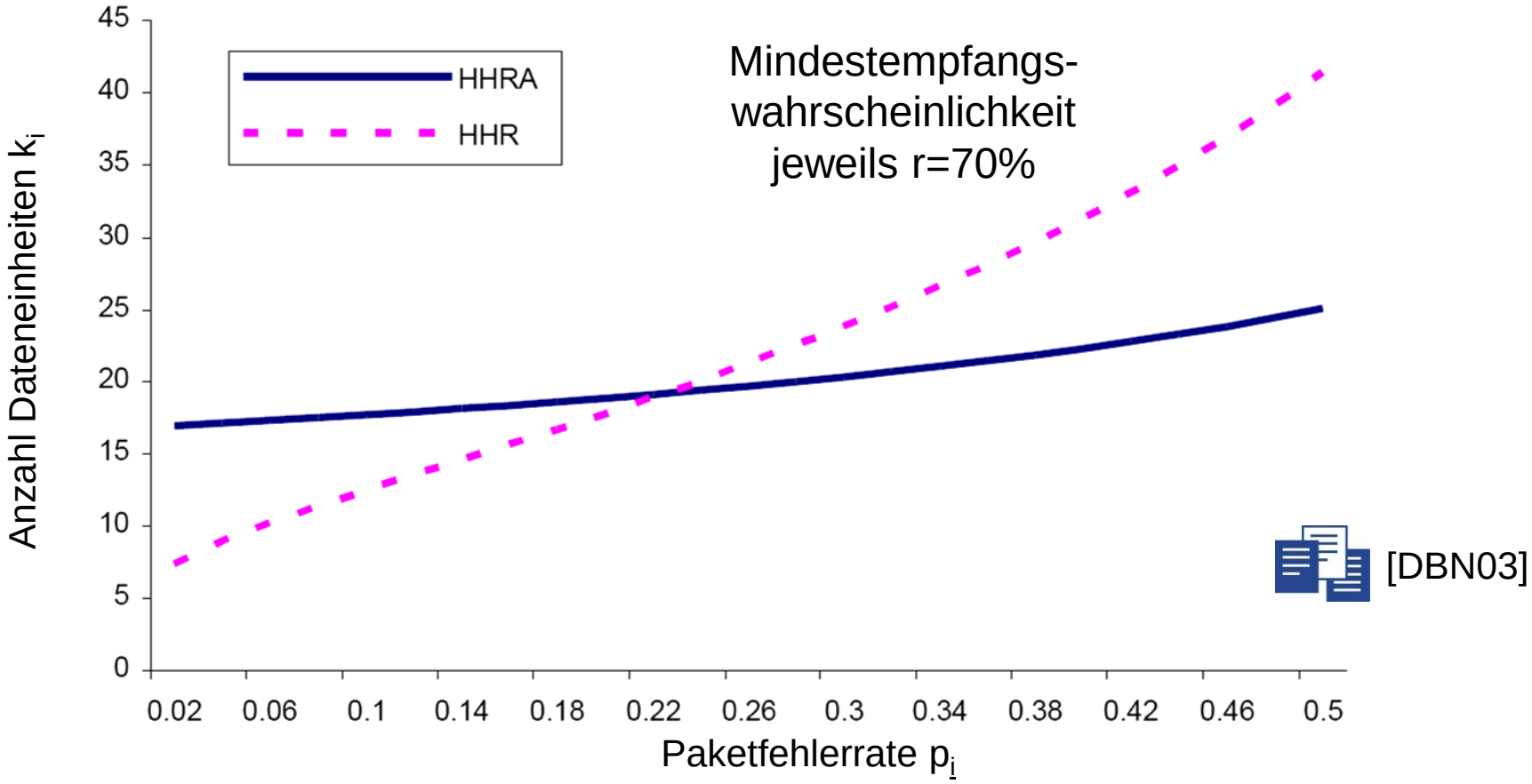
- HHRA
  - Sende bis zu k_i mal – warte aber jedes Mal auf Quittung
  - Höre auf zu senden, sobald erste Quittung empfangen oder k_i erreicht

- Vergleich von HHR und HHRA
  - Simulation (ohne Betrachtung von Kollisionen, Medienzugriffen, etc.)
  - $n = 10$ Systeme
    - 1 Quelle, 1 Senke, 8 Zwischensysteme
  - Wahrscheinlichkeit, dass Senke Dateneinheit empfangen soll, betrage $r = 70\%$
  - Quelle will eine Dateneinheit an Senke schicken
  - Gemessen
    - Anzahl der insgesamt gesendeten Dateneinheiten mit HHR bzw. HHRA

[DBN03]

Evaluierung

- Bei steigendem p_i steigt k_i → mehr Dateneinheiten werden gesendet
- Pro Hop Quittungen lohnen sich erst bei hohen Paketfehlerraten

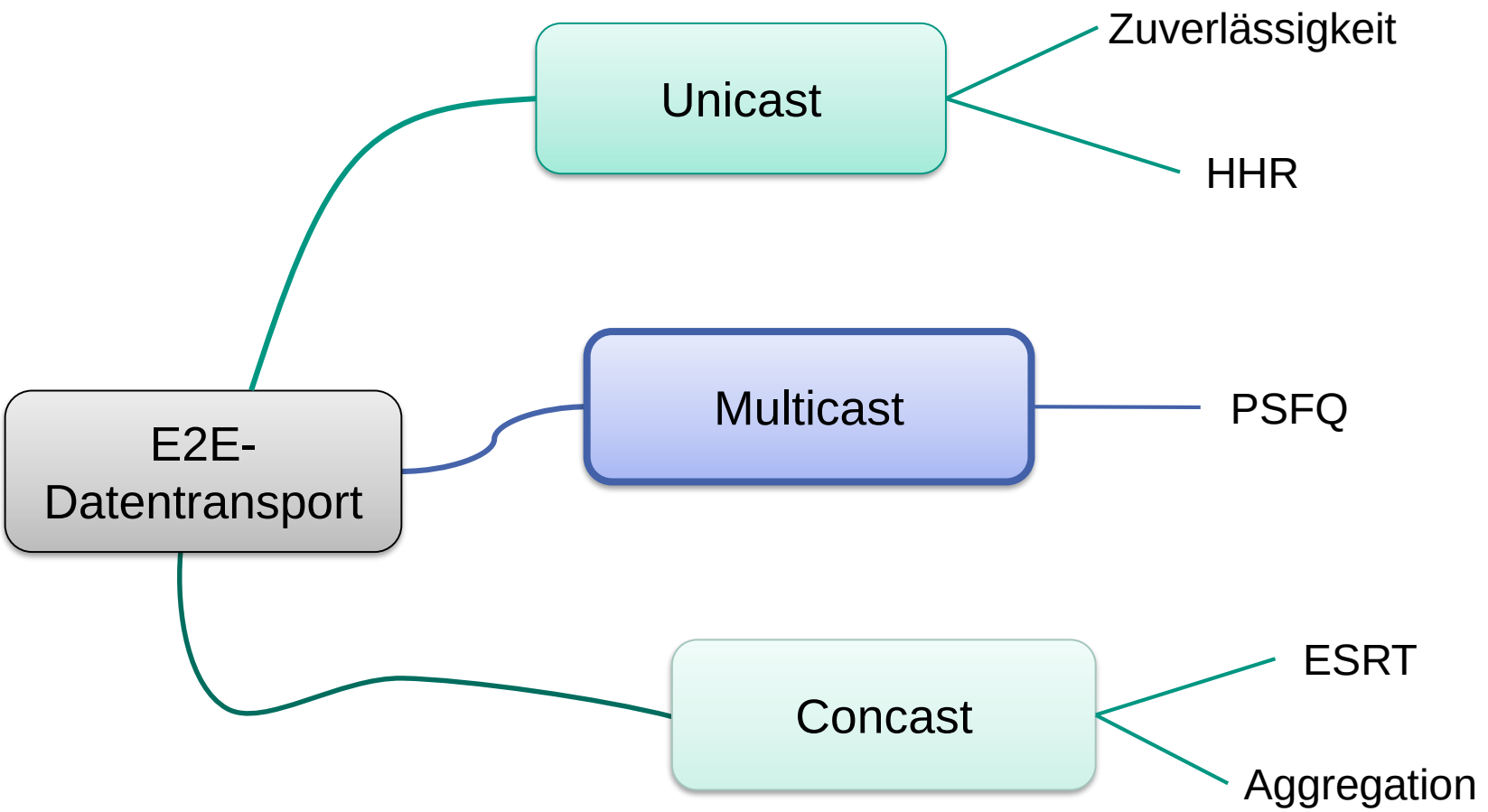


[DBN03]

HHR vs. HHRA

- HHR
  - Vorteil
    - Einsparen von Übertragungen durch Verzicht auf Quittungen
    - Bei niedrigen Paketfehlerraten werden Dateneinheiten weniger oft übertragen
  - Nachteil
    - Paketwiederholung stoppt nicht, sobald Dateneinheit erfolgreich empfangen wurde

- HHRA
  - Nachteil
    - Overhead der Quittungen bei HHRA lohnt sich bei geringen Paketfehlerraten nicht
  - Vorteil
    - Bei hohen Paketfehlerraten wird auf weitere Sendevorgänge nach erfolgreicher Quittung verzichtet


E2E-
Datentransport
Unicast
Zuverlässigkeit
HHR
Multicast
PSFQ
Concast
ESRT
Aggregation

Multicast

- Anwendungsbeispiele
  - Anfrageverteilung
    - Basisstation stellt Anfrage an (Teil)menge von Sensoren
  - Befehlsverteilung
    - Basisstation schickt Kommando an (mehrere) Aktoren
  - Code-Update: Menge von Sensoren erhält neuen Programmcode

- Generell
  - 1 Basisstation → n Sensoren/Aktoren

- Herausforderung
  - Zuverlässigkeit

Pump Slowly Fetch Quickly (PSFQ)

- Ziel: Daten von Basisstation zu einer Menge Sensoren transportieren
  - Beispiel: Code Update
    - Datenverlust inakzeptabel, aber Latenz nicht so kritisch
  - Annahme: Routing existiert

[WCK02]

- Grundprinzip
  - Basisstation pumpt Dateneinheiten ins Sensornetz
    - Zwischen dem Versand von zwei Dateneinheiten wird gewartet
  - Systeme speichern Dateneinheiten, falls es eine „neue“ ist
    - Weiterleitung (pumpen) nur falls Sequenznummer konsistent, d.h. keine Dateneinheit fehlt
  - Falls Sequenznummer fehlen
    - Nach verlorenen Dateneinheiten anfragen, Fetch Operation (NACK mit fehlenden Sequenznummern versenden)
    - Vorgängersystem versendet verlorene Dateneinheiten neu
      - Empfangene Dateneinheiten deshalb immer puffern
    - Pumpen langsam (um Fetchen zu ermöglichen), Fetch schnell

Lokaler Reparatur-mechanismus

Details zu PSFQ

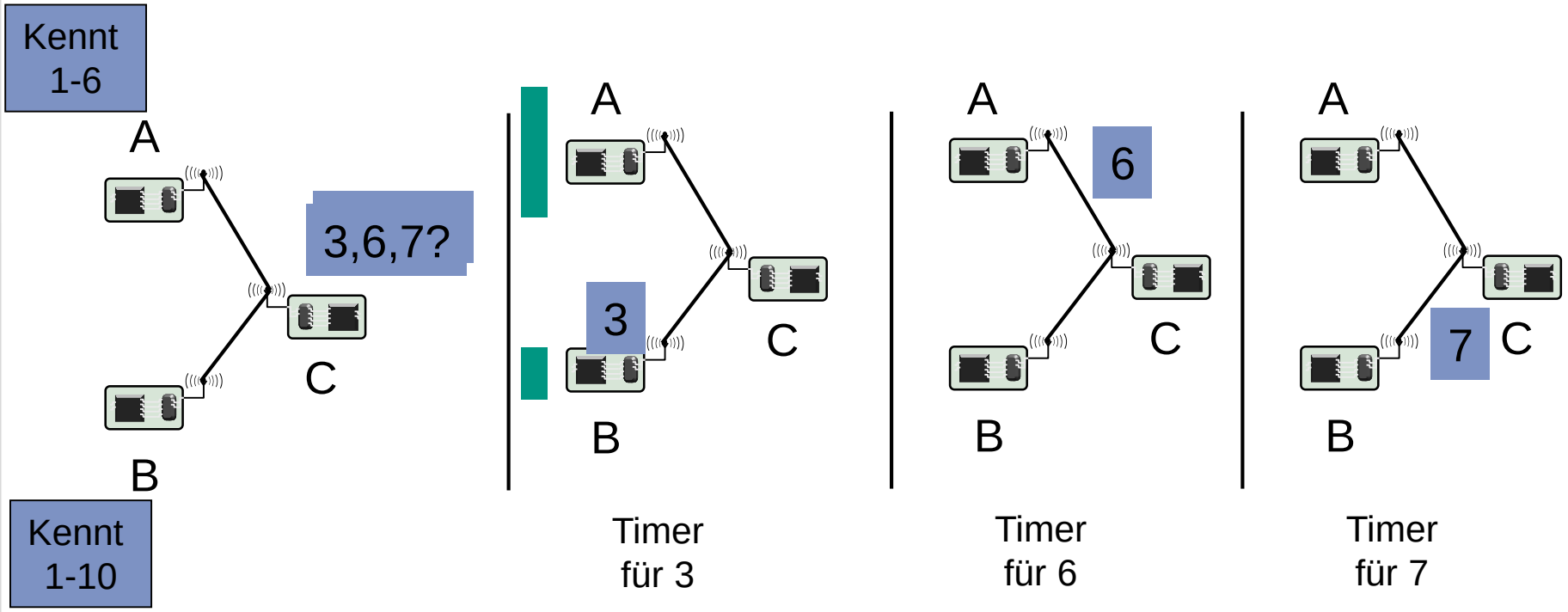
- Verzögerung zwischen zwei Pump-Operationen groß
  - Vorteil: ein oder mehrere Fetches dazwischen durchführbar
  - Wahrscheinlichkeit klein, dass neue Dateneinheit versendet wird, während alte noch nicht zuverlässig zugestellt
  - Vorschlag: Ermögliche 5 Fetch Operationen
- Probabilistisches Weiterleiten von Dateneinheiten mit aufeinanderfolgenden Sequenznummern
  - Warte zufälliges Zeitintervall t $(T_{min} < t < T_{max})$
  - Höre währenddessen Medium ab
  - Leite nur dann weiter, falls ≤ 3 Nachbarn die gleiche Dateneinheit weitergeleitet haben
    - „Nachbarabhängiges Fluten“: Ab 4 Nachbarn erhöht sich Netzabdeckung nur um $\leq 0.05\%$ durch zusätzliche Weiterleitung!
- Dateneinheiten mit falschen Sequenznummern
  - Nicht weiterleiten, fehlende Dateneinheiten holen (fetch)

Details zu PSFQ

- Fetch-Anfragen (= NACKs)
  - NACKs enthalten Liste fehlender Sequenznummern (z.B. 3, 6, 7)
  - NACKs sind Broadcasts
    - Erreichen mehrere Nachbarsysteme
    - Nachbarn empfangen NACK, kennen u.U. selbst nicht alle fehlenden Dateneinheiten
  - Idee: Für jede fehlende Dateneinheit im NACK
    - Systeme, die fehlende Dateneinheit kennen, starten Timer t und warten $(0 < t < T_r)$, Abhören des Mediums während dieser Zeit
    - Falls fehlende Dateneinheit vor Ablauf des Timers auf Medium gehört → Timer abbrechen
    - Falls Dateneinheit nicht gehört, Dateneinheit senden
    - …mit nächster fehlender Dateneinheit vom NACK fortfahren
  - So lange „fetchen“/NACKs senden, bis alle fehlenden Dateneinheiten empfangen, warte dazwischen jeweils T_r
    - …oder obere Schranke k von NACKs überschritten

Beispiel

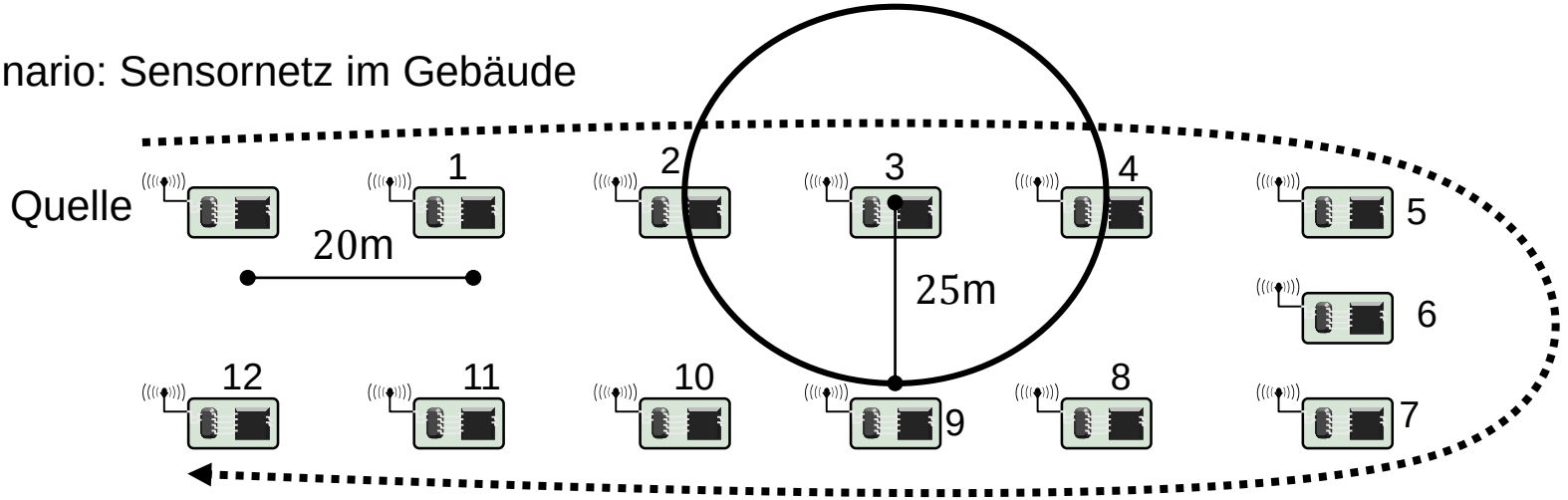
- System C fehlen Dateneinheiten 3,6,7
- C meldet fehlende Dateneinheiten per NACK



Evaluierung PSFQ

- Simulation, Quell-System sowie 12 „Empfänger“-Systeme
  - 802.11 CSMA/CA
  - 100 m x100 m Simulationsfläche
  - 2 Mbit/s Datenrate, 25 m Funkreichweite
  - Ziel: 2,5 KByte Daten sollen an alle Systeme im Netz verschickt werden
    - Dateneinheit jeweils 50 Byte groß → insgesamt 50 Dateneinheiten
    - Quelle verschickt alle 10 ms neue Dateneinheit
  - $T_{min} = 50$ms, $T_{max} = 100$ms, $T_r = 20$ms, max. $k = 5$ NACKs pro System

Szenario: Sensornetz im Gebäude

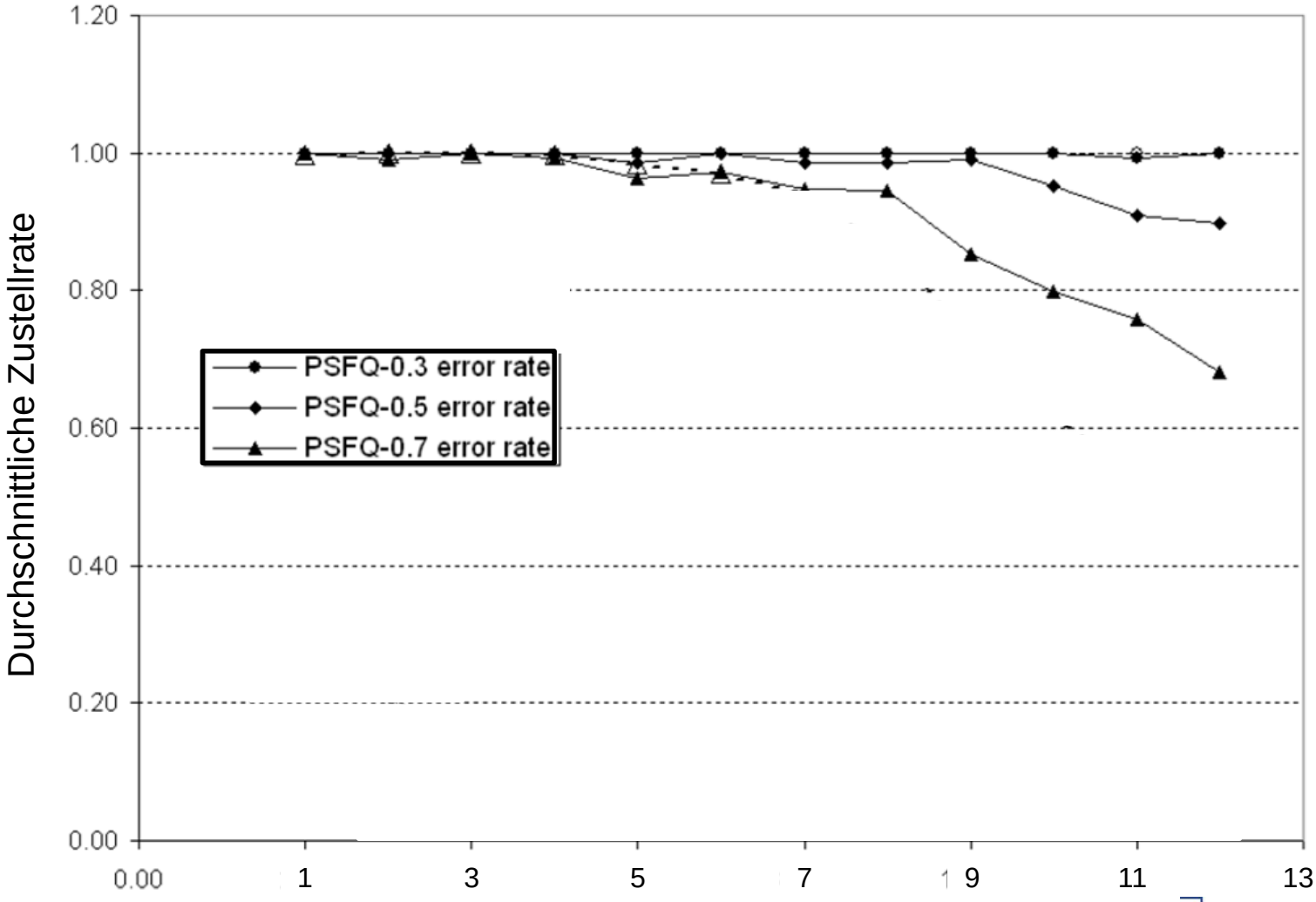


Evaluierung PSFQ

- Annahme wieder: Paketfehlerrate p
- Variiere p
  - Jeweils 10 Durchläufe pro Parameter
- Dabei gemessen: Durchschnittliche Zustellrate
  - Prozentsatz der Dateneinheiten, die bei einem der 12 Empfänger-Systeme angekommen sind

[FJMLZ95]

Evaluierung PSFQ

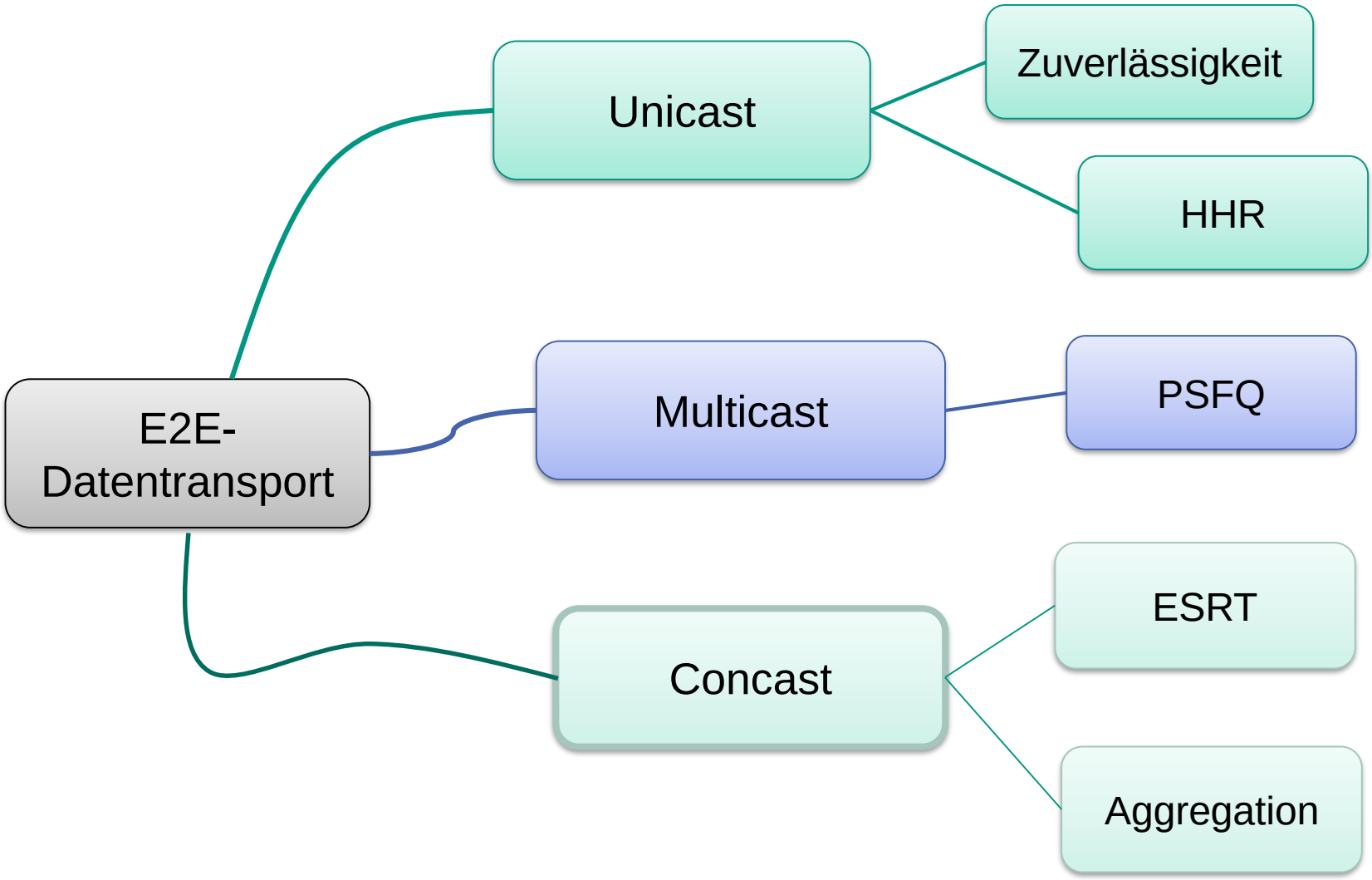


[WCK02]

Beobachtungen

- Je größer die Distanz zur Quelle, desto niedriger die Zustellrate
  - Systeme, die Dateneinheiten nicht empfangen, können diese auch nicht weiterleiten
- Je höher die Paketfehlerrate, desto niedriger die Zustellrate
  - Mehr Dateneinheiten und NACKs gehen verloren
- PSFQ erreicht selbst bei $p = 70\%$ noch Zustellraten von $> 70\%$

- Insgesamt
  - Keine 100% Zustellgarantie

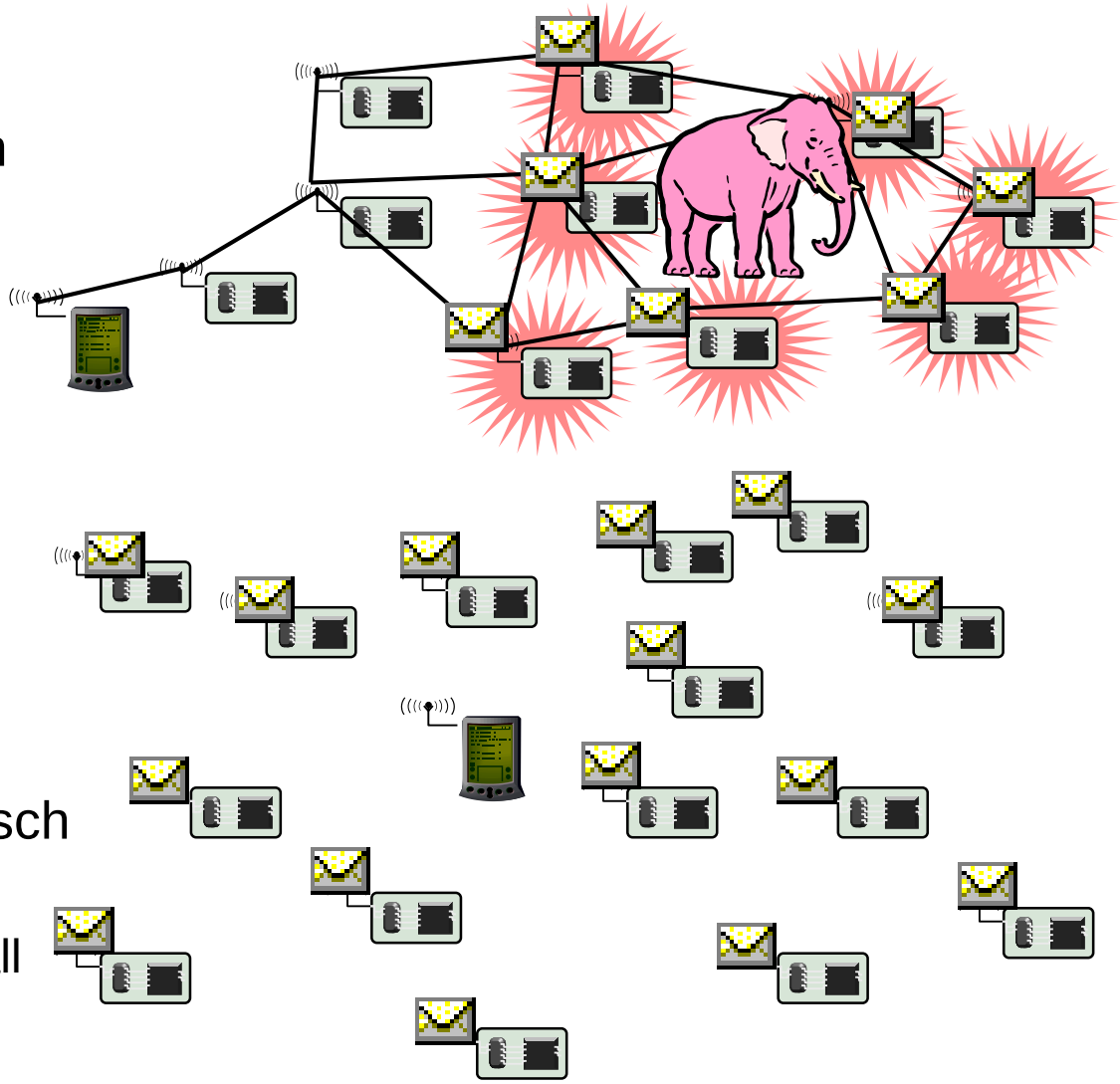

E2E-
Datentransport
Unicast
Zuverlässigkeit
HHR
Multicast
PSFQ
Concast
ESRT
Aggregation

Concast

- Anwendungsbeispiele
  - Ergebnistransport: Sensoren beantworten Anfrage mit Messdaten
  - Ereignistransport: Sensoren schicken spontane Beobachtung an Basis
  - Monitoring: Periodischer Concast von Messdaten
- Generell
  - n Sensoren/Aktoren → 1 Basisstation/Senke
- Herausforderungen
  - Zuverlässigkeit
  - Stau
  - Energie-Effizienz (insbesondere im periodischen Fall)

Stau

- Datenströme können zu Stau bzw. Überlast führen
- Einige Sensoren beobachten Ereignis und senden zur Senke
  - Hohe Systemdichte / Engpass
  - Stausituation durch Ereignis
- Viele verschiedene Sensoren senden periodisch an Senke
  - Zu kurzes Sendeintervall
  - Stausituation bei Senke



Auswirkungen von Stau

- Problem: Datenverluste
  - Gegenmaßnahme: Dateneinheit neu übertragen
  - Nachteil: Kostet zusätzlich Energie

  oder
  - Gegenmaßnahme: keine
  - Nachteil: Messdaten gehen verloren

- Problem: Daten kommen verzögert an
  - Gegenmaßnahme: Weniger Daten senden
  - Abhängig von Anforderungen der Anwendung!

ESRT (Event-to-Sink Reliable Transport)

- Szenario
  - Viele Sensoren senden periodisch Dateneinheiten zur Senke
  - Senke braucht ausreichende Anzahl von Dateneinheiten – egal von welchen Sensoren
- Ziel: in jeder Periode i mit Länge τ sollen ca. R Dateneinheiten zugestellt werden
- Ansatz: Anpassen der Senderate f_i zum Zeitpunkt i
  - Senke misst in Periode i die Anzahl r_i der empfangenen Dateneinheiten
  - Senke berechnet Zuverlässigkeit $\eta_i = r_i/R$
    - η_i in $[1-\varepsilon, 1+\varepsilon]$ soll gelten
- Senke berechnet f_{i+1} basierend auf f_i und η_i (siehe nächste Folien)
  - Dann: f_{i+1} wird per Broadcast an alle Systeme verschickt
    - Annahme: Senke kann alle Systeme durch hohe Sendeleistung erreichen

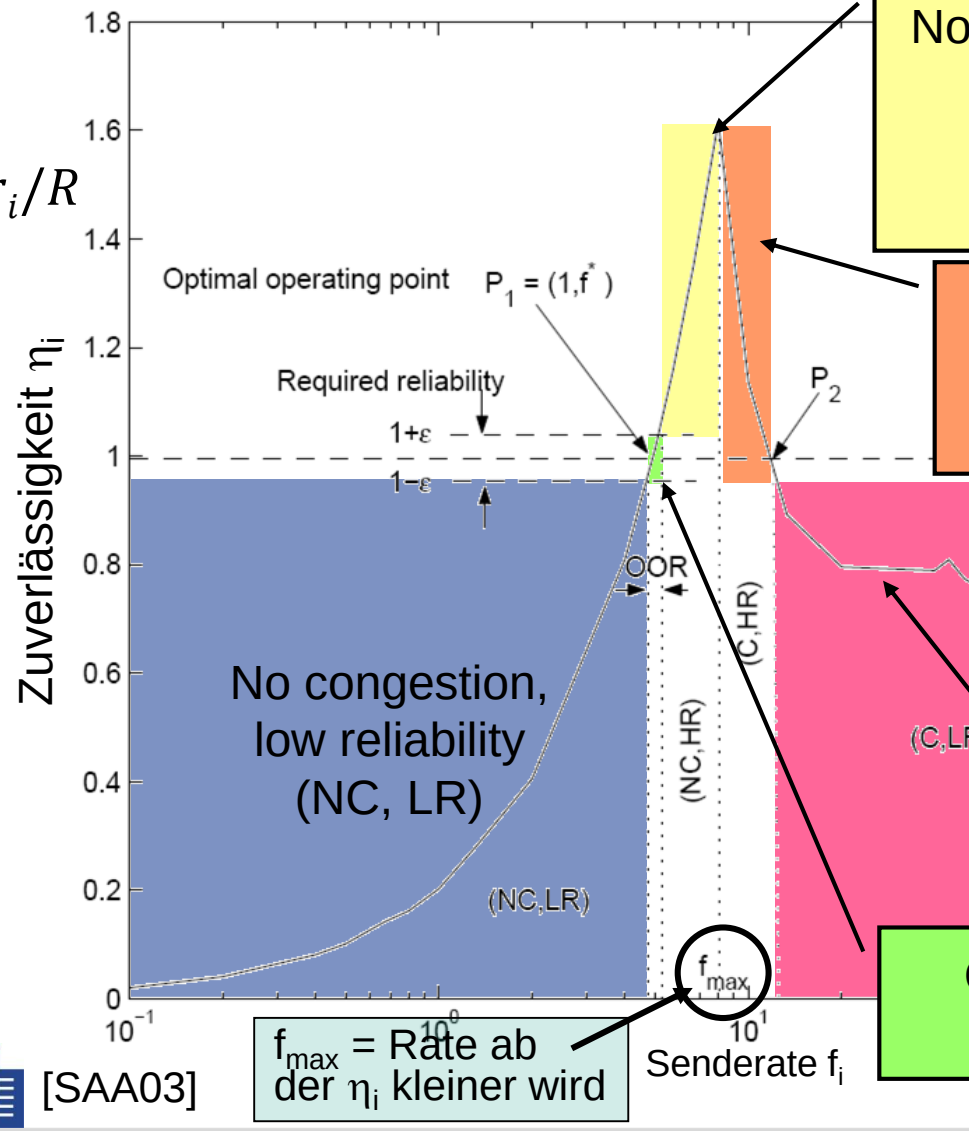
[SAA03]

Bestimmen von f_{i+1}

- Zu welchen Situationen kann es bei Stau im Netz überhaupt kommen?
- Beispielszenario, Simulation
  - 200 Systeme zufällig, gleichverteilt platziert
  - $100\ x\ 100$m Feld
  - CSMA/CA, Funkreichweite 40m
  - 81 Systeme senden periodisch an Senke
  - Variiere Senderate f_i
  - Gemessen: Zuverlässigkeit η_i in Abhängigkeit von f_i
- Senderate f_{i+1} wird bestimmt in Abhängigkeit
  - Der „Region“, in der sich f_i befindet (siehe folgende Folie)
    - Kein Stau, niedrige Zuverlässigkeit
    - Kein Stau, optimale Zuverlässigkeit
    - Kein Stau, hohe Zuverlässigkeit
    - Stau, hohe Zuverlässigkeit
    - Stau, niedrige Zuverlässigkeit

Bestimmen von f_{i+1}

$\eta_i = r_i/R$



No congestion, high reliability (NC, HR) – aber mehr Daten als notwendig

Congestion, High Reliability (C,HR)

Congestion, Low Reliability (C,LR) – Stau und schlechte Zuverlässigkeit

Optimal operating region (OOR)

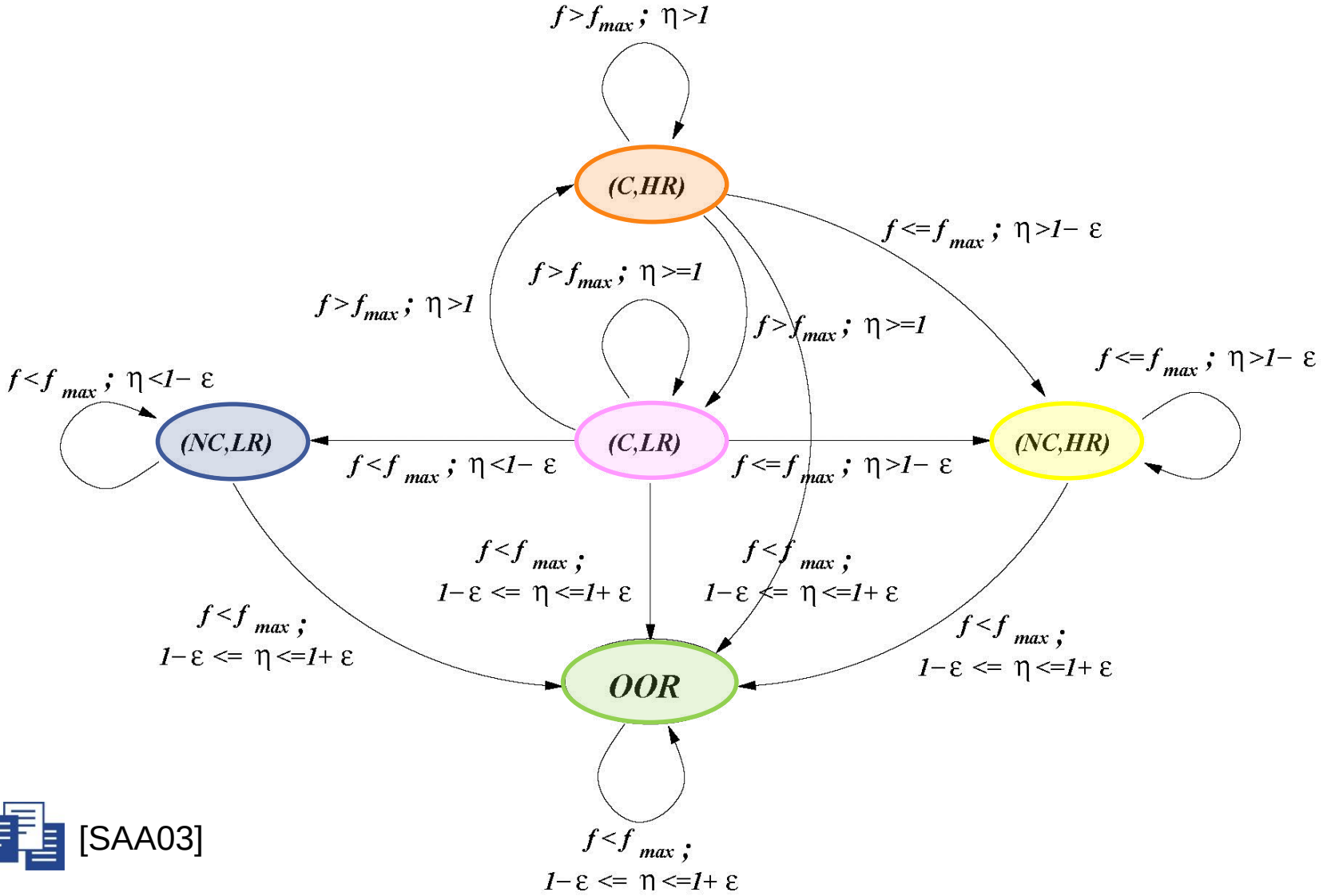
[SAA03]

Fallunterscheidung

Fall	Bedingung	$f_{i+1} =$
Kein Stau, Niedrige Zuverlässigkeit (NC, LR)	$\eta_i < 1, f_i < f_{\max}$	f_i/η_i
Optimales Verhalten (OOR)	$1 - \varepsilon \leq \eta_i \leq 1 + \varepsilon, f_i < f_{\max}$	f_i
Kein Stau, Hohe Zuverlässigkeit (NC, HR)	$\eta_i > 1, f_i \leq f_{\max}$	$f_i/2\,(1 + 1/\eta_i)$
Stau, Hohe Zuverlässigkeit (C, HR)	$\eta_i > 1, f_i > f_{\max}$	$f_i/\eta_i,$
Stau, Niedrige Zuverlässigkeit (C, LR),	$\eta_i \leq 1, f_i > f_{\max}$	$f_i^{\eta_i/k}$

- Anmerkungen
  - (NC, LR): $\eta_i < 1$ gilt, da weniger Dateneinheiten als erwünscht ankommen
  - (NC, HR): $\eta_i > 1$, da mehr Dateneinheiten als erwünscht ankommen
  - (C, LR): k ist Anzahl Perioden, in der das Netz in diesem Zustand bereits verweilt
  - Senke führt Fallunterscheidung durch Vergleich von η_i mit 1 und f_i mit $f_{\max}$ durch!

Zustandswechsel



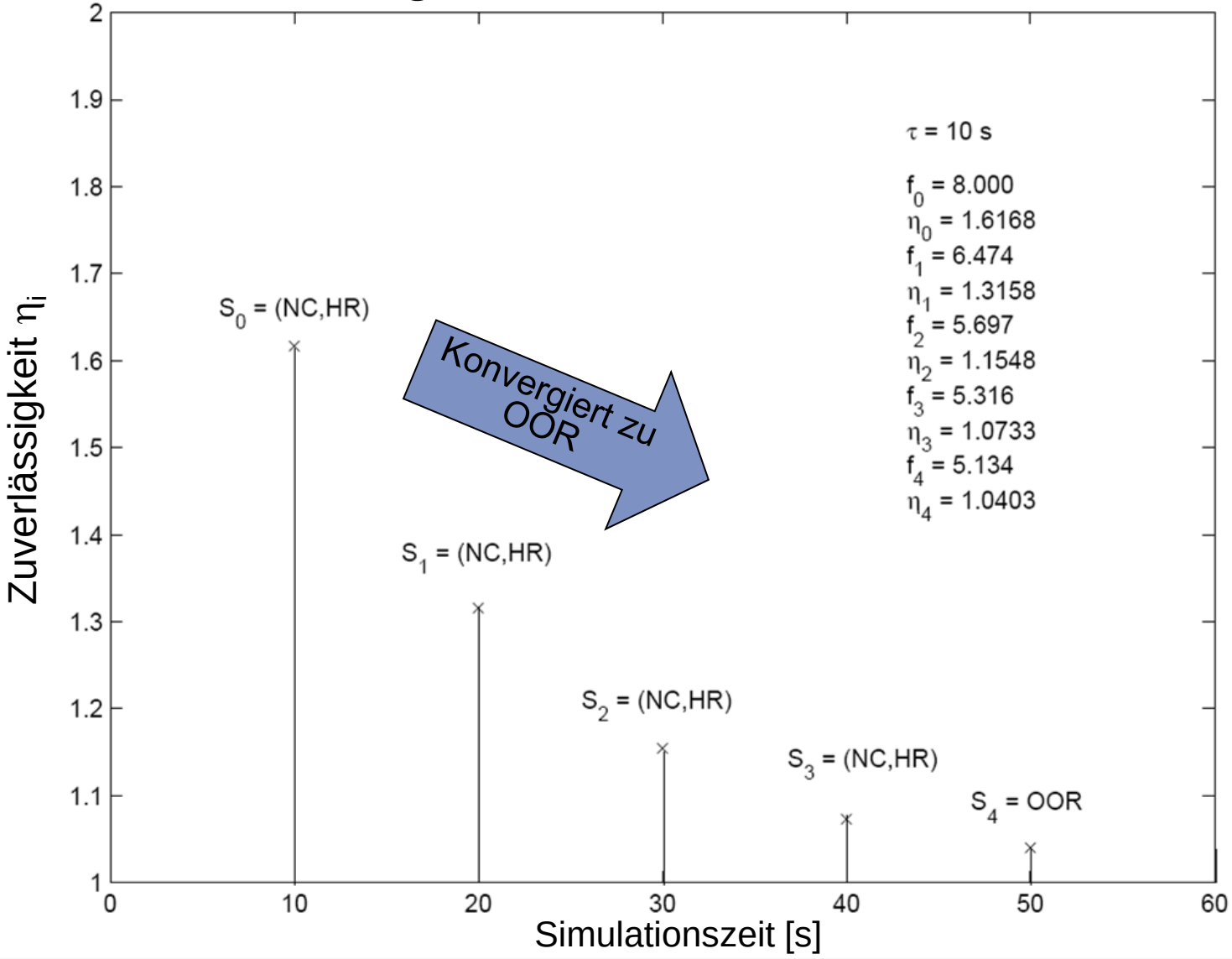
[SAA03]

Simulation

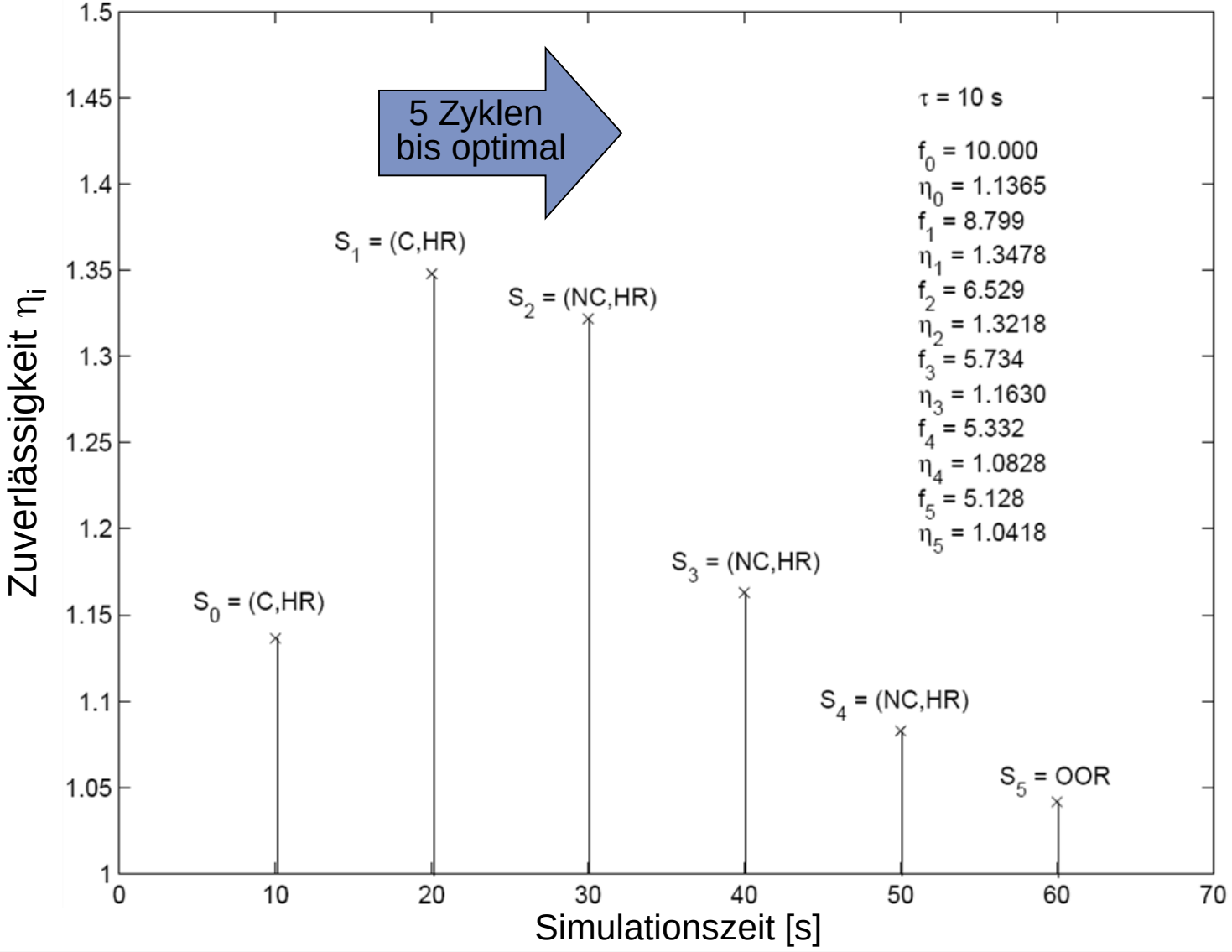
- Konfiguration
  - 200 Systeme zufällig, gleichverteilt platziert
  - 100 m x 100 m Feld
  - CSMA/CA, Funkreichweite 40 m
  - 81 Systeme senden periodisch an Senke
- τ = 10 s, ε = 5%

- Gemessen
  - Wie schnell konvergiert ESRT?
  - Variiere Region, in der sich f_i befindet

ESRT: Simulationsergebnisse



ESRT: Simulationsergebnisse



Beobachtung

- ESRT konvergiert immer in die OOR
  - Schnell, nur wenige Perioden notwendig

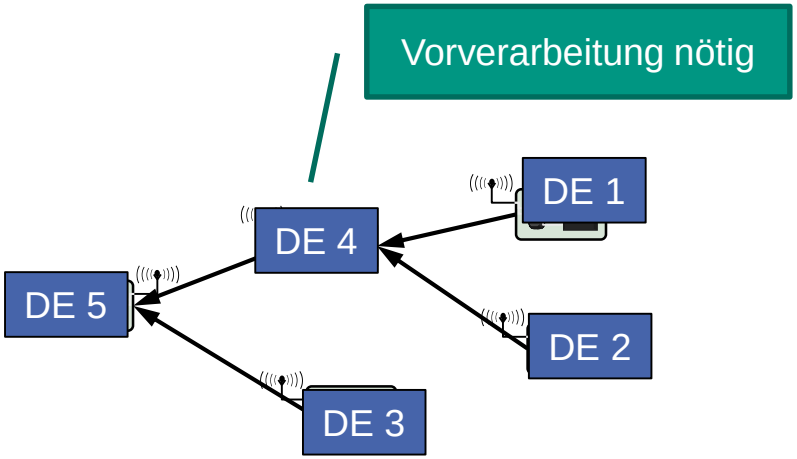
- Mögliche Pfade
  - (C, LR) → (NC, LR) → OOR
  - (C, HR) → (NC, HR) → OOR

Fazit

- Problem: Stau bzw. Überlast durch zuviel Kommunikation im Netz
  - Zu viele Systeme senden zu viele Dateneinheiten
  - Erkennen und Beheben von Stau in Sensornetzen schwierig
- Ansatz ESRT
  - Stau wird erkannt, wenn weniger Daten als gewünscht (R) die Senke erreichen
  - Dann: Dynamisches Anpassen der Senderate f durch Datensenke
- Wie bestimmt der Benutzer eigentlich R?
  - Abschätzen! → R muss groß genug sein, damit Informationen an der Senke genau genug sind
- Wie wirkt sich Paketfehlerrate aus? Was passiert bei Knotenausfällen?
  - Nicht untersucht!

Aggregation

- Anwendungsmöglichkeit bei Concast
- Ziel
  - Daten zusammenfassen und weniger Dateneinheiten verschicken



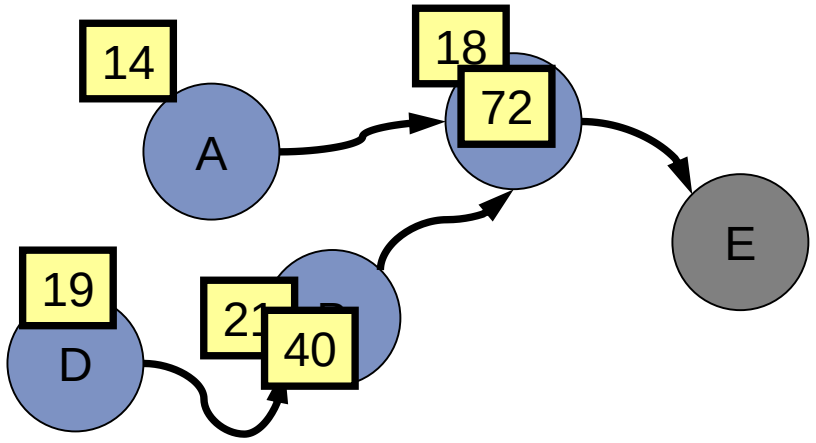
Concast

Aggregationsformen

- Keine Aggregation
  - Eingehende Daten werden direkt weitergeleitet, keine Vorverarbeitung
- Paketaggregation
  - Eingehende Daten werden zwischengespeichert
  - Gesammelte Daten werden in einem großen Paket gesammelt weitergeleitet
- Datenaggregation
  - Eingehende Daten werden zwischengespeichert
  - Gesammelte Daten werden mit Aggregationsfunktion zu einem Aggregat zusammengefasst, welches weitergeleitet wird

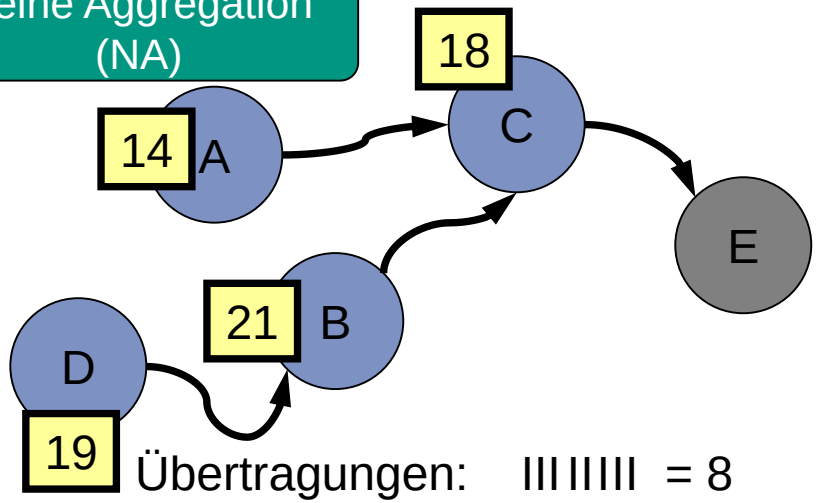
Aggregationsformen

Datenaggregation (DA)



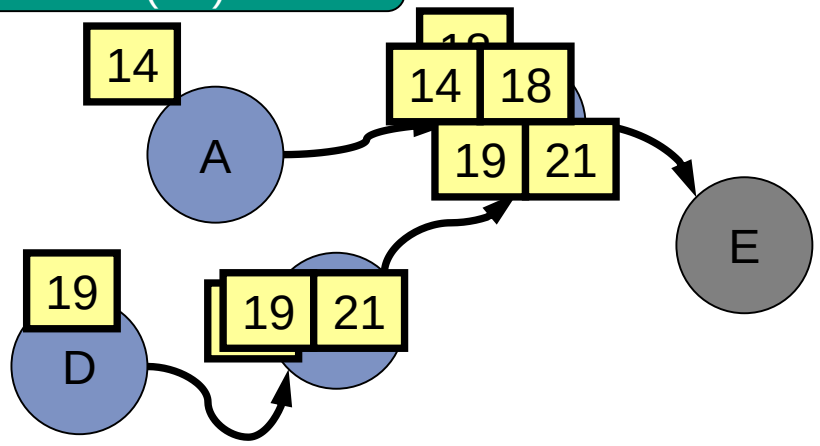
Übertragungen: IIII = 4

Keine Aggregation (NA)



Übertragungen: IIIIIIII = 8

Paketaggregation (PA)

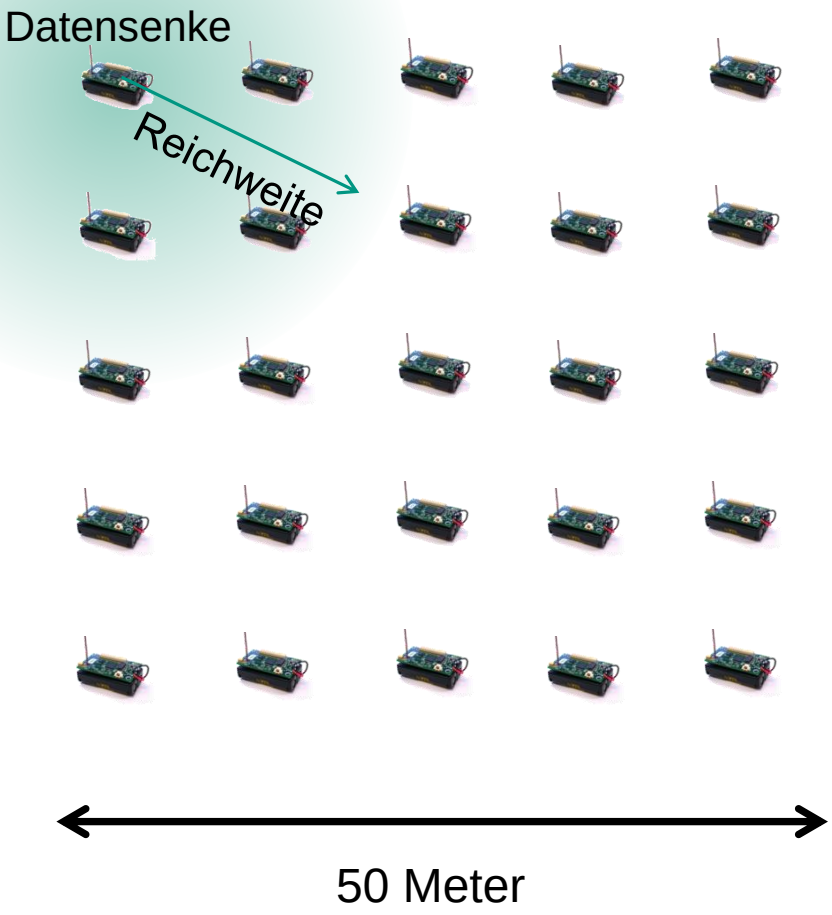


Übertragungen: IIII = 4

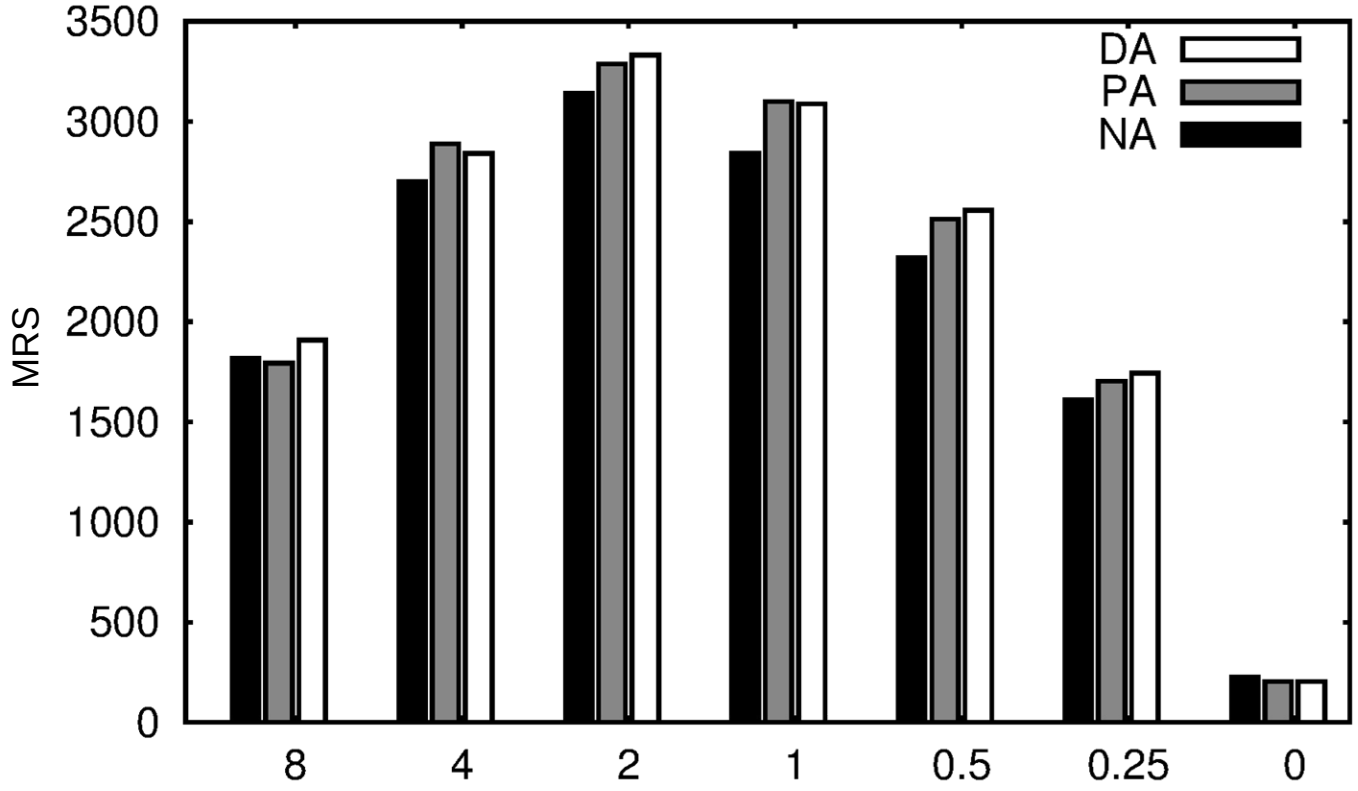
Jeder Knoten muss mit Aggregation nur eine Übertragung durchführen.

Beispiel

- Randbedingungen
  - Avrora+ Emulator für MICAz
  - 50 Seeds, randomisierter Start
  - 5 x 5 = 25 Sensorknoten auf 50 x 50 Metern
  - Knoten mit 50 Joule Energie ausgestattet
  - Periodendauer 1 Minuten
  - Verschiedene MAC-Protokolle
- Evaluations-Metrik: MRS
  - MRS = Anzahl Messwerte die Datensenke erreichen

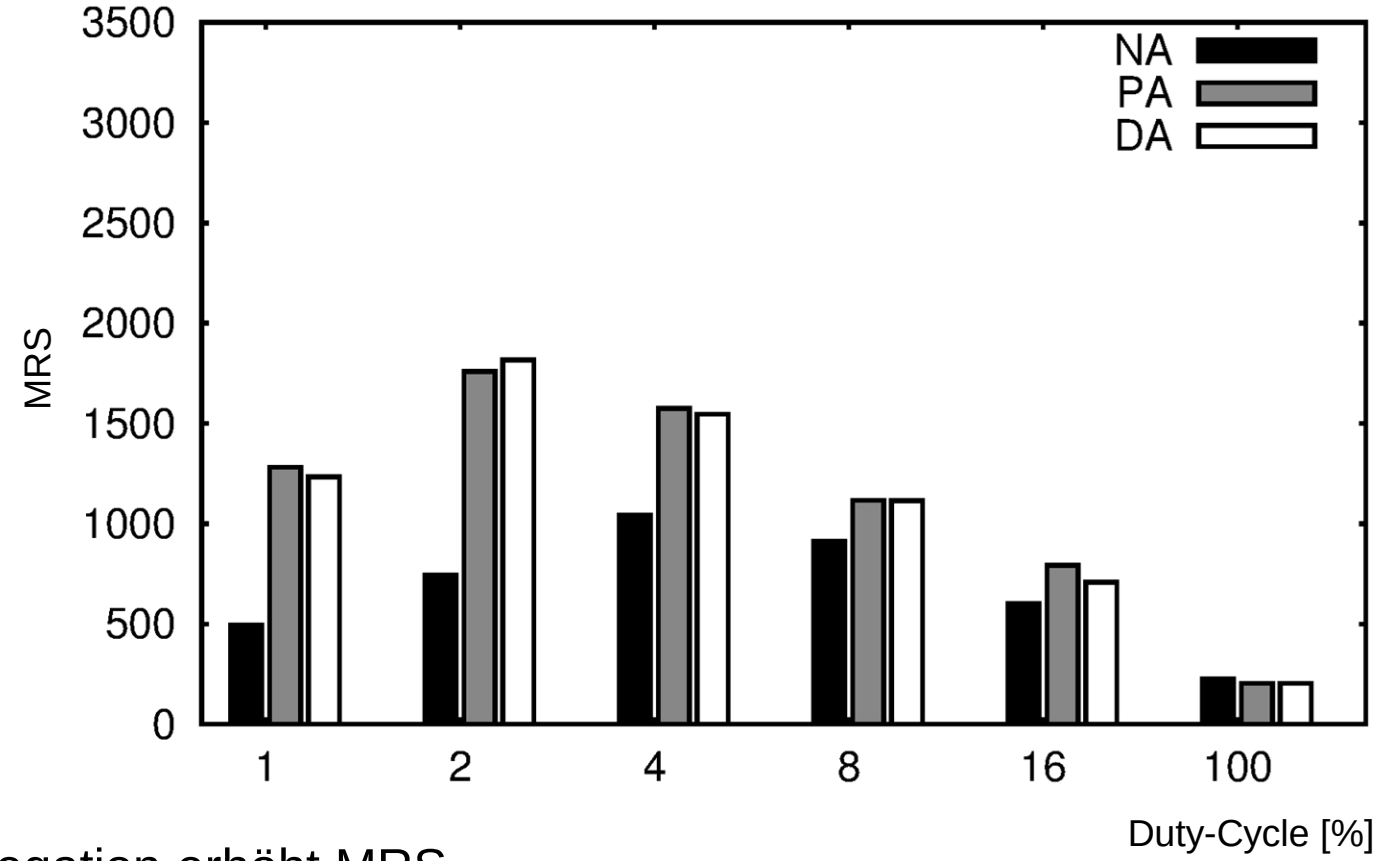


Ergebnisse mit TinyOS LPL



- Unterschiede durch Aggregation minimal
  - Ursache: LPL synchronisiert Knoten lokal
  - Folgeübertragungen günstiger, da Präambel kürzer
  - Kein unmittelbarer Zusammenhang zwischen Energiebedarf und Anzahl Datenpakete oder –volumen!

Ergebnisse mit S-MAC



- Aggregation erhöht MRS
  - Ursache: Durchsatz mit SMAC bei kleinem Duty-Cycle gering
  - Aggregation reduziert Anzahl Übertragungen
  - Pro Übertragung werden mehrere Messwerte transportiert → höheres MRS

Fazit

- Ergebnis stark Abhängig von MAC-Protokoll
  - Parameter des MAC-Protokolls in der Regel wichtiger als Aggregation
  - Aggregation hilft Stausituationen aufzulösen
- Daten- und Paketaggregation kein signifikanter Unterschied
  - Paketaggregation der Datenaggregation vorzuziehen, da alle Quelldaten vorliegen und nicht nur Aggregat

- Schlussfolgerungen
  - Datenvolumen hier keinen großen Einfluss auf Energiebedarf
    - Maximale Paketgröße in Sensornetzen beschränkt aber Möglichkeiten der Paketaggregation
  - Anzahl Übertragungen ist oft wichtiger Faktor

Die von uns zur Erstellung der Folien genutzte

LITERATUR

Literatur

[Karl06] H. Karl, A. Willig; Protocols and Architectures for Wireless Sensor Networks; Wiley, 2. Auflage, 2006, ISBN 978-0470-09510-2

[DBN03] B. Deb, S. Bhatnagar, B. Nath, Information Assurance in Sensor Networks, International Workshop on Wireless Sensor Networks and Applications, San Diego, USA, September 2003

[FJMLZ95] S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang, A Reliable Multicast Framework for Light-weight Sessions and Application Framing", ACM SIGCOMM Conference, Cambridge, USA, Oktober1995

[SAA03] Y. Sankarasubramaniam, O.B. Akan, I.F. Akyldiz, ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks, ACM International Symposium on Mobile Ad Hoc Networking and Computing, Annapolis, USA, Juni 2003

[TAH03] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, Infrastructure Tradeoffs for Sensor Networks, International Workshop on Sensor Networks and Applications, Atlanta, USA, November 2003

[WCK02] C.-Y. Wan, A.T. Campbell, L. Krishnamurthy, PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks, ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, USA, September 2002

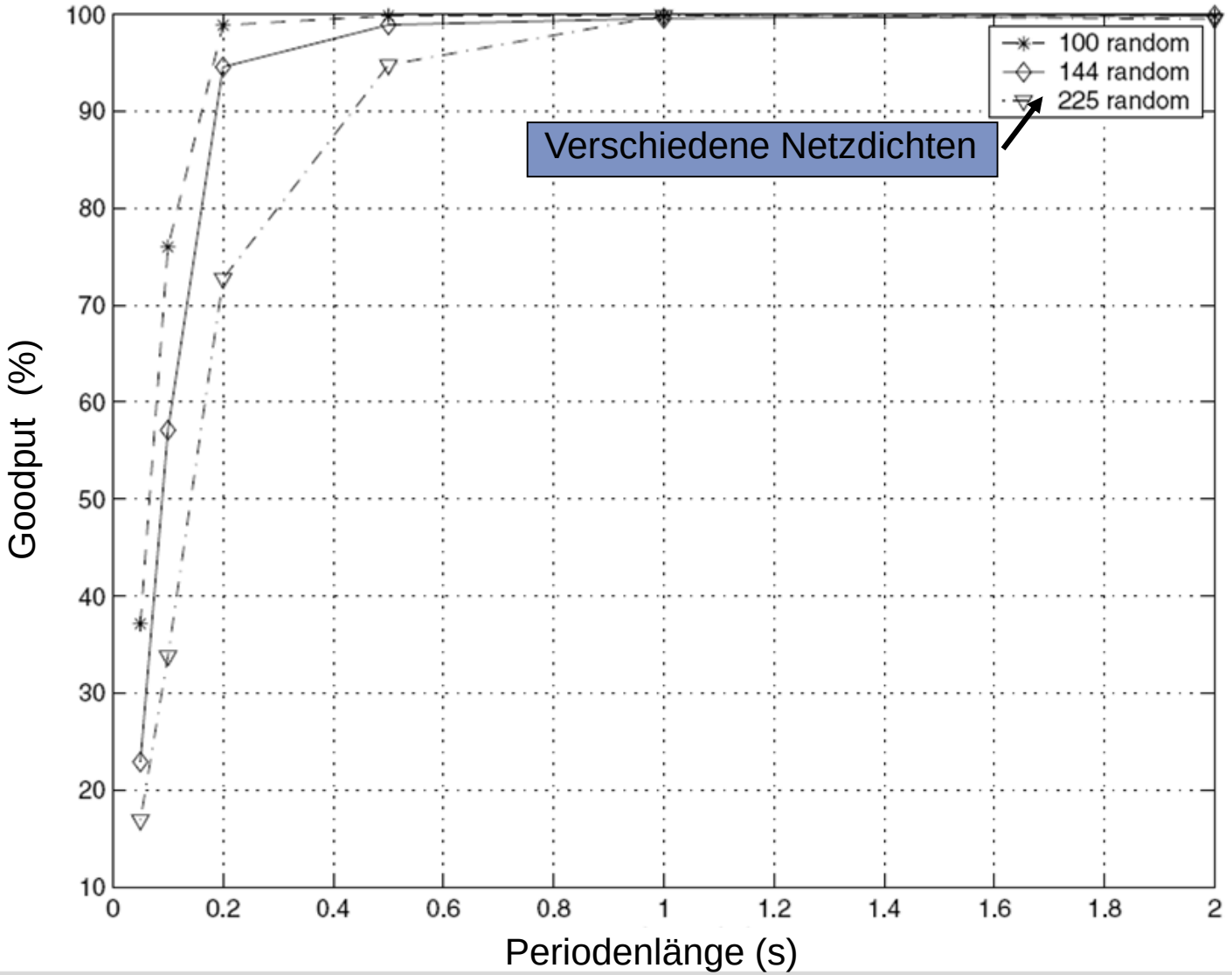
Weitere Beispiele und Folien, die erwähnenswert sind, aber im Hauptteil keinen Platz mehr gefunden haben

ANHANG

Auswirkungen von Stau/Überlast

- Simulation
  - Phänomen bewegt sich durchs Sensornetz (800 m x 800 m)
    - 2 m/s, „Random Waypoint“
  - Verschiedene Netzwerkdichten
    - 100, 144, 255 Systeme zufällig gleichverteilt platziert
  - Reichweiten
    - Sensorik 200 m
    - Funk 250 m
  - 802.11 CSMA/CA, 2 Mbit/s Datenrate
  - Sensoren messen Position eines sich bewegenden physikalischen Phänomens und schicken diese mit fester Periode zur Senke
- Gemessen: Goodput
  - Anzahl der die Senke erreichenden Dateneinheiten durch Anzahl der gesendeten Dateneinheiten (%)

Beispiel (1): Auswirkungen von Stau/Überlast



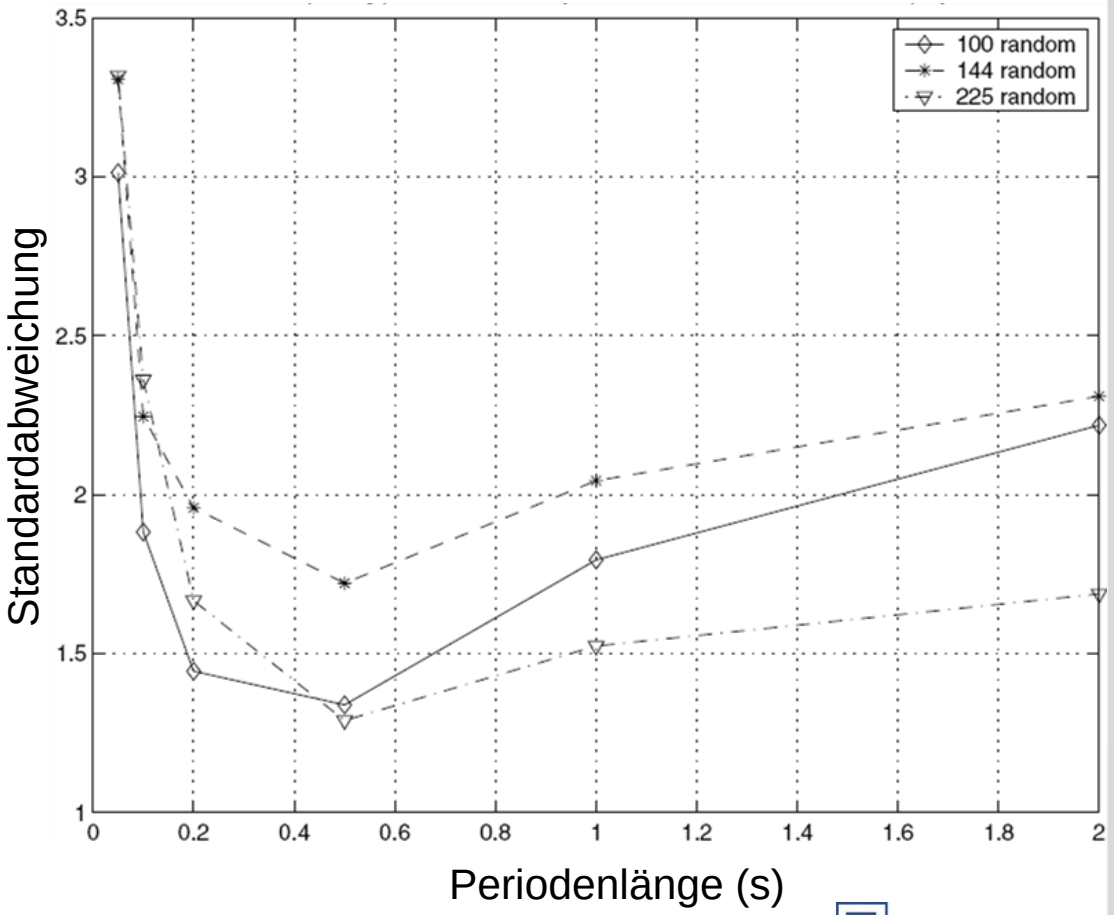
[TAH03]

Beobachtungen

- Je höher die Dichte, desto eher Stau
  - Mehr Kollisionen → Überlast
- Je kleiner die Perioden, desto eher Stau
  - Mehr Dateneinheiten → Mehr Kollisionen → Überlast
- Stau bedeutet
  - Schlechter Durchsatz bzw. schlechter Goodput
  - Energieineffizienz
    - Dateneinheiten müssen ggf. neu übertragen werden

Beispiel (2): Auswirkungen von Stau/Überlast

- Gleiches Szenario
- Gemessen: „Genauigkeit“
  - Standardabweichung zwischen tatsächlichen Positionen und gemeldeter Positionen
- Durch Stau erreichen die Senke „veraltete“ Dateneinheiten
- In Sensornetzen bedeutet Stau damit auch Ungenauigkeit
- Weiteres Problem: Stau erkennen



[TAH03]

Wie kann Stau überhaupt erkannt werden?

- TCP vermutet Stau bei Ausbleiben von ACKs
  - Aber: ACKs in Sensornetzen nicht immer geeignet
  - Nur rein lokales Vermuten von Stau möglich?

- Erkennen von Stau durch MAC Beobachtung? Schwer!
  - Zustand des Mediums nicht immer Indiz für Stau
    - CSMA MACs: Hohe Kanalauslastung → Stau (einfach zu erkennen – durch Abhören des Kanals)
    - TDMA MACs
      - Hohe Kanalauslastung kein Problem für Durchsatz (Zeitslots sind reserviert)
      - Stau entsteht durch volle Warteschlangen, daher schwieriger zu erkennen

Wie kann Stau überhaupt erkannt werden?

- Alternativer Ansatz: Lokal Stau erkennen
  - Systeme überlastet, falls Sendepuffer voll
  - Regel: “Überlastet := Puffer > Schwellenwert”
    - Problem: Puffer voll → zu spät
    - Besser: aktuelle Wachstumsrate des Puffers mitbetrachten, z.B. :
      - Puffer nicht voll aber Wachstum hoch → Stau
      - Puffer fast voll und negatives Wachstum hoch → Stau löst sich auf
  - Problem: nur lokales Erkennen von Stau möglich

- Andere Ansätze?
  - ESRT (→ gleich!)
    - Erkennung durch fehlende/zu wenig Dateneinheiten bei Senke