Opportunistische Weiterleitung von netzwerkcodierten Multicast-Übertragungen in drahtlosen Sensornetzen

zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Dipl.-Inform. Jens Horneber

aus Dachau

Tag der mündlichen Prüfung: 03. Juni 2015

Erste Gutachterin:	Prof. Dr. Martina Zitterbart Karlsruher Institut für Technologie (KIT)
Zweiter Gutachter:	Prof. Dr. Uwe Hanebeck Karlsruher Institut für Technologie (KIT)

Danksagung

Diese Doktorarbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Telematik des Karlsruher Instituts für Technologie. Frau Prof. Dr. Martina Zitterbart ermöglichte mir durch die Anstellung an ihrem Lehrstuhl und ihre Übernahme des Referats, dass diese Arbeit in den Jahren 2009 bis 2015 entstehen konnte. Ich danke ihr für die bereitwillige Betreuung und die interessante und bereichernde Zeit, die ich am Lehrstuhl verbringen durfte.

Darüber hinaus danke ich Prof. Dr. Uwe Hanebeck, der trotz seiner zahlreichen Verpflichtungen am Karlsruher Institut für Technologie, freundlicher Weise und mit großem Interesse die Zeit gefunden hat, das Koreferat meiner Promotion zu übernehmen.

Auch den Mitgliedern des Graduiertenkollegs GRK1194 „Selbstorganisierende Sensor-Aktor-Netze“ gilt mein besonderer Dank für die zahlreichen und anregenden Diskussionen zu unterschiedlichen Betrachtungs- und Herangehensweisen in der Welt der Sensornetze.

Den Mitarbeitern des Instituts für Telematik, die mich in meiner Zeit am Institut begleitet haben, danke ich für die unzähligen fachlichen Diskussionen, kritischen Anmerkungen und hilfreichen Anregungen.

Schließlich gilt mein Dank auch meiner Familie und meinen Freunden, die nie an der Entstehung dieser Arbeit gezweifelt haben, für zahlreiche Stunden des Korrekturlesens, für die Unterstützung und Aufmunterung in anstrengenden Phasen und für ihr Verständnis wenn gelegentlich mal nicht genug Zeit miteinander blieb.

Karlsruhe, im Juni 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

P_i	Paket
M_i	Nachricht
$\|P\|$	Anzahl der in einem Paket enthaltenen Nachrichten $M_1, ..., M_k$
$M_i \in P$	Das Paket P besteht aus einer Kombination aller Elemente aus der Nachrichtenmenge $\mathcal{M}$ mit $M_i \in \mathcal{M}$
C_i	Codierungsvektor zur Identifikation der Nachricht M_i
$MsgID_i$	Hashwert zur Identifikation der Nachricht $M_i \in P$
$PacketID$	Identifikator eines Paketes
$\mathcal{D}(P) \rightarrow \{M\}$	Decodingfunktion
$\mathcal{R}(P) \rightarrow \{M\} \cup \{P\}$	Resolutionsfunktion
$\mathcal{E}(\{M\}) \rightarrow P$	Codierungsfunktion
$\mathcal{H}$	Multiplikative Hashfunktion
N_i	Nachbarknoten
m	Anzahl der Nachrichten, die maximal gemeinsam in ein Paket P codiert werden
h	Parameter zur initialen Weiterleitung neuer Nachrichten mittels Fast-Forwarding-Verfahrens
c	Codierungskomplexität entsprechend der Freiheitsgrade nach dem verwendeten Codierungsschema
t_{enc}	Zeitgeber für das periodische Codieren neuer Pakete aus gepufferten Nachrichten
t_b	Zufälliger Backoff der Fast-Forwarding-Komponente
t_{ret}	Zeitgeber für Sendewiederholungen (*Retransmission Timer*)
thr_{drop}	Schwellenwert zum Empfang des gleichen Paketes bevor die Weiterleitung eingestellt wird
thr_{enc}	Schwellenwert zum Codieren neuer Pakete aus dem Encoding Buffer
r	Anzahl der Simulationsdurchläufe
$\mathcal{B}_{dec}$	Decoding Buffer

$\mathcal{B}_{enc}$	Encoding Buffer
$\mathcal{B}_{atom}$	Atomic Buffer
$\mathcal{B}_{slq}$	Send-Listen-Queue
$\|\mathcal{B}\|$	Größe eines Pufferspeichers
$R(P)$	Empfangshäufigkeit des Paketes P
Q	Quellknoten
S	Senkenknoten
Z	Zwischenknoten
$GF(n)$	n-elementiges Galois-Feld
$\mathcal{A}$	Menge der Ausgabeblöcke
$\mathcal{Q}$	Menge der Quellblöcke

1. Einleitung

Die Vision von drahtlos vernetzten Kleinstcomputern, die in unserer Alltagsumgebung eingebettet sind und uns in unauffälliger Form allgegenwärtig begleiten, hat bereits Mark Weiser 1991 in seinem populären Aufsatz „The Computer for the 21st Century“ [115] thematisiert. Auf dieser Vision gründen heute weltweit zahlreiche Forschungsanstrengungen, die insbesondere auch die Geräteklasse der drahtlosen Sensornetze hervor gebracht haben. Die Architektur eines Sensornetzes zeichnet sich durch die stark anwendungsbezogene Auswahl von optimierter Hard- und Software aus. Dabei sind Leistungsfähigkeit und Ressourcen der einzelnen Netzwerkknoten stark beschränkt. Dies gilt vor allem für die verfügbare Energie, Rechenleistung, den Speicher und die Nutzung des drahtlosen Mediums.

Zukünftig werden sich Sensornetze von sehr spezialisierten und optimierten Netzen hin zu Netzen entwickeln, die unterschiedlichste Aufgaben, teilweise zeitgleich und teilweise sequentiell, wahrnehmen können. Sensornetze sind damit nicht länger an einen einzelnen Nutzer oder eine einzelne Anwendung gebunden, sondern können sowohl durch mehrere Nutzer, als auch durch mehrere Anwendungen beansprucht werden, die sich die beschränkten Ressourcen des Sensornetzes teilen.

Dazu reicht es nicht länger aus, das Kommunikationsverhalten einer einzelnen Anwendung zu optimieren. Vielmehr muss nun das Kommunikationsverhalten aus dem Datenverkehr der vorliegenden Anwendungskombination optimiert werden.

Beispielsweise könnte ein drahtloses Sensornetz in einer Großstadt ausgebracht sein, um sowohl der Überwachung von Verkehrsströmen, als auch der Beobachtung von Umweltdaten zu dienen. Das gleiche physische Sensornetz könnte darüber hinaus beim Auffinden freier Parkplätze, der Detektion von Unfällen und Verbrechen oder bei der Überprüfung von Infrastruktur- und Gebäudeintegrität hilfreich sein.

Das Ausbringen einer Vielzahl von spezialisierten Sensornetzen, die in dieser urbanen Umgebung für zusätzliche Störeinflüsse untereinander sorgen könnten, wäre ein unnötiger Kosten- und Wartungsfaktor. Schon bei einfachen Experimenten mit überschaubarer Knotenzahl hat sich insbesondere das Ausbringen und die geeignete Positionierung der Knoten als äußerst komplexer und zeitaufwändiger Prozess erwiesen, der gegen die Nutzung mehrerer physischer Sensornetze für unterschiedliche Aufgaben mit jeweils kurzzeitiger Nutzungsdauer spricht.

Die Nutzung eines Sensornetzes für unterschiedliche Anwendungszwecke führt mit Blick auf die resultierende Netzwerktopologie zu einer Überlagerung der unterschiedlichen anwendungsspezifischen Kommunikationsmuster. Der Datenverkehr von verschieden Quellen trifft auf weiterleitenden Sensorknoten innerhalb des Sensornetzes zusammen. Ein typisches Merkmal dieser Kommunikationsform ist die Entstehung von konkurrierendem Multicast-Datenverkehr ausgehend von den jeweiligen Datenquellen.

Daten die in einer bestimmten Anwendungsumgebung oder einem bestimmten Nutzungskontext anfallen, müssen bei einer verteilten Anwendung auch über Knoten hinweg erhalten bleiben. Es muss sichergestellt werden, dass jeder Nutzer und jede Anwendung die relevanten Daten aus dem entsprechenden Kontext erhält. Wird die Kommunikation mehrerer Anwendungen im gleichen Netzwerk kombiniert, so führt dies auch zu einer Überlagerung der unterschiedlichen Kontexte der Anwendungen und Nutzer. Potentiell damit verbundene Fehler- und Konfliktsituationen um Netzwerkkapazitäten und -ressourcen müssen durch frühzeitige Koordination bereits auf den Sensorknoten oder im Netzwerk aufgelöst werden. Zudem unterscheiden sich die Anwendungsanforderungen der im Netzwerk *koexistierenden Anwendungen* bezüglich der Netzwerkstruktur und Ansprüche z. B. an Durchsatz, Latenz, Priorisierung und Zuverlässigkeit.

Neben dem Einsatz von Virtualisierungstechniken zur Abstraktion von Knotenressourcen ist daher verstärkt eine Koordination des Netzes durch ein geeignetes Kommunikationsschema für die Realisierung zukünftiger Sensornetze notwendig. Dabei müssen die Besonderheiten von drahtlosen Sensornetzen berücksichtigt werden, etwa dass Nutzereingriffe vor Ort oft nicht möglich oder nicht vorgesehen sind und stattdessen Methoden der Selbstorganisation favorisiert werden.

Im Gegensatz zur Internetarchitektur sind in ressourcenarmen Sensornetzen Netzwerkschichten übergreifende Optimierungen der Hard- und Software zur Verbesserung der Energieeffizienz nicht ungewöhnlich. Der parallele Betrieb mehrerer Protokolle, die passend zur Anwendung ausgewählt werden, ist im Allgemeinen nicht vorgesehen. Dennoch muss auch bei gemeinsamer Nutzung des Sensornetzes jeweils ein geeigneter Kompromiss im Spannungsfeld von Ressourcennutzung, Leistungsfähigkeit der Kommunikation, Zuverlässigkeit und Energiebedarf gefunden werden.

Abbildung 1.1 zeigt beispielhaft die Ausgangssituation mit mehreren überlagerten Anwendungen, die im gleichen Sensornetzwerk koexistieren sollen, aber unter-

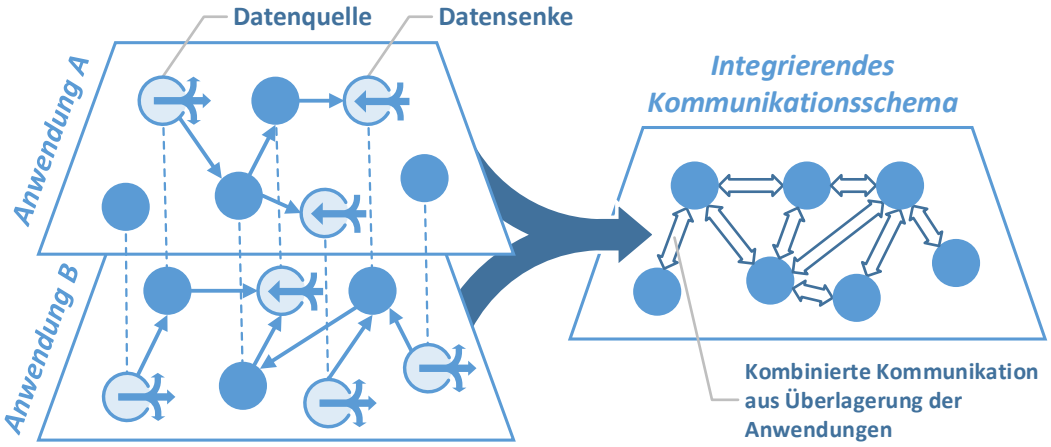

Abbildung 1.1 Zusammenfassung der Anwendungskommunikation in einem einheitlichen Kommunikationsschema

schiedliche Anforderungen hinsichtlich ihrer Datensenke, Datenquellen, Kommunikationsrichtung und benötigten Sensorknoten aufweisen.

Die Kombination der Anwendungen in einem geeigneten Kommunikationsschema muss daher die Überlagerung und Integration unterschiedlichster Kommunikationsmuster vorsehen. Eine weitere Konsequenz ist die weitgehende Abstraktion von anwendungsoptimierten Zustellungs- und Routingmechanismen.

Vielversprechende Kommunikationskonzepte, die in Forschungspublikationen der letzten Jahre veröffentlicht wurden, sind insbesondere das gemeinsame Codieren von Nachrichten aus unterschiedlichen Datenquellen innerhalb des Netzwerks mittles sogenannter *Netzwerkcodierung* und das *opportunistische Weiterleiten* von Daten über unterschiedliche Pfade zwischen Datenquelle und Datensenke entsprechend der aktuellen Netzwerksituation.

Eine Kombination der beiden Konzepte ist vielversprechend, da zusätzliche Redundanz, die durch die unterschiedliche Pfade entsteht, durch die Netzwerkcodierung genutzt werden kann, um die vorhandene Netzwerkkapazität auszulasten.

1.1 Problemstellung und Zielsetzung

Bisherige Forschungsarbeiten konzentrierten sich überwiegend auf die Konzeption und Optimierung der Kommunikationseigenschaften einzelner Protokolle für ausgewählte Anwendungsklassen.

Zentrale Forschungsfrage der Arbeit ist daher, wie ein geeignetes, anwendungsübergreifendes Konzept für drahtlose Sensornetze aussehen kann, das die Multicast-Kommunikation unterschiedlicher Anwendungen in einem einheitlichen Kommunikationsschema integriert.

Eine Herausforderung bei der Entwicklung eines derartigen Kommunikationsschemas besteht insbesondere darin, geeignete Kommunikationskonzepte als Grundlage zu finden, die den gewünschten Grad an Flexibilität, Selbstorganisation und

dezentraler Informationsbereitstellung realisieren können. Schwerpunkt bilden Kommunikationskonzepte die typischerweise für Aufgaben der Netzwerk- und Transportschicht eingesetzt werden, insbesondere die Untersuchung von Konzepten zur Netzwerkcodierung und der opportunistischen Mehrwege-Weiterleitung in Multicast-Kommunikationsszenarien.

Alltagsgegenstände und mobile Geräte, wie Smartphones und Tablets, werden im urbanen Umfeld und dem zukünftigen „Internet der Dinge“, zukünftig vorherrschend sein, so dass auch die betrachteten Kommunikationskonzepte mit unterschiedlich leistungsfähigen Geräteklassen realisierbar sein müssen.

Ein weiterer wichtiger Aspekt ist hier die Adaption der Kommunikation an die jeweils vorhandenen Hardware und Systemarchitektur drahtloser Sensornetze, sowie die vorhandenen Netzwerkressourcen. Neben den zusätzlichen Problemstellungen durch die eingeschränkte Rechenleistung und Speicherkapazität von Sensorknoten, muss auch die dynamische Topologie des Netzwerks berücksichtigt werden.

Es stehen somit nicht anwendungsspezifische Lösungen im Vordergrund, sondern vielmehr Lösungen, die es letztlich erlauben, eine Vielzahl von möglichen Anwendungen realisieren zu können. Dabei steht in dieser Arbeit ein flexibler, effizienter und zugleich robuster Entwurf des Kommunikationsschemas im Vordergrund, der auf einer dezentralen und selbstorganisierenden Koordination mittels des lokalen Knotenzustands basiert.

Auch das klassische Verständnis des Informationsflusses in drahtlosen Sensornetzen, bei dem zahlreiche Datenquellen entlang einer Baumtopologie ihre Daten in Richtung einer anwendungsspezifischen Datensenke übermitteln, ist in diesem Szenario hinfällig.

So hat die Detektion und Signalisierung von Fehlern etwa stark ereignisbasierenden Charakter, während die Kommunikationsmuster und Datenströme zur periodischen Überwachung der Knotenintegrität eher dem Einsammeln von Daten in bisherigen Sensornetzen ähneln.

Sensordaten können zudem von unterschiedlichen Senken für individuelle Anwendungen und Nutzungsinteressen benötigt werden. Da die Art, Anzahl, Position und Existenz von Senken mit den ausgebrachten Anwendungen wechselt und starken zeitlichen Veränderungen unterliegt, ist eine starke Abhängigkeit der genutzten Kommunikationskonzepte von einzelnen oder statischen Senken zu vermeiden. Mit der Position von Senken wechseln auch die Richtungen von Nachrichtenströmen. Geeignete Kommunikationskonzepte müssen daher eine dynamische Netzwerktopologie unterstützen und dürfen keine eigenen Anforderungen an die Senke stellen.

Sensornetze sind durch Umwelteinflüsse, drahtlose Kommunikation und Anwendungsvielfalt, Fehlersituationen in besonderem Maße ausgesetzt. Daher ist die Unterstützung von Maßnahmen zur Fehlerbeherrschung und zur Anpassung der genutzten Redundanz, zum Beispiel durch Kommunikation über mehrere Pfade, erforderlich.

Energieeffizienz ist in drahtlosen Sensornetzen das entscheidende Kriterium für die Lebenszeit des Netzes und die Verfügbarkeitsdauer der Anwendungen. Typischerweise hat die drahtlose Kommunikation eines Sensorknotens einen großen Anteil am Energiebedarf des Systems, so dass Kommunikationsprotokolle hinsichtlich Effizienz entworfen werden. In diesem Zusammenhang sind auch die Eigenschaften und Auswirkungen der Protokolle von tieferen Netzwerkschichten von Bedeutung, insbesondere das verwendete Medienzugriffsverfahren.

Somit unterscheidet sich die hier gewählte Vorgehensweise von dem Entwurf einer optimierten Speziallösung für eine ausgewählte Anwendungskombination. Der erwartete Zusatznutzen rechtfertigt daher die Adaption und Integration eines koordinierenden Kommunikationsschemas erst unter Bedingungen, in der sich die Kombination der Anwendungen dynamisch verändern kann.

Existierende Ansätze zur Nutzung von Netzwerkcodierung bei der effizienten Verteilung von Daten konzentrieren sich auf Kommunikationsszenarien mit sehr wenigen Knoten, fester Topologie und größeren zusammenhängenden Datenmengen, die als Datenstrom betrachtet werden können.

Bisher wenig durch Forschungsansätze untersucht, ist dagegen die Situation mit mehreren Anwendungen im gleichen Sensornetz. Außerdem wurde die Übermittlung einzelner Pakete an Stelle von Nachrichtenströmen in dieser Situation kaum untersucht. Die bisherigen Ansätze treffen darüber hinaus weitreichende Annahmen zur vorliegenden Topologie oder Knotenanzahl, ohne mögliche Dynamiken zu berücksichtigen.

1.2 Dissertationsbeiträge

Im Rahmen dieser Dissertation sind die folgenden wissenschaftlichen Beiträge mit Bezug zum Dissertationsthema entstanden.

OCSSIM-Kommunikationsschema

Das entworfene Kommunikationskonzept vereint heuristische Netzwerkcodierung mit opportunistischer Mehrwege-Weiterleitung zur effizienten Kommunikation. Dazu wurde ermittelt, in wie fern sich die betrachteten Mechanismen auf stark speicherbeschränkten Systemen kombinieren lassen, um letztendlich ein möglichst umfassendes Anwendungsspektrum zu bedienen.

Priorisierung, schnelle Weiterleitung und Neukombination von Nachrichten

Einen weiteren Beitrag stellt das gezielte Aufspalten des eintreffenden Datenverkehrs in zwei sich ergänzenden Prozessen auf jedem an der Kommunikation beteiligten Knoten dar. Dies dient der Unterscheidung des Datenverkehrs anhand unterschiedlicher Anforderungen hinsichtlich eines priorisierten oder effizienteren und zuverlässigeren redundanten Kommunikationsverhaltens, bei dem Nachrichten unterschiedlicher Quellen kombiniert werden.

Lokale Weiterleitung ohne Zusatzaufwand für anwendungsspezifischen Topologieaufbau

Durch ein geeignetes Schema zur Netzwerkcodierung, das eine Multicast-Kommunikation bei überschneidenden Nachrichtenströmen aus unterschiedlichen Quellen realisiert, konnte dabei selbst bei unbekannten oder hoher Dynamik unterliegenden Netzwerktopologien auf zusätzliche Routing-Mechanismen verzichtet werden. Darüber hinaus braucht kein eigenständiges Verfahren zur Verteilung von Codierungsinformationen genutzt werden, um ein knoten-lokales Decodieren im Netzwerk zu ermöglichen. Diese Eigenschaften konnten genutzt werden, um ein opportunistisches Weiterleitungsverfahren zu entwerfen, das ebenfalls auf knoten-lokalen Entscheidungen basiert.

Effiziente senkenunabhänige Kommunikation in Mehrwege-Topologien

Um zu ermöglichen, dass Kommunikation zwischen Sensorknoten auch dann stattfinden kann, wenn keine Senke anwesend ist, muss gewährleistet werden, dass anfallende Daten effizient verteilt und zwischengespeichert werden können. Dazu werden in dieser Arbeit Konzepte zur opportunistischen Kommunikation erarbeitet, die sich durch Verfahren zur Weiterleitung von Nachrichten über mehrere Pfade auch für Teilausfälle des Netzes eignen. Eine später erscheinende Senke kann zur Rekonstruktion der Daten eingesetzt werden. Für die Kommunikation muss dazu auch das Problem der implizit bestätigten Kommunikation zwischen benachbarten Knoten betrachtet werden, die sich lediglich durch Mithören benachbarter Übertragungen koordinieren.

Robustes Kommunikationsrahmenwerk für verteilte Sensornetzanwendungen (*ROSMARIN*) auf Basis des OCSSIM-Kommunikationsschemas

Zur Analyse des OCSSIM-Kommunikationsschemas und der eingesetzten Kommunikationskonzepte werden in dieser Arbeit die nötigen Werkzeuge und ein Rahmenwerk entwickelt. Insbesondere wird untersucht, in wie fern sich das Kommunikationsverhalten in typischen Fehlersituationen wie Paketverlusten und Knotenausfällen verändert und welchen Einfluss die ressourcenbeschränkte Hardware auf das Kommunikationsverhalten hat. Das Rahmenwerk bildet die Grundlage zur Evaluation, um das Zusammenspiel prototypischer Realisierungen der untersuchten Konzepte in drahtlosen Sensornetzen zu betrachten. Dazu kombiniert es die Funktionalität der realisierten Konzepte, um Anwendungen die benötigten Eigenschaften der Netzwerk- und Transportschicht anzubieten. Das Rahmenwerk bietet zudem die erforderlichen Schnittstellen für Fehlersignalisierung, Monitoring und Datenverteilung im Sensornetz.

1.3 Gliederung

Hier wird nun ein kurzer Überblick über die weitere Gliederung der Arbeit und den Inhalt der einzelnen Kapitel gegeben.

In Kapitel 2 werden die Grundlagen und die Anforderungen diskutiert, die aus dem Nutzungsszenario koexistierender Anwendungen in drahtlosen Sensornetzen entstehen. Ein Schwerpunkt bildet die Betrachtung der zu Grunde liegenden Kommunikationsmuster und die Konsequenzen für ein geeignetes Kommunikationsschema, das diese Muster integriert.

Kapitel 3 behandelt die elementaren Konzepte, Anwendungsbereiche und Kernproblematiken der Netzwerkkodierung in Hinblick auf die Eignung für drahtlose Sensornetze. Zusätzlich werden bereits existierende Ansätze analysiert und geeignete Bewertungskriterien ausgewählt.

Der Kern von Kapitel 4 bildet die Betrachtung von Konzepten zur Datenverteilung und Weiterleitung in drahtlosen Sensornetzen. Ein Schwerpunkt bilden dabei opportunistische Verfahren, die sich an die Gegebenheiten des aktuellen Netzwerkzustandes anpassen können und Verfahren, die eine Weiterleitung über unterschiedliche Pfade zulassen.

Die Zusammenführung der Konzepte zur heuristischen Netzwerkcodierung und opportunistischen Weiterleitung in das entworfene Kommunikationsschema *OCSSIM* erfolgt in Kapitel 5. Dabei werden die getroffenen Entwurfsentscheidungen und ihre Auswirkungen auf das Kommunikationsverhalten diskutiert.

Die Entwicklung des Rahmenwerkes *ROSMARIN* zur Realisation von prototypischen Kommunikationsszenarien auf Grundlage des OCSSIM-Kommunikationsschemas, sowie deren Evaluation ist Inhalt von Kapitel 6. Ergänzt wird dies durch die Vorstellung der Gesamtarchitektur des Rahmenwerkes und die Erläuterung des Zusammenspiels der benötigten Komponenten.

Abschließend werden die Ergebnisse dieser Arbeit in Kapitel 7 zusammengefasst. Die Arbeit endet mit einem kurzen Ausblick auf mögliche weitere Entwicklungen des Forschungsgebietes.

2. Grundlagen

In diesem Kapitel werden zunächst die Besonderheiten der Geräteklasse drahtloser Sensornetze diskutiert und die Herausforderungen für die Unterstützung koexistierender Anwendungen in einem Sensornetz hergeleitet. Im Anschluss werden die typischen Kommunikationsmuster und Netzwerktopologien von Anwendungen in drahtlosen Sensornetzen betrachtet und die resultierenden Veränderungen durch die Einführung verschiedener Anwendungen im gleichen Netzwerk diskutiert. Außerdem wird die Rolle der Anwendungssenken und -quellen im Detail betrachtet und die Anforderungen für neue Kommunikationsmuster identifiziert, die sich aus dem Verzicht auf statisch geprägte Senkenmodelle aus klassischen Kommunikationsmustern ergeben. Diese Anforderungen bilden die Betrachtungsgrundlage für alternative Kommunikationskonzepte der nachfolgenden Kapitel und dem Entwurf des OCSSIM-Kommunikationsschemas in Kapitel 5.

2.1 Drahtlose Sensornetze

Heutige drahtlose Sensornetze zeichnen sich durch eine hohe Anzahl drahtlos kommunizierender Kleinstgeräte, der sogenannten *Sensorknoten* aus. Anders als klassische Rechnernetze sind Sensornetze typischerweise für eine Einzelanwendung entworfen und optimiert. Auf den einzelnen Sensorknoten ist dementsprechend eine anwendungsoptimierte Sensorik vorhanden, die der Messung von Daten dient, um kontextbezogen den Umgebungs- und Anwendungszustand eines Sensorknotens zu erfassen.

Die gemessenen Sensordaten sind somit nicht nur von der Art des Sensors, sondern insbesondere auch vom Ausbringungsort des Sensorknotens abhängig und in einer dynamischen Umgebung auch vom Zeitpunkt der Messung. Die drahtlose Kommunikation der Sensorknoten erlaubt den Austausch der gemessenen Daten über kurze Entfernungen, unabhängig von existierender Kommunikations-

infrastruktur am Ausbringungsort. Damit eröffnen drahtlose Sensornetzwerke die Möglichkeiten, neue Anwendungstypen zu entwerfen, die auf räumlich-zeitlich verteilte Sensordaten angewiesen sind, wie etwa zur Analyse und Beobachtung von (Ober-)Flächen. Weitere Anwendungsmöglichkeiten drahtloser Sensornetze finden sich auch in der Überwachung von mobilen Objekten durch Einbettung der Sensorknoten in dynamische Umgebungen.

Als Energiequelle kommen bislang häufig Batterien oder Solarzellen zum Einsatz, um einen weitgehend autonomen Betrieb zu ermöglichen. Bereits beim Entwurf eines Sensornetzes muss die Energiequelle berücksichtigt werden, da z. B. bei Solarzellen die Nutzungsphasen eingeschränkt sind und keine kontinuierliche Energieversorgung sichergestellt werden kann.

Batterien dagegen beschränken die wartungsfreie Lebenszeit des Netzes. Um den Energiebedarf der Sensorknoten zu reduzieren, ist die Kommunikationsreichweite der Knoten zudem auf wenige Meter beschränkt. Werden Sensornetze im urbanen Umfeld ausgebracht, so ist oft auch eine heterogene Energieversorgungssituation realisierbar, bei der einzelne Knoten über eine kontinuierliche Energieversorgung verfügen und sich damit für besondere Aufgaben mit höheren Leistungsanforderungen im Netzwerk auszeichnen.

Aufgrund der restriktiven Energieversorgung und einer stark miniaturisierten Bauweise wird in Sensorknoten bevorzugt Hardware kostengünstiger und einfacher Bauweise, sowie eingeschränkter Leistungsfähigkeit verwendet. Im Allgemeinen findet ein Mikrocontroller, Speicher, ein Modul zur drahtlosen Kommunikation und bestimmungsgemäße Sensorik, optional auch zusätzliche Aktorik, Verwendung.

Trotz der eingeschränkten Leistungsfähigkeit der einzelnen Sensorknoten und der nicht zwangsläufig homogenen Ausstattung, ist ein typisches Charakteristikum von Sensornetzen die gemeinsame Bearbeitung einer gestellten Aufgabe durch Kooperation einer Vielzahl von Knoten im gleichen Netzwerk. Dennoch sollen die einzelnen Knoten möglichst selbstorganisierend im Netzwerk agieren, um wenig zusätzlichen Aufwand für Koordinations- und Wartungsaufgaben zu verursachen.

Wie auch in leistungsstärkeren drahtlosen Netzwerken prägen die geringere Zuverlässigkeit der Nachrichtenübertragung gegenüber kabelgebundenen Netzen und die komplexere Koordination des Medienzugriffs die Kommunikation in Sensornetzen. Die teils rauen, ungeschützten Umgebungen, in denen Sensornetze eingesetzt werden und die einfache Hardware spitzen diese Probleme oft noch zu, so dass erhöhte Fehlertoleranz gegenüber Knotenausfällen, Paketverlusten und Bitfehlern notwendig sind.

2.2 Unterstützung koexistierender Anwendungen

Der Realisierung einer breiten Anwendungsvielfalt, die im gleichen Sensornetzwerk unterstützt werden kann, sind durch die Ressourcenbeschränkungen, der einzelnen Sensorknoten Grenzen gesetzt. Dazu zählt offensichtlich die Beschränkung der allgemeinen Lebensdauer durch die verfügbare Energie. Durch den Betrieb

mehrerer Anwendungen wird eine höhere Leistung des Knotens erzielt oder es ist eine längere Leistungsbereitschaft des Knoten erforderlich. Beides spiegelt sich in einem höheren Energiebedarf wider.

2.2.1 Beschränkung des verfügbaren Speichers

Besondere Schwierigkeiten für die Nutzung verschiedener Anwendungen stellt die Beschränkung des vorhandenen Speichers dar. Dabei ist für Geräte in Sensornetzen üblicherweise zwischen drei beschränkten Speicherarten zu unterscheiden.

Beschränkung des Speichers für Sensormessungen: Dieser Speicher wird in drahtlosen Sensornetzen üblicherweise als nicht-flüchtiger, aber möglichst energiesparender Speicher, beispielsweise durch Flash-EEPROM, realisiert. Er dient zur lokalen Speicherung der gemessenen Sensordaten, bis diese zum Zweck der Datensammlung versendet werden oder im Netzwerk weiterverarbeitet werden können. Ein Speicherkonflikt ist hier vor allem dann zu erwarten, wenn mehrere sensordatenintensive Anwendungen in häufigem Wechsel oder schnell alternierend ausgeführt werden. Weniger problematisch bei diesem Speichertyp ist ein sequentieller Betrieb mehrerer längerfristiger Anwendungen, wenn lokal zwischengespeicherte Messungen vor dem Anwendungswechsel noch sicher aus dem Speicher entfernt werden können. Die Speicherbeschränkung lässt sich in heterogenen Sensornetzen auch umgehen, indem die Messdaten nicht lokal zwischengespeichert werden, sondern auf leistungsfähigerer Sensorknoten mit verfügbarer Speicherkapazität verlagert werden.

Beschränkung des Programmspeichers: Der Programmspeicher ist ebenfalls oft als nicht-flüchtiger Flash-EEPROM realisiert. In Sensornetzen werden bevorzugt RISC-Mikrocontroller mit einer Havard-Architektur verwendet. Die Größe des Binärcodes der Anwendungen hat in dieser Architektur keine Auswirkungen auf den verfügbaren Arbeitsspeicher.

Für den Betrieb eines Sensornetzes mit mehreren Anwendungen ergibt sich folglich jedoch ein Speicherkonflikt um den Programmspeicher, da typischerweise die gewünschten Anwendungen bereits zum Ausbringungszeitpunkt der Sensorknoten im Programmspeicher vorkonfiguriert werden. Zur Unterstützung mehrerer Anwendungen werden daher Sensorknoten mit ausreichend großen Kapazitäten der Programmspeicher benötigt. Die Verwendung von dedizierten Protokollen zur nachträgliche Programmierung über die Netzwerkverbindung, sogenanntes *Over-the-Air-Programming* (OTAP) [113], lassen es zu, Anwendungen zur Laufzeit des Sensornetzes austauschen. Eine tatsächliche Koexistenz der Anwendungen kann mit diesem Ansatz jedoch nicht ermöglicht werden.

Beschränkung des Arbeitsspeichers: Der Arbeitsspeicher der Sensorknoten ist bevorzugt ein flüchtiger Speicher mit wahlfreiem Zugriff zur Manipulation von laufzeitabhängigen Daten. Es wird aus Gründen der Energieeffizienz

eine Realisierung als sehr kleiner statischer SRAM bevorzugt, da hier auf ein regelmäßiges Auffrischen der Speicherzellen verzichtet werden kann. Typische Größen des Arbeitsspeichers dedizierter Sensorknoten liegen im Bereich weniger Kilobytes bis einzelner Megabytes. Die Beschränkung des Arbeitsspeichers ist für den Betrieb mit mehreren Anwendungen besonders problematisch, da die gleichzeitige Kontext- und Zustandshaltung zur Ausführung mehrerer Anwendungen praktisch ausgeschlossen wird. Erschwerend kommt hinzu, dass leistungsschwache Sensorknoten im Allgemeinen keine Speicherverwaltungseinheiten (MMU) verwenden, die eine Virtualisierung des Speichers ermöglichen. Folglich ist keine Isolation der Adressbereiche unterschiedlicher Anwendungen im Hauptspeicher vorgesehen und das Sichern des Anwendungszustands bei Kontextwechseln zwischen Anwendungen wird erheblich erschwert.

Existierende Betriebssysteme für Sensornetze, wie TinyOS [64] oder Contiki [23] erlauben aufgrund der Beschränkungen des Speichers und der Hardwarearchitektur keine echte Nebenläufigkeit zwischen verschiedenen Anwendungen. Durch zusätzliche Maßnahmen der Betriebssysteme ist jedoch eine rudimentäre Form der Nebenläufigkeit durch preemptive Threads auf Anwendungsschicht möglich. Sie lässt einen asynchronen und konkurrierenden Zugriff auf Hardwareressourcen wie z. B. Sensoren und Zeitgeber zu. Verschiedene Anwendungen können daher auf aktuellen Systemen vorwiegend sequentiell ausgeführt werden.

2.2.2 Kooperation mit leistungsfähigen drahtlosen Geräten in heterogenen Netzwerken

Heterogene drahtlose Netzwerke mit Geräten unterschiedlicher Leistungsklassen stellen für drahtlose Sensornetze, die für mehrere Anwendungen konzipiert sind, ein vielversprechendes Einsatzgebiet dar.

Beim Ausbringen von drahtlosen Sensornetzen in urbanen Umgebungen ist mit einer Integration des Sensornetzes in existierende Netzwerke zu rechnen. Der parallele Betrieb mehrerer anwendungsspezifischer Netze erscheint aufgrund zu erwartender Mehrkosten, zusätzlichem Betriebs- und Wartungsaufwand, sowie zusätzlichem Energiebedarf, dagegen nicht vorteilhaft.

Insbesondere das sogenannte *Internet der Dinge*, das zahlreiche netzwerkfähige Alltagsgegenstände, sowie mobile Geräte wie Mobiltelefone und Tabletts einbezieht, ist hier als zukünftiges Einsatzszenario prädestiniert. Im Zusammenspiel ergänzen sich dabei die leistungsfähigen Mobilgeräte, die aufgrund ihrer Funktionalität typischerweise individuellen Anwendern zugeordnet werden können, und die weiträumigen Mess- und Analysefunktionen drahtloser Sensorknoten.

Zur Anwendungskoordination in heterogenen Netzwerkumgebungen zählt daher auch die Allokation von geeigneten Knoten und Hardwareresourcen, damit verteilte Anwendungen ihre Aufgaben erwartungsgemäß ausführen können. Dazu sind skalierbare Konzepte zur Kommunikation und Koordination erforderlich, um auch

auf ressourcenbeschränkten Kleinstgeräten, möglichst wenige Einschränkungen und Kompromisse eingehen müssen.

2.2.3 Middleware zur Virtualisierung des Sensornetzes

Die Allokation geeigneter Knoten und der erforderlichen Hardwareressourcen kann durch zusätzliche Middleware im Sensornetz erfolgen. Diese wird typischerweise zwischen Betriebssystem und den Anwendungen mit dem Zweck realisiert, die verschiedenen Anwendungen zu kapseln.

Zugleich soll die Middleware eine zusätzliche Abstraktion zum koordinierten Zugriff auf die Hardware der Sensorknoten bieten, die speziell in Hinblick auf den konkurrierenden Ressourcenzugriff, über eine vom Betriebssystem gebotene Hardwareabstraktion hinausgeht. Dabei sollte die Middleware selbst eine möglichst leichtgewichtige Architektur aufweisen, die den Mehraufwand für Speicher-, Rechenzeit- und Energiebedarf nicht signifikant erhöht. Ein weiteres Ziel ist die Vermeidung negativer Beeinflussung der Übertragungseigenschaften, wie z. B. Latenz und Durchsatz der Anwendungen durch die Middleware.

Die Konzepte früher Middleware-Entwürfe wie *Impala* [73] sind bereits in die Entwicklung aktueller Betriebssystemversionen eingeflossen und sind verantwortlich für ein grundlegend modulares Design und asynchrone Ereignissignalisierung zwischen Anwendungs- und Betriebssystemkomponenten. Problematisch hat sich jedoch die damit verbundene starre Koppelung der Komponenten für den Betrieb mit mehreren Anwendungen erwiesen.

Eine übliche Technik der Middleware zur Unterstürzung mehrerer Anwendungen stellt die Virtualisierung von Hardwareressourcen der Sensorknoten dar. Für eine Virtualisierung zum Betrieb konkurrierender Anwendungen können die folgenden Kernbereiche identifiziert werden.

Virtualisierung des Speichers: Um einen Konflikt der Anwendungen um die verfügbare Speicherressourcen zu verhindern, muss die Middleware geeignete Techniken bereitstellen, die den Zugriff auf Adressbereiche verhindert, die von fremden Anwendungen genutzt werden (*Application-Isolation*). Dazu eignen sich beispielsweise Ansätze wie Maté [66], die auf Basis von virtuellen Maschinen eine Kapselung von sehr kleinen Anwendungen vornehmen und dabei von komplexen Hardwareoperationen weitgehend abstrahieren. Zusätzlich sind Funktionen der Middleware zum dynamischen Ausbringen und Nachladen von Programmkomponenten zur Laufzeit hilfreich, um die vorhandenen Speicherressourcen adaptiv auslasten zu können.

Virtualisierung der Sensorik: Zwei Anwendungen können etwa dann um den gleichen Sensor konkurrieren, wenn periodische Messungen mit unterschiedlicher Messfrequenz, -intervallen oder -genauigkeiten erforderlich sind. Auch eine für beide Anwendungen unterschiedliche Konfiguration oder Kalibration des Sensors kann zu Anwendungskonflikten führen. Die Aufgabe der Middleware ist die Koordination der Zeitpläne (*Scheduling*) und das Bereit-

stellen entsprechender Pufferspeicher für Messdaten, sowie die Zuordnung der Messdaten zu Anwendungen gemäß ihrer Anforderungen.

Virtualisierung von Zeitgebern: Sensornetze sind sehr reaktive Systeme, deren Anwendungsstrukturen stark durch periodisch ablaufende Prozesse und Unterbrechungen durch definierte Ereignisse geprägt sind. Die Sensorknoten besitzen oft nur einen Zeitgeber, so das eine Abstraktion für parallel stattfindende Prozesse erforderlich ist. Auch im Zusammenspiel mit Schlaf- und Wachphasen der Sensorknoten ist die Virtualisierung von Zeitgebern erforderlich, um Mess- und Verarbeitungsprozesse mit dem Medienzugriff zur Kommunikation zwischen den Knoten zu koordinieren. Eine Middleware muss daher in der Lage sein, Zeitgeber-Ereignisse den korrekten Prozessen und Anwendungen zuzuordnen.

Virtualisierung des Netzwerkzugriffs: Unterschiedliche Anwendungen können bei gemeinsamer Nutzung eines Sensornetzes aufgrund ihres Kommunikationsmusters unterschiedliche Anforderungen hinsichtlich der gewünschten Netzwerkkonfiguration haben. Dies betrifft vordringlich die zugrunde liegende Netzwerktopologie und damit verbundene Eigenschaften wie Netzwerkkapazitäten, Sendereichweiten der Knoten, Latenzen oder die Zuverlässigkeit der Übertragungen. Zudem muss sichergestellt werden, dass der Verkehr fremder Anwendungen sich nicht hinderlich auf eine Anwendung auswirkt (*Traffic-Isolation*) oder das Netzwerk in Partitionen zerfällt, die eine netzwerkweite Kommunikation verhindern. Daher muss eine Middleware effektive Maßnahmen zur Topologiekontrolle vorsehen, die ein anwendungsspezifisches Routing, *Overlay-Netze* (vgl. *SenShare* [63]) und eine Koordination der Duty-Cycles verschiedener Anwendungen erlauben.

Eine beispielhafte Middleware-Umsetzung findet sich in *DAViM* [43, 79]. Von den Autoren werden neben der Kappselung von Anwendungen in virtuellen Maschinen insbesondere zusätzliche Module in ihrer Architektur vorgesehen, die ein dynamisches Nachladen von zusätzlichen Softwarekomponenten erlauben. So können neben den eigentlichen Anwendungen auch die virtuellen Maschinen durch Nachladen des Instruktionssatzes adaptiert werden. Zusätzlich koordiniert ein dedizierter Controller den konkurrierenden Zugriff auf Speicheradressen und virtualisiert das Scheduling und die Ereignissignalisierung der Anwendungen.

Eine Übersicht früher und aktueller Virtualisierungs- und Middleware-Ansätze zur Unterstützung konkurrierender Anwendungen im gleichen Sensornetz findet sich in [35] bzw. [51].

2.2.4 Unabhängigkeit von anwendungsspezifischen Datensenken

Viele Anwendungen sind auf das Sammeln von Informationen mittels Sensorik und das Bündeln der Informationen in Richtung Senke ausgelegt. Deshalb ist der Datenverkehr zwischen Quellknoten und Senke vorherrschend, wogegen zwischen räumlich entfernten Quellknoten in der Regel kein Verkehr zu beobachten ist.

Üblich ist dabei eine stark anwendungsbezogene Ausrichtung der Netztopologie. Eine übliche Annahme existierender Middleware ist daher, dass das Sensornetz durch einen einzelnen Betreiber zur Verfügung gestellt wird (vgl. [43, 63]). Damit verbunden ist die allgemeine Annahme, dass nur eine einzelne Senke im Netzwerk verfügbar ist, die Daten verschiedener Anwendungen sammelt und zusammenführt oder dass einzelne anwendungsspezifische Senken existieren, die Sensormessdaten über wenige ausgewählte Knoten am Netzwerkrand erhalten. Kommunikationsszenarien in denen mehrere Senken die gleiche Anwendung unterstützen bilden bisher die Ausnahme und werden oft als Spezialfall behandelt oder vermieden.

Wechselt stattdessen die Art, Anzahl, Position und Existenz der Senke mit der ausgebrachten Anwendung, so werden *senkenunabhängige Kommunikationsmuster* benötigt, welche die Existenz einer einzelnen Senke, zu der eine Datenübertragung gerichtet ist, nicht voraussetzen.

Während einzelne Anwendungen mobile Senken voraussetzen können, ist auch die Nutzung von mehreren Senken etwa zur parallelen Nutzung von Sensorinformationen durch mehrere Nutzer möglich. Eine adaptive Middleware muss dazu erkennen, dass mehrere Anwendungen und/oder Senken gleichzeitig im Netz aktiv sind. Auch der Betrieb eines Sensornetzes in dem zeitweise keine Senke vorhanden ist oder sich die Position der Senke ändert, muss berücksichtigt werden.

2.3 Anwendungsspezifische Kommunikationsmuster

Als drahtlose, datenzentrierte Netzwerke basieren Sensornetze auf Paketvermittlung. Daher ist die folgende Diskussion auf Kommunikationsmuster in paketvermittelnden Netzwerken eingeschränkt.

Definition 1. ***Kommunikationsmuster:*** *Für die Ende-zu-Ende-Kommunikationsbeziehungen, die aus einer Anwendung resultieren, sind in Multi-Hop-Netzen einzelne Weiterleitungsschritte erforderlich, um Pakete von einem Endsystem zum anderen zu übertragen. Werden die Weiterleitungsschritte und Ende-zu-Ende-Kommunikationsbeziehungen zusammen mit der Kommunikationsrichtung und der übermittelten Datenmenge auf eine Netzwerktopologie abgebildet, so wird dies als Kommunikationsmuster bezeichnet.*

Zur analytischen Diskussion in dieser Arbeit werden die Endsysteme jeweils entsprechend ihrer Rolle in der Anwendung als *Datensenke* oder *Datenquelle* der erzeugten und transportierten Anwendungsdaten bezeichnet.

Kommunikationsmuster bieten darüber hinaus einige bemerkenswerten Eigenschaften zur Analyse der Kommunikation, auf die im Folgenden kurz eingegangen wird:

- Die Kommunikationsbeziehungen eines Kommunikationsmusters sind gerichtet. Die Enden werden entsprechend als *Sender* und *Empfänger* bezeichnet.
- Durch Betrachtung eines Kommunikationsmusters wird ein globales Bild der Kommunikationsvorgänge im Netzwerk vermittelt. Es ist daher zunächst

nicht für das dezentral Optimierungen einzelner Sende- und Empfangsvorgänge geeignet.

- Ähnlich wie Entwurfsmuster heute typischerweise mittels UML grafisch repräsentiert werden können, lassen sich Kommunikationsmuster grafisch z. B. in tabellarischer Matrixrepräsentation darstellen (vgl. [90]), indem die Sender- und Empfänger-Beziehungen zwischen jeweils zwei Knoten mit der übermittelten Datenmenge vermerkt werden.
- Ändert sich die zu Grunde liegende Netzwerktopologie, so ändert sich auch das darauf abgebildete Kommunikationsmuster. Im Folgenden wird daher jeweils nur ein einzelner Zeitpunkt betrachtet.

Klassische Basiskommunikationsmuster sind beispielsweise *Unicast*, *Broadcast*, *Multicast* oder *Convergecast* (in einzelnen Publikationen auch kurz *Concast* genannt). In Multi-Hop-Szenarien können diese Kommunikationsmuster zu komplexeren Kommunikationsmustern mit neuem Kommunikationsverhalten kombiniert werden. Zudem werden auch einige stärker anwendungsorientierte Kommunikationsmodelle, wie z. B. *Publish-Subscribe* als Kommunikationsmuster bezeichnet. Diese können jedoch auf eine Kombination einfacherer Basiskommunikationsmuster zurückgeführt werden.

Abbildung 2.1 zeigt im oberen Teil die grafische Darstellung eines exemplarischen Kommunikationsmusters für das Kommunikationsszenario im unteren Teil der Abbildung. Das Kommunikationsmuster zeigt dabei die ausgehenden Datenübertragungen eines Knotens (Sender), zu einem benachbarten Knoten (Empfänger). Die Farbe der Einträge unterscheidet zwischen der Datenübertragung zur Weiterleitung (orange) zu einem Nachbarknoten und der einer Ende-zu-Ende-Kommunikationsbeziehung zwischen Datenquellen und -senken (blau) bei der im Allgemeinen eine Multi-Hop-Übertragung über weitere Zwischenknoten erforderlich ist. Die Farbintensität repräsentiert die Datenmenge die über eine Kommunikationsbeziehung ausgetauscht wird. Dunklere Farben stehen für eine höhere Datenmenge.

In dem Bereich des Kommunikationsmusters, wo Datenquellen als Sender und Datensenken als Empfänger eingetragen sind (im Beispiel ist das der obere rechte Bereich), ist die Datenmenge eingetragen, die aus Sicht der Anwendung letztlich Ende-zu-Ende zwischen Quellen und Senken übertragen wird. Dabei bleiben die Zwischenknoten im Netzwerk unberücksichtigt. Eine Übertragung durch Weiterleitung an Nachbarknoten kann in diesem Bereich ebenfalls verzeichnet sein, z. B. falls Datenquellen und -senken in einer Topologie direkt benachbart sind und eine direkte Weiterleitung durchführen (vgl. Abb. 2.2, Beispiel ①).

Die untere linke Hälfte des Kommunikationsmusters beinhaltet Kommunikationsbeziehungen in Rückrichtung. Bei unidirektional gerichteten, zyklenfreien Netzen kann die Reihenfolge der Knoten so angeordnet werden, dass der Teil des Kommunikationsmusters unter der Diagonalen keine Einträge enthält. Entsprechend

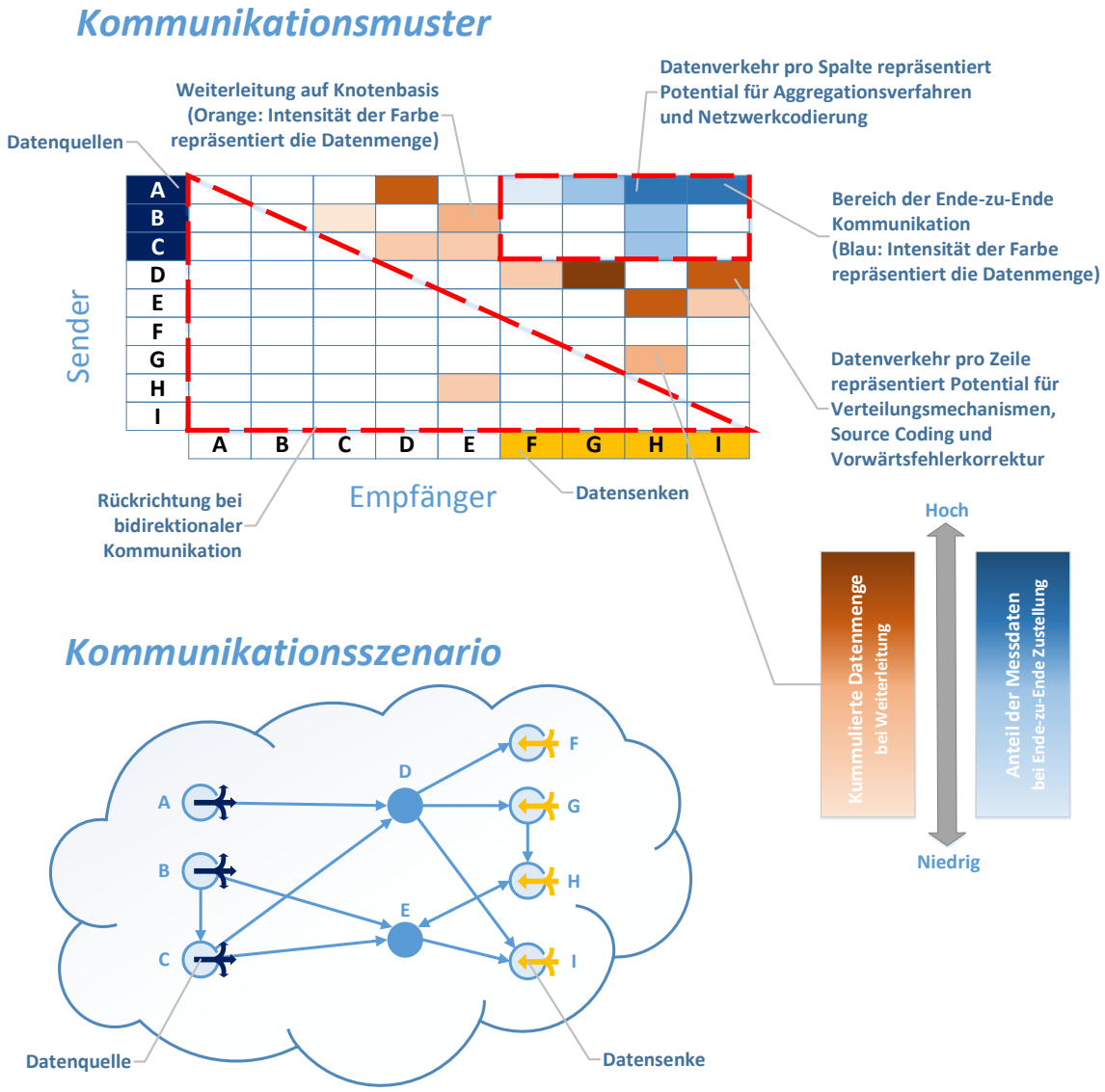

Abbildung 2.1 Kommunikationsmuster und entsprechendes Kommunikationsszenario

werden bidirektionale Kommunikationsbeziehungen in beiden Hälften des Kommunikationsmusters eingetragen.

Das Kommunikationsszenario im unteren Teil der Abbildung zeigt die Netzwerktopologie, die durch Abbildung des angegebenen Kommunikationsmuster schließlich resultiert.

Aus der graphischen Darstellung des Kommunikationsmuster können nun Schlüsse hinsichtlich Kommunikationsverhaltens und insbesondere des *Weiterleitungsverhaltens* gezogen werden. Durch Bestimmung von zusammenhängenden Kommunikationsbeziehungen kann dann mögliches Optimierungspotential identifiziert werden.

Eine Optimierung des Weiterleitungsverhaltens ist dann möglich, wenn sich innerhalb einer Spalte oder einer Zeile mehrere Einträge in *Spalten-* oder *Zeilenform* befinden.

Einträge in Spaltenform zeigen an, dass ggf. mehrere Pakete aggregiert oder zu größeren Paketen zusammengefasst werden können.

Sind dagegen mehrere Einträge zur Weiterleitung in eine Zeile enthalten, bedeutet das für weitergeleitete Daten, dass mehre Empfänger von den Daten eines sendenden Knotens profitieren. Zur Optimierung des Weiterleitungsverhaltens kann daher oft an Sendevorgängen gespart werden, wenn die Empfänger identischer Daten zum gleichen Zeitpunkt bedient werden können. Dies ist zum Beispiel dann möglich, wenn statt einzelnen Unicast-Übertragungen mit direkter Adressierung der Empfänger auf eine Übertragung mit Gruppenadressierung zurückgegriffen wird.

Tritt die Zeilenform für die Ende-zu-Ende-Übertragung auf, so kann die Nutzung von Verfahren für *Source-Coding* bzw. *Vorwärtsfehlerkorrektur* zweckmäßig sein. Insbesondere bei größeren Datenströmen kann in solchen Kommunikationsszenarien dann davon profitiert werden, dass Übertragungsfehler bei Ende-zu-Ende-Übertragungen nicht für jede Datensenke individuell behandelt werden müssen. Stattdessen werden die Übertragungsfehler durch redundante Daten ausgeglichen, die durch die Datenquelle erzeugt werden können und in hintereinanderfolgenden Pakete entlang des Pfades zu den Senken übermittelt werden.

Besonders vielversprechend für eine Optimierung des Kommunikationsverhaltens sind Bereiche des Kommunikationsmusters, bei dem es zur Blockbildung kommt, also mehrere Spalten mit vielen identischen Zeileneinträgen vorhanden sind. Die Datenmenge von Einträgen, die zu einem Block zusammengefasst werden, muss nicht notwendigerweise identisch sein. Vielmehr ist Blockbildung ein Hinweis darauf, dass mehrere Verbindungen für Datenverkehr aus konkurrierenden Kommunikationsbeziehungen genutzt wird. Ziel einer Optimierung ist in dieser Situation eine gemeinsame und koordinierte Übermittlung des konkurrierenden Datenverkehrs, um Kapazitätsengpässe zu vermeiden und die vorhandenen Ressourcen bestmöglich auszunutzen.

2.3.1 Typische Kommunikationsmuster drahtloser Sensornetze

Energieeffizienz ist in drahtlosen Sensornetzen das entscheidende Kriterium für die Lebenszeit des Netzes und die Verfügbarkeitsdauer der Anwendungen. Typischerweise hat die drahtlose Kommunikation eines Sensorknotens einen großen Anteil am Energiebedarf des Gesamtsystems, so dass die Protokolle aller Netzwerkschichten hinsichtlich Effizienz entworfen werden. Um die Ende-zu-Ende-Kommunikation möglichst effizient gestalten zu können, basieren die meisten gebräuchlichen Kommunikationsmuster in Sensornetzen daher auf einer Baumtopologie. Die Kommunikationsmuster zeichnen sich vorrangig durch einen unidirektionalen Datenverkehr und durch Zyklenfreiheit aus. Abseits von Clusterbasierten Netzwerkprotokollen für Sensornetze wird eine starke Vermaschung in Sensornetzen zu Gunsten der Effizienz eher vermieden.

Weitere verbreitete Mechanismen zur Steigerung der Energieeffizienz, die vordringlich zur Topologiekontrolle von Sensornetzen genutzt werden, sind das Reduzieren der Anzahl der Verbindungen zwischen Knoten z. B. durch Reduktion der Sendestärke und das Reduzieren der Anzahl aktiver Knoten. Beides führt dazu, dass sich ein Kommunikationsmuster über die Zeit ändert und sich ein geeignetes Protokoll an das aktuelle Muster adaptieren muss.

Drahtlose Sensornetze werden durch ihre datenzentrierte Topologie geprägt, bei der die relative Position von Datensenken und Datenquellen im Mittelpunkt steht. Die Baumtopologie mit gerichteten Kommunikationsverbindungen bewirkt somit, dass Sender und Empfängerrollen der Knoten in den Kommunikationsmustern eindeutig definiert sind. Diese Rollenzuteilung erleichtert insbesondere den koordinierten Medienzugriff und die Synchronisation des Knotenverhaltens.

Aus diesen Eigenschaften können drei typische Kommunikationsmuster für drahtlose Sensornetze abgeleitet werden, die bevorzugt Anwendung finden:

Aggregation: Aggregationsverfahren dienen dem Zusammenfassen von Daten oder Paketen, mit dem Ziel dass weniger Daten bzw. Pakete in Richtung Datensenke weitergeleitet werden müssen. Für die Aggregation von Daten werden üblicherweise mathematische Funktionen eingesetzt, die zu einer Reduktion des resultierenden Informationsgehaltes führen. Die Aggregation von Paketen fasst dagegen mehrere Pakete zu einem größeren Paket zusammen und ist üblicher Weise nicht verlustbehaftet. Dennoch kann die Kommunikation von einer reduzierten Anzahl von Sende- und Empfangsvorgängen durch aggregierte Pakete profitieren, die auch den Koordinationsaufwand zwischen den Knoten reduzieren kann. Das Kommunikationsmuster einer Anwendung, die Aggregation zum Sammeln von Daten einsetzt, basiert im Wesentlichen auf einem Convergecast-Szenario.

Dissemination: Sogenannte *Dissemination*-Verfahren zur Datenverteilung im Sensornetzwerk werden genutzt, um Anfragen nach Messdaten oder gemeinsame Zustandsinformationen zu verteilen (siehe auch Kapitel 4.3). Weitere Anwendungsgebiete umfassen beispielsweise das Ausbringen von Anwendungscode. Die meisten Anwendungen zur Datenverteilung setzen daher auf ein durch Multicast oder Broadcast beeinflusstes Kommunikationsmuster.

Einzelabruf von Daten: Der Einzelabruf von Daten einer Senke durch eine Quelle ist entweder durch ein Kommunikationsmuster möglich, dass per Multi-Hop mehrerer Unicast-Übertragungen kombiniert. Alternativ ist auch eine Kombination aus Multicast und Convergecast möglich, bei der zunächst eine Abfrage im Netzwerk verteilt wird, bis Knoten mit den gewünschten Eigenschaften auf die Anfrage mit eigenen Daten antworten. Die abgerufenen Daten werden dann per Convergecast in Rückrichtung eingesammelt.

Abbildung 2.2 zeigt beispielhaft die Kommunikationsmuster einiger typischer Kommunikationsszenarien von Sensornetzanwendungen. Zur Vereinfachung wur-

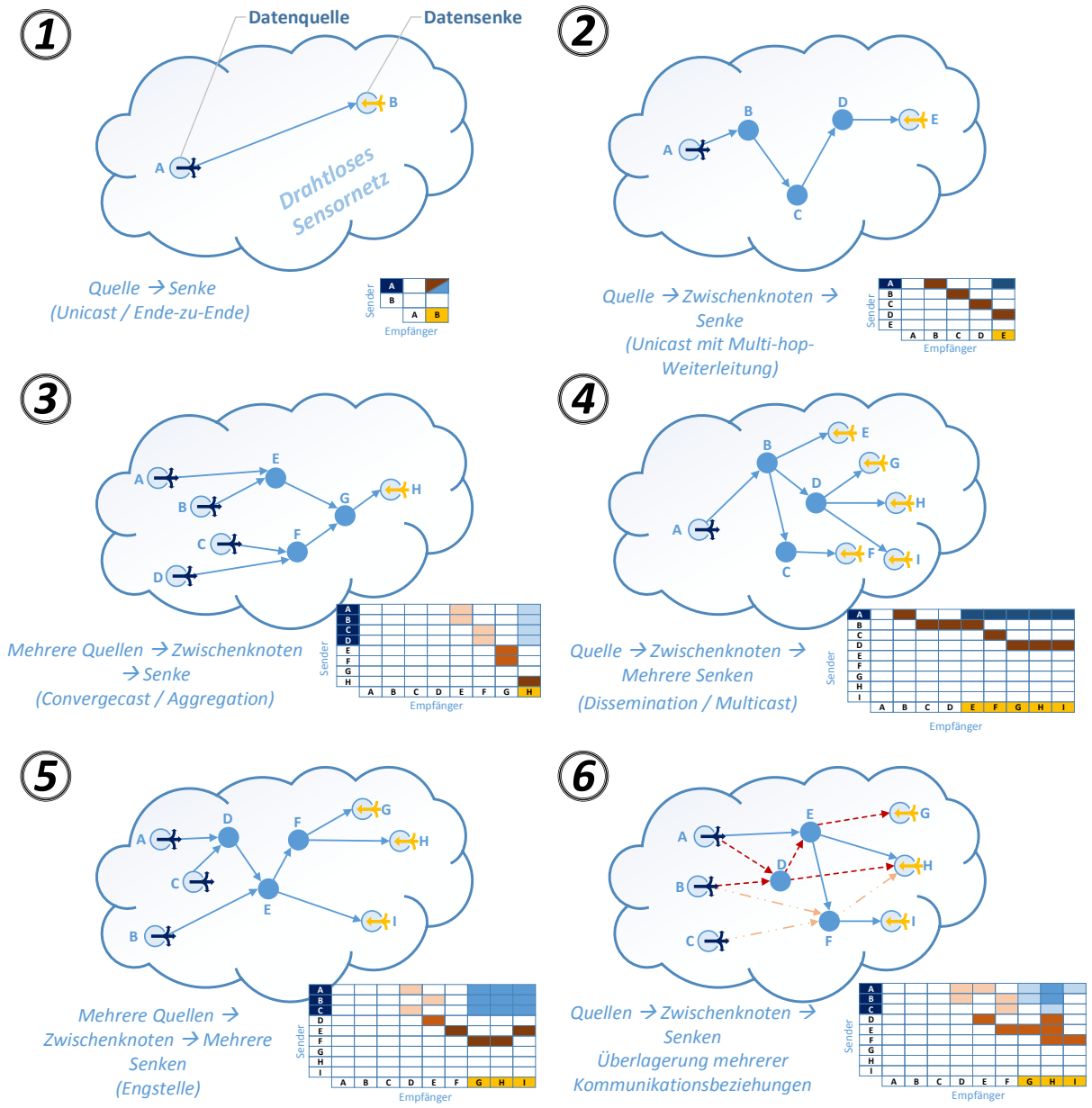

Abbildung 2.2 Beispiele für unterschiedliche Kommunikationsmuster in drahtlosen Sensornetzen

den nur Beispiele gewählt, die zwischen Datensenke und Datenquelle keine Zyklen oder Rückwärtspfade aufweisen. Somit ist jeweils nur die rechte obere Hälfte des Kommunikationsmusters von Interesse.

In ① ist eine direkte Kommunikation zwischen einer Datenquelle und der Datensenke abgebildet. Das Kommunikationsmuster, das in diesem Fall entsteht, nutzt die Verbindung zwischen den Knoten somit zur Weiterleitung, als auch zur direkten Ende-zu-Ende-Übertragung des vollständigen Datenverkehrs.

In Beispiel ② erfolgt die Ende-zu-Ende-Übertragung zwischen den Knoten A und E über mehrere Zwischenknoten per Multi-Hop-Weiterleitung. In Anwen-

dungen kann dieses Muster beispielsweise bei der gezielten Signalisierung von Ereignismeldungen Verwendung finden. Das Kommunikationsmuster zeigt in der grafischen Repräsentation für die Weiterleitung typische kaskadierende Stufen bei den sendenden Knoten. Auf dem Weg zur Quelle muss jeder Knoten den vollständigen Datenverkehr weiterleiten, der von der Quelle erzeugt wurde, so dass die Datenmenge der Knoten hier den Maximalwert zeigt. Eine Optimierung des Weiterleitungsverhaltens kann in diesem Szenario nicht erwartet werden.

Das Kommunikationsmuster für ein Convergecast-Szenario bzw. einen Aggregationsbaum zeigt Beispiel ③. Dieses Kommunikationsmuster ist typisch für zahlreiche Anwendungen, die das großräumige Sammeln und Überwachen von Messdaten beinhalten. Es kann festgestellt werden, dass die meisten Empfänger-Knoten mehrere Spalteneinträge aufweisen. Somit ist das Zusammenführen der weitergeleiteten Daten durch die jeweiligen Empfängerknoten mittels Aggregationstechniken oder größeren Pakete möglich, um ein effizienteres Weiterleitungsverhalten zu erzielen. Zugleich können die Knoten in einer Reihenfolge in der Tabelle dargestellt werden, so dass eine ähnliche Stufenbildung wie im Multi-Hop-Szenario auch hier bei der Weiterleitung auftritt. Während der Anteil der weitergeleiteten Daten an der Menge der erzeugten Daten in den ersten Stufen noch gering ist, nimmt die Menge des Datenverkehrs in Richtung Datensenke zu, so dass ein potentieller Kapazitätsengpass in der Nähe der Senke zu erwarten ist.

Beispiel ④ zeigt das Kommunikationsszenario eines Multicast-Baumes, wie er typischerweise bei Anwendungen zur Datenverteilung Verwendung findet. Jeder Zwischenknoten muss dabei typischer Weise den vollständigen Datenverkehr der Quelle weiterleiten. Daneben ergibt sich ein Kommunikationsmuster mit einer Tendenz zu vielen Zeileneinträgen. In Konsequenz kann in diesem Szenario davon profitiert werden, wenn Zwischenknoten beim Senden die Daten zeitgleich an mehrere Empfänger übermitteln, beispielsweise durch lokales Broadcasting. Da sich auch die Ende-zu-Ende-Übertragung in Zeilenform darstellt, kann die Nutzung von Verfahren zur Vorwärtsfehlerkorrektur bzw. für Source-Coding bei Datenströmen lohnen.

Das Kommunikationsszenario in Beispiel ⑤ zeigt ein Netzwerk, das Eigenschaften eines Convergecast- und Multicastszenario vereint. Im Gegensatz zu den vorherigen Szenarien liegt hier keine Baumstruktur zu Grunde. Dabei wird der Datenverkehr der durch die Datenquellen *A* bis *C* erzeugt wird vollständig über den Knoten *E* geleitet und anschließend an die Senken *G* bis *I* verteilt. Der Knoten *E* bildet dabei eine mögliche Engstelle für den Netzwerkdurchsatz. Die Grafik des Kommunikationsmusters zeigt für die Ende-zu-Ende Übertragung eine deutliche Blockbildung, die für ein Zusammenfassen und Codieren der übertragenen Daten spricht. Die weiterleitenden Zwischenknoten zeigen jedoch ein vollständig anderes Kommunikationsverhalten. Zur Weiterleitung verspricht ein Verhalten des Knotens *E* hohe Effizienz, bei dem zunächst die eintreffenden Daten der Knoten *B* und *D* gesammelt werden, bevor eine gemeinsame Weiterleitung erfolgt.

Beispiel ⑥ zeigt schließlich ein Szenario, bei dem die Kommunikationsbeziehungen mehrerer Anwendungen überlagert wurden. Dabei wird erkennbar, dass sich in der grafischen Abbildung des Kommunikationsmusters verstärkt Blöcke aus Zeilen und Spalteneinträgen bilden. Die Blockbildung kann sowohl bei den Ende-zu-Ende-Übertragungen, als auch bei der Weiterleitung beobachtet werden. Dies kann als deutlicher Hinweis interpretiert werden, dass die Effizienz des Kommunikationsmusters durch ein Zusammenfassen und Codieren der übermittelten Daten gleichzeitig für sequentielle Daten eines Quellknotens, als auch für Daten aus verschiedenen Quellen verbessert werden kann. In Kapitel 3 wird dazu das Konzept der *Netzwerkcodierung* eingeführt, die dies Knoten innerhalb des Netzwerks ermöglicht.

2.3.2 Problematik multipler Senken und Quellen bei überlagerter Kommunikation

Überlagert sich der Datenverkehr unterschiedlicher koexistierender Anwendungen in einem Netzwerk, so überlagern sich auch die entsprechenden Kommunikationsmuster und Kommunikationsszenarien der jeweiligen Anwendungen.

In Abbildung 2.3 wird die Überlagerung der anwendungsspezifischen Kommunikation von drei unterschiedlichen Anwendungen exemplarisch dargestellt. Die Überlagerung führt zu einem neuen kombinierten Kommunikationsszenario, bei dem Datenquellen und Datensenken der überlagerten Anwendungsszenarien übernommen werden.

Die Überlagerung zieht nach sich, dass eine Verbindung zwischen zwei Knoten nicht länger gerichtet ist, wenn zwei Anwendungen gegenläufige unidirektionale Verbindungen aufweisen. Diese werden im kombinierten Kommunikationsszenario zu bidirektionalen Verbindungen ergänzt.

Knoten des kombinierten Kommunikationsszenarios können sowohl die Rolle einer Datenquelle für eine Anwendung spielen, während sie die Datensenke einer weiteren Anwendung sind. Somit

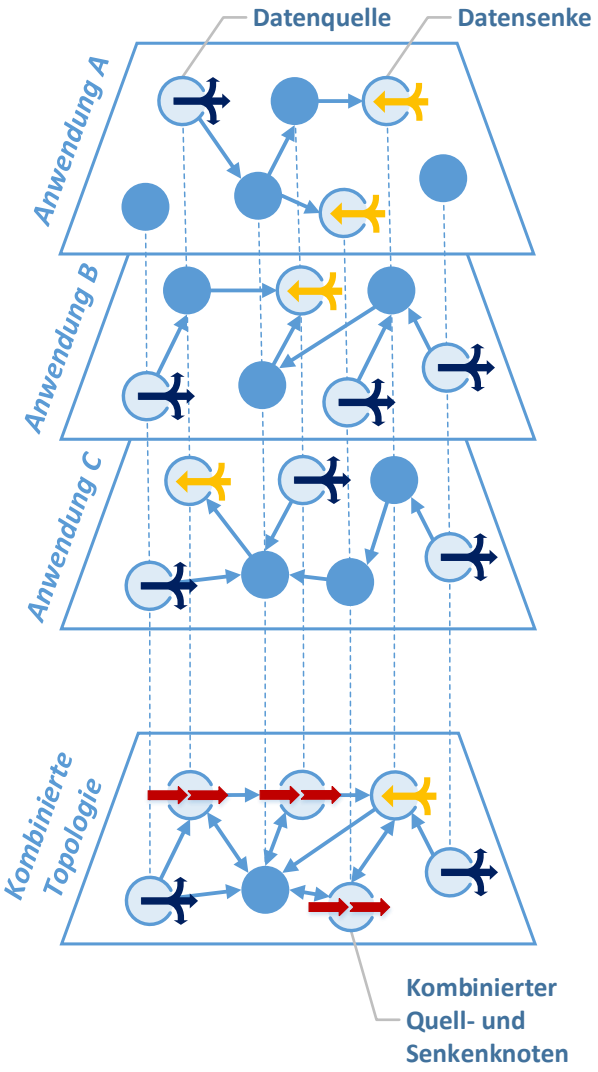

Abbildung 2.3 Kombination der Kommunikationsmuster verschiedener Anwendungen

sind ihre Rollen nicht länger eindeutig. Außerdem existieren nicht länger zwischen jedem Quellen-Senken-Paar Kommunikationsbeziehungen.
Das typische Kommunikationsmuster, das bei dieser Überlagerung entsteht wird im Folgenden als *Multi-Source-Multicast-Kommunikationsmuster* bezeichnet.

Definition 2. ***Multi-Source-Multicast:*** *Als Multi-Source-Multicast wird das Kommunikationsmuster bezeichnet, das entsteht, wenn mehrere Multicast-Kommunikationsmuster mit unterschiedlichen Quellen derart überlagert werden, dass mindestens ein weiterleitender Knoten oder eine Verbindung zwischen zwei Knoten der unterliegenden Netzwerktopologie von mehreren der Multicast-Kommunikationsmuster genutzt wird.*

Die folgenden Eigenschaften können Multi-Source-Multicast-Kommunikationsmuster aufweisen:

- Datensenken können mit mehreren Datenquellen Daten austauschen, so dass das Muster für diese Teilbeziehungen die Eigenschaften eines Convergecast-Musters aufweist.
- Ein Knoten kann sowohl Datenquelle, als auch Datensenke zugleich sein. In diesem Fall wird er als *Kombinierter Quell- und Senkenknoten* bezeichnet.
- Der Datenverkehr zwischen zwei beteiligten Knoten ist nicht länger nur in eine Richtung gerichtet (Simplex) wie in den zugrunde liegenden Multicast-Kommunikationsmustern. Vielmehr können auch Vollduplex-Verbindungen zur Datenübermittlung zwischen zwei Knoten entstehen wenn Quellen und Senken der überlagerten Multicast-Kommunikationsmuster in entgegengesetzten Netzwerkbereichen liegen.
- Die Struktur eines Multi-Source-Multicast-Kommunikationsmusters kann vollständig in einzelne überlagernde Convergecast-Kommunikationsmuster zerlegt werden. Daher ist die Struktur jedes Multi-Source-Multicast-Musters äquivalent zu einem entsprechenden *Multisink-Convergecast*-Muster.
- Im Gegensatz zu den ursprünglichen Multicast-Bäumen kann das resultierende Kommunikationsmuster redundante Pfade zwischen Quellen und Senken aufweisen, die partiell oder vollständig disjunkt sind.
- Ein Multi-Source-Multicast-Kommunikationsmuster ist im Allgemeinen nicht zyklenfrei und weist keine Baumstruktur auf.
- Der mittlere Knotengrad und somit die mittlere Anzahl der Nachbarn pro Knoten ist durch die Überlagerung größer oder gleich im Vergleich zu den zugrunde liegenden Kommunikationsmustern.

Während die Überlagerung von Kommunikationsmustern zusätzliche Optimierungsmöglichkeiten für effizientere Datenübertragung bietet, führt sie zugleich zu neuen Schwierigkeiten für Routingstrategien. Insbesondere können sich einzelne

Abschnitte eines Routingpfades durch die Kombination der Kommunikationsmuster als kritische Beschränkungen für den resultierenden Datendurchsatz erweisen und es müssen ggf. zusätzliche Maßnahmen gegen Zyklenbildung ergriffen werden. Einen Lösungsansatz für Routing in Multi-Source-Multicast-Szenarien stellt z. B. [99] vor. Dabei wird eine clusterbasierte Routingstrategie verwendet, die aber wiederum zu sehr langen Pfaden zwischen Quellen und Senken führen kann.

2.4 Zusammenfassung

Zunächst wurde in diesem Kapitel die Geräteklasse der drahtlosen Sensornetzen eingeführt. Diese zeichnen sich neben ihren beschränkten Hardwareressourcen insbesondere durch ihre anwendungsspezifische Ausrichtung aus. Sollen mehrere Anwendungen in einem Sensornetzwerk koexistieren, so werden bereits durch die verfügbare Speicherkapazität auf vielen Geräten harte Grenzen gesetzt. Auswege können geeignete Middlewaresysteme darstellen, die neben dem Speichers auch weitere Komponenten eines Sensorknotens virtualisieren können. Sollen Anwendungen darüber hinaus auch unabhängig von vorhandenen Datensenken operieren können, so sind zusätzliche Anforderungen bezüglich der Adaptivität an ein Middlewaresystem zu stellen.

Im Anschluss erfolgte in diesem Kapitel die Definition des Begriffs *Kommunikationsmuster* und der Betrachtung, wie ein solches in einer aussagekräftigen grafischen Repräsentation dargestellt werden kann. Die typischen Kommunikationsmuster drahtloser Sensornetze wurden identifiziert und Konsequenzen für ihr Kommunikationsverhalten erläutert. Einen weiteren Schwerpunkt des Kapitels bildet die Betrachtung zur Kombination unterschiedlicher anwendungsspezifischer Kommunikationsmuster in einem kombinierten *Multi-Source-Multicast-Kommunikationsmuster*. Dieses bietet wiederum eine Grundlage für die Nutzung von Netzwerkcodierung, die im folgenden Kapitel nun detaillierter vorgestellt wird.

3. Netzwerkcodierung in drahtlosen Sensornetzen

Das Konzept der *Netzwerkcodierung* wurde im Jahr 2000 von [2] als neuer Ansatz etabliert, um damit gängige Kapazitätsbeschränkungen in Multicast-Netzwerken zu umgehen.

Dem Ansatz liegt ein informationstheoretisches Modell zugrunde, das $q = |Q|$ unabhängige Informationsquellen und $s = |S|$ Informationssenken vorsieht, die jeweils über ein Netzwerk mit beschränkter Kapazität $\mathcal{K}_{Netz}$, wie in Abbildung 3.1 dargestellt, kommunizieren. Die Kommunikation zwischen den Quellen $Q_1, \ldots, Q_n$ und den Senken $S_1, \ldots, S_m$ wird dazu als Informationsfluss betrachtet.

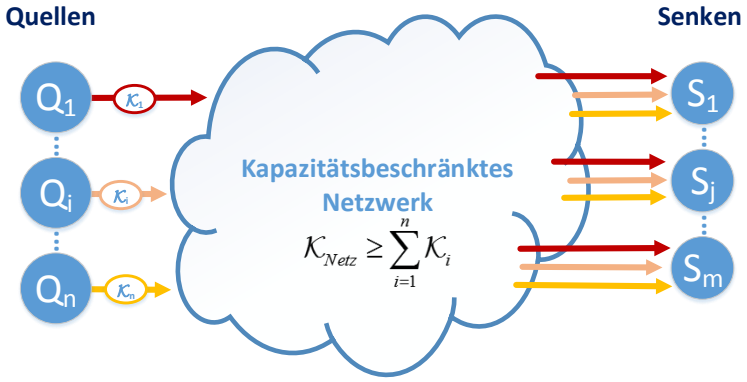

Abbildung 3.1 Kommunikationsszenario der Netzwerkcodierung

Während die Kapazität des Netzwerkes zwar ausreicht, die Informationen der Quellen an jede einzelne Senke zu transportieren, kann im Allgemeinen für Routing-

Strategien nicht geschlussfolgert werden, dass die Informationen zeitgleich an alle Senken übertragen werden können.

Die Kernidee zur Lösung dieses Multicast-Problems liegt darin, sich hier von der Vorgehensweise des klassischen Routings zu lösen, bei der ein Datenpaket durch Knoten zwischen Quelle und Senke ausschließlich weitergeleitet wird und die eigentlichen Nutzdaten unverändert im Paket gekapselt bleiben.

Die Netzwerkcodierung sieht zur Lösung des Problems daher die aktive Beteiligung von Zwischenknoten im Netzwerk auf dem Pfad zwischen Quellknoten und Senkenknoten vor. Die Zwischenknoten sollen nicht nur die erhaltenen Informationen in Richtung Senke weiterleiten, sondern werden aktiv dazu genutzt, die Informationen von verschiedenen Quellen vor der Weiterleitung gemeinsam zu codieren. Ebenso müssen die Zwischenknoten auch in der Lage sein, erhaltene Informationen vor der Weiterleitung wieder zu decodieren. Durch diese Vorgehensweise lassen sich insbesondere Verbindungen mit Kapazitätsengpässen in der Netzwerktopologie effizienter nutzen, die für den gemeinsamen Informationsfluss unterschiedlicher Quellen benötigt werden. Allerdings müssen zu diesem Zweck auf den Zwischenknoten auch die nötigen Ressourcen zur Verarbeitung, wie zusätzliche Rechenleistung und vor allem zusätzlicher Speicher, vorgesehen werden.

Während sich die ursprüngliche Idee der Netzwerkcodierung konzeptionell vordringlich auf kabelgebundene Netzwerke konzentriert, ist die sinnvolle Erweiterung auf drahtlose Netzwerke und Sensornetzwerke im Besonderen möglich.

In diesem Kapitel wird hierzu zunächst die grundlegenden Problematik vorgestellt, die zur Einführung des informationstheoretischen Konzeptes der Netzwerkcodierung führen. Dann werden typische Nutzungsmöglichkeiten des Konzepts aufgezeigt. Anschließend werden die nötigen Begriffe definiert und Grundlagen hergeleitet, die für die verwendeten Codierungsverfahren benötigt werden. Es folgt eine Diskussion typischer Codierungsverfahren mit Betrachtung ihrer Eignung für die Verwendung in drahtlosen Sensornetzen. Dazu werden im Anschluss die existierenden Forschungsarbeiten mit besonderem Fokus auf die Verwendung von Netzwerkcodierung in drahtlosen Sensornetzen analysiert und bewertet. Das Kapitel schließt mit der Identifikation verbleibender Kernproblematiken bei der Nutzung von Netzwerkcodierung in drahtlosen Sensornetzen.

Beschränkung des Durchsatzes durch Max-Flow-Min-Cut-Theorem

Grundlage der Kapazitätsbetrachtung eines gegebenen Netzwerkes bildet das sogenannte *Max-Flow-Min-Cut*-Theorem [25]. Die Gegebenheiten in einem Netzwerk mit lediglich einer einzelnen Quelle und einer einzelnen Senke werden dazu zu einem gerichteten azyklischen Graphen abstrahiert. Ziel der Betrachtung ist eine Analyse, ob die Quelle Daten einer bestimmten Kapazität innerhalb eines Zeitschrittes an die Senke senden kann.

Die Kanten des Graphen entsprechen der Kommunikationsrichtung im Netzwerk zwischen Quelle und Senke und die Kapazität der Kanten dem erreichbaren Durchsatz der Verbindung. Im Folgenden wird eine allgemeine Kapazität von 1 pro

Kante verwendet, analog zu einem Durchsatz von einem Paket fester Länge pro betrachtetem Zeitschritt.

Um zu analysieren wie viele Daten einen Abschnitt des Graphen passieren können, ist zunächst die Bestimmung der Kapazität eines Schnittes notwendig. Dazu werden die Knoten in zwei disjunkte Gruppen partitioniert. Während die Quelle in der ersten Gruppe enthalten ist, wird die Senke der zweiten Knotengruppe zugeordnet. Die aufsummierte Kapazität der Kanten zwischen den Knotengruppen ergibt die Kapazität des Schnittes $\mathcal{K}_{cut}$. Um mögliche Kapazitätsengpässe im Netzwerk zu betrachten, interessiert hier insbesondere der Schnitt mit der kleinstmöglichen Kapazität (*Min-Cut*).

Theorem 1. ***Max-Flow-Min-Cut:*** *Gegeben sei ein gerichteter azyklischer Graph mit einer Senke s und einer Quelle q und nicht-negativen Kantenkapazitäten. Die maximal erzielbare Kapazität (Max-Flow) in einem Graph zwischen q und s entspricht dann der Kapazität des minimalen Schnittes (Min-Cut).*

Somit richtet sich der nutzbare Durchsatz im Netzwerk jeweils nach dem kleinsten Engpass. Die Daten der Quelle können außerdem innerhalb eines Zeitschrittes zur Senke übermittelt werden, solange sie mit einem geeigneten Verfahren auf die einzelnen Verbindungen aufgeteilt werden und die Gesamtmenge sich unterhalb des minimalen Schnittes des Netzes befindet.

Problemstellung in Multicast-Netzen

Im Falle einer vorgegebenen Topologie mit einer einzelnen Quelle und Senke ist der erzielbare Durchsatz und eine geeignete Aufteilung der Datenströme mit dem *Ford-Fulkerson-Algorithmus* [25] ermittelbar. Wird das Kommunikationsszenario jedoch um weitere Quellen oder Senken erweitert und wird insbesondere Multicast-Kommunikation betrachtet, so kann mittels des Ford-Fulkerson-Algorithmus jedoch keine algorithmische Berechnung des maximalen Flusses mehr durchgeführt werden (vgl. [55]). Während das Max-Flow-Min-Cut Theorem als obere Schranke weiterhin eine gültige Aussage über das theoretische Kapazitätsmaximum des Netzes macht, ist die Erreichbarkeit des Max-Flow-Limits im Einzelfall nicht sichergestellt.

Ein Beispiel dazu ist das Netzwerk des sogenannten *Butterfly-Schemas* in Abbildung 3.2. In diesem Multicast-Kommunikationsschema sind die zwei Nachrichten M_1 und M_2 der Quelle Q an beide Senken S_1 und S_2 zu übertragen. Beide Nachrichten benötigen jeweils die Kapazität $\mathcal{K}(M_i) = 1$ zur Übertragung, so dass ein minimaler Schnitt $\mathcal{K}_{mincut} = 2$ zu jeder Senke erforderlich ist. Die Kapazität ist an den Pfeilspitzen der einzelnen Verbindungen abgebildet und beträgt hier jeweils 1.

Wie leicht erkennbar ist, existiert zu jeder Senke zwar der notwendige maximale Fluss nach dem Max-Flow-Min-Cut Theorem, dennoch können über den Kapazitätsengpass in der Mitte nicht beide Nachrichten in unveränderter Form übermittelt werden, um das theoretische Limit auszuschöpfen. Insbesondere klassische Routingverfahren in *Store-and-Forward*-Netzen sind hier betroffen, da die in Paketen

gekapselten Nachrichten typischerweise sequenziell und unabhängig voneinander durch das Netzwerk transportiert werden (vgl. [26, 38, 121]).

Allerdings wird am Beispiel auch deutlich, dass hier das theoretische Limit erreicht werden kann und somit eine Lösung für das Multicast-Fluss-Problem existiert. Dazu muss auf der Verbindung mit dem Kapazitätsengpass die Nachrichtenkombination M_1 XOR M_2 übermittelt werden. Bei den Senken S_1 und S_2 kann die Nachrichtenkombination durch erneutes XOR mit den Einzelnachrichten M_1 bzw. M_2 wieder aufgelöst werden.

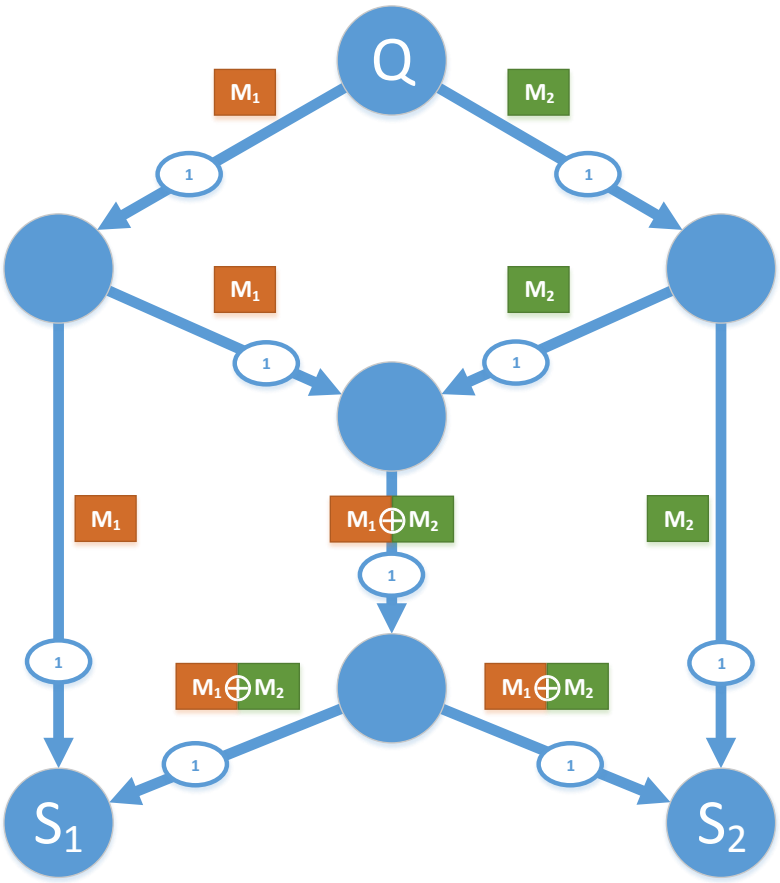

Abbildung 3.2 Maximaler Fluss mit mehreren Quellen im Butterfly-Schema (vgl. [2])

Lösung durch Netzwerkcodierung

In [2] konnte erstmals in einer graphentheoretischen Betrachtung des erzielbaren Informationsflusses in Multicast-Netzwerken gezeigt werden, dass sich die theoretische Grenze des Max-Flow-Min-Cut-Theorems auch praktisch erreichen lässt. Dazu muss eine Codierung verwendet werden, um Informationen auf kritischen Pfaden und Engpässen im Netzwerk zu transportieren, die sonst durch Einzelübertragungen in Routingverfahren zwischen verschiedenen Quell- und Senkenknoten

simultan genutzt würden. Essentieller Bestandteil der Forschungsarbeit ist das folgende und als *Hauptsatz der Netzwerkcodierung* bekannte Theorem:

Theorem 2. ***Hauptsatz der Netzwerkcodierung:*** *Gegeben sei eine gerichtete, schleifenfreie Netzwerktopologie. Wenn eine Quelle k Nachrichten mit fester Rate an jeweils jede einzelne von n Senken senden kann, so dass der Minimale Schnitt (Min-Cut) zwischen Quelle und jeder einzelnen Senke mindestens die Kapazität der k Nachrichten hat, dann ist auch eine Multicast-Übertragung an alle n Senken gleichzeitig möglich. Dazu müssen die Quellnachrichten per Netzwerkcodierung linear kombiniert werden.*

Eine weitere Erkenntnis aus [2] ist, dass das Finden einer optimalen Codierung für mehrere unabhängige Informationsquellen (Multi-Source-Problematik) nicht trivial lösbar ist. Das gilt speziell für den allgemeinen Fall, bei dem jede Senke eine individuelle Teilmenge der von den Quellen bereitgestellten Nachrichten benötigt.

In späteren Arbeiten wie [55] und [70] konnte darüber hinaus gezeigt werden, dass einige Einschränkungen des Hauptsatzes fallen gelassen werden können und eine Erweiterung möglich ist, ohne dass er seine Gültigkeit verliert. So ist es ebenso möglich, dass statt einer einzelner Quelle, die k Nachrichten sendet, auch k verschiedene Quellen jeweils mit einer normierten Rate gleichzeitig ihre Nachrichten an alle Senken senden können. Darüber hinaus konnte die Forderung nach einer festen Rate der Einzelquellen durch *ratenlose Codierung* (siehe Kapitel 3.3.1) gelockert werden. Außerdem konnte gezeigt werden, dass die Einschränkung hinsichtlich der Schleifenfreiheit fallen gelassen werden kann, die speziell für die praktische Realisierbarkeit der Netzwerkcodierung bedeutend ist.

Die Codierung durch Knoten innerhalb des Netzwerkes kann somit dazu beitragen, dass die vorhandenen Kapazitäten effizienter genutzt werden. Da durch Codierung insgesamt weniger Kapazitäten für die gleiche Datenmenge im Netzwerk benötigt werden, können frei werdende Kapazitäten auch genutzt werden, um mehr Daten zu transportieren, zusätzliche Redundanz und Robustheit zu realisieren oder die Belegungsdauer des Netzwerkmediums zu verringern. Eine Analyse des erreichbaren Durchsatzes bei ungerichteter Halbduplex-Kommunikation verspricht mit Netzwerkcodierung eine maximale Verbesserung um etwa den Faktor zwei [26].

3.1 Einsatzbereiche und Eigenschaften

Netzwerkcodierungs-Techniken sind auf unterschiedlichen Schichten eines Protokollstapels nutzbar. So kann Netzwerkcodierung auf der Anwendungsschicht beispielsweise genutzt werden, um eine verteilte Datenspeicherung zu realisieren oder eine effiziente Datenverteilung zu ermöglichen. Auf der Transportschicht kann Netzwerkcodierung eingesetzt werden, um die Zuverlässigkeit der Datenübermittlung zwischen den Endpunkten der Kommunikation zu verbessern. Hier wird zusätzliche Redundanz genutzt, um die Anwendungsdaten in vergleichbarer Weise mit sogenanntem *Source Coding* codiert zu übertragen. Paketverlusten kann dabei vorgebeugt werden, indem die Anwendungsdaten rekonstruiert werden, sobald ein ausreichend großer Anteil an Paketen die Senke erreicht.

Auf der Netzwerkschicht kann Netzwerkcodierung als Ersatz für gängige Routingstrategien verwendet werden [38, 71]. Ausgehend von Multicast-Kommunikation zwischen einer Menge von Quellen und einer Menge von Senken zielt die Netzwerkcodierung hier vorrangig auf eine Optimierung des Durchsatzes. Dazu können redundante Pfade zwischen Quelle und Senke ausgenutzt werden.

Inzwischen existieren auch Ansätze auf der Bitübertragungsschicht. Bei der Verwendung von sogenanntem *Physical Network Coding* wird durch Aufsummieren der durch eine Antenne empfangenen Signale und unter Verwendung angepasster Protokolle zum Medienzugriff die Rekonstruktion von Paketen bei kollidierenden Übertragungen ermöglicht [40, 50]. Im Rahmen dieser Arbeit liegt der Schwerpunkt der Betrachtung jedoch auf Ansätzen der Netzwerk- und Transportschicht.

Netzwerkcodierung findet im Wesentlichen zu den folgenden Bestimmungen im Netzwerk Verwendung (vgl. [27, 77]):

- Netzwerkdurchsatz und Auslastung vorhandener Netzwerkressourcen und -kapazitäten können mittels Netzwerkcodierung verbessert werden.

- Netzwerkcodierung kann für zuverlässigere Nachrichtenzustellung in Netzwerken mit unzuverlässigen Verbindungen genutzt werden. Insbesondere verteilte Ansätze können hier profitieren, die das Senden von Nachrichten über mehrere Pfade ausnutzen. Der Verlust von Einzelpaketen kann zugleich kompensiert werden, wenn eine Quellnachricht in mehreren Paketen übertragen wird, von denen nicht alle zur erfolgreichen Decodierung benötigt werden.

  Ein typischer Ansatz dieses Anwendungsbereiches wird in [5] präsentiert. Der Ansatz nutzt *Network Protection Codes* auf der Grundlage eines einfachen XOR-Schemas zur Netzwerkcodierung. Der Ansatz nutzt Network Protection Codes zur Fehlerkorrektur auf einzelnen Verbindungen und erweitert die Codes zur Fehlerkorrektur von mehreren Verbindungen auf Kosten der Netzwerkkapazität. Um einen geeigneten Network Protection Code mittels ganzzahliger linearer Optimierung zu berechnen, ist jedoch die Kenntnis des Gesamtnetzes erforderlich. Daher eignet sich der Ansatz nicht zur Anwendung in verteilten Kommunikationsszenarien wie Sensornetzen.

  Auch der Forschungsansatz in [31] nutzt Mehrwegeausbreitung in Kombination mit Netzwerkcodierung, um effiziente Fehlerbehandlung zu realisieren. Zur Untersuchung wurde ein Sensornetz in einer Unterwasserumgebung simuliert. Dabei konnte gezeigt werden, dass Network Coding gegenüber anderen Verfahren zur Fehlerbehandlung, wie ARQ und FEC, effizienter hinsichtlich Zustellrate und Bitfehlerrate auf Ende-zu-Ende-Verbindungen einsetzbar ist. Als entscheidend für eine hohe Effizienz hat sich dabei herausgestellt, dass zur Netzwerkcodierung 3-4 Nachbarn pro Knoten vorhanden sind und die Übertragung auf dem ersten Hop redundant durchgeführt wird, da Paketverluste sich hier noch besonders kritisch auswirken können.

- Verbesserung der Sicherheit gegen Lauschangriffe auf Einzelverbindungen zwischen Knoten. Durch das Senden von codierten Nachrichten ist ein potentieller Angreifer erst in der Lage die Nachricht zu decodieren wenn er die benötigten Pakete zur Decodierung mithören kann. Dazu reicht es nicht, eine einzelne Verbindung zu überwachen, wenn die benötigten Pakete über unterschiedliche Pfade und Verbindungen den Empfänger erreichen. Dieser Vorteil wird jedoch in drahtlosen Netzwerken relativiert, da ein Angreifer leicht in der Lage ist, dass gesamte Medium mit allen Verbindungen des Empfängers zu überwachen. Darüber hinaus kann die Netzwerkcodierung auch zu einer Verringerung der effektiven Datensicherheit führen, wenn der Angreifer in der Lage ist Zwischenknoten zu manipulieren, da die Codierung eine Verarbeitung der empfangenen Daten auf jedem Knoten vorsieht [40].

- Verbesserung der Energieeffizienz durch Reduktion der Sendevorgänge. Die Reduktion erfolgt vornehmlich durch Ausnutzen einer bekannten Vorverteilung der Informationen bei den Kommunikationspartnern. So können Informationen derart gemeinsam kodiert werden, dass sichergestellt werden kann, dass allen beteiligten Empfängern das Decodieren möglich ist und jeweils neue Informationen aus einer Übertragung gewonnen werden können.

  Exemplarisch ist dies in Abbildung 3.3 dargestellt.

  Der Quellknoten Q soll hier die Nachricht A an die Senke S_2 verteilen und außerdem Nachricht B an die Senke S_1. Zusätzlich ist Q die Information bekannt, dass S_1 bereits zuvor die Nachricht A empfangen hat und Knoten S_2 die Nachricht B kennt.

  In Fall ① ohne Netzwerkcodierung ist jeweils eine einzelne Übertragung für die Nachricht A und Nachricht B notwendig, da die Information über die bekannte Vorverteilung der Nachrichten nicht sinnvoll genutzt werden kann. In Fall ② reicht die Übertragung eines einzelnen Paketes gleichzeitig an S_1 und S_2 aus, um beide Senken mit den erforderlichen Nachrichten zu versorgen. Die Quelle Q kann hier die bekannte Vorverteilung der Nachrichten nutzen, um mittels einer effizienten Codierung die Anzahl der benötigten Übertragungen zu reduzieren.

  Um diese in der Theorie erzielbare Verbesserung der Energieeffizienz auch in praxistauglichen Protokollen nutzen zu können, ist neben der Verwendung eines optimierten und auf eine spezifische Topologie angepassten Codierungsschemas auch eine Anpassung des Medienzugriffsverfahren notwendig. Im Zusammenspiel können dabei die Sendezeitpunkte optimiert werden, so dass möglichst lange Zeiträume erreicht werden, in denen ein System in einem energiegünstigen Zustand verbringen kann (siehe Kapitel 3.6).

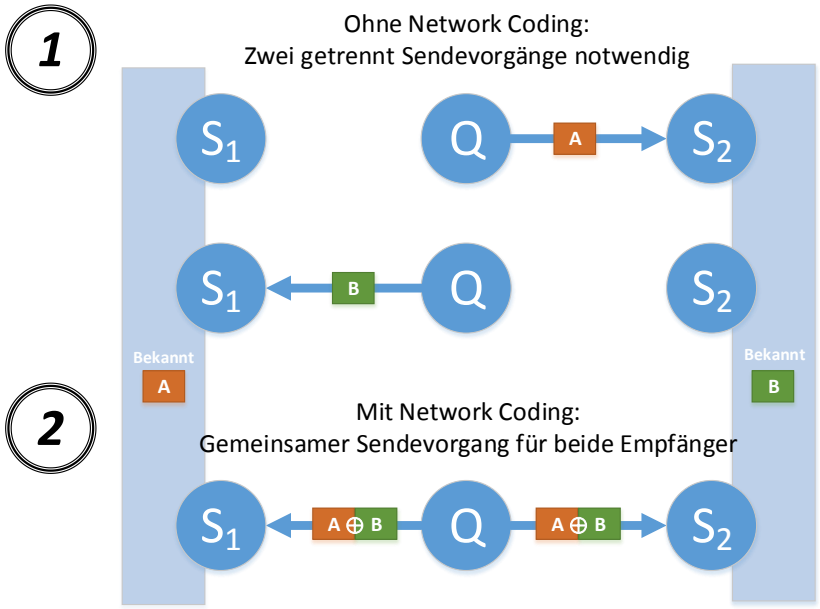

Abbildung 3.3 Reduktion der Sendevorgänge durch Netzwerkcodierung bei bekannter Datenvorverteilung

3.2 Theoretischer Hintergrund

Das *Network Coding Problem* beschreibt die Fragestellung, wie die Informationen der Quellen an alle m Senken unter Ausnutzung der vorhandenen Kapazität $\mathcal{K}$ des Netzwerkes möglichst effizient übertragen werden können.

Die Kapazität eines Netzwerks ist in der Praxis eine zeitabhängige Größe so dass das Network Coding Problem für anwendungsbezogene Problemstellungen umformuliert werden kann: Wie ist es möglichst effizient, also in möglichst kurzer Zeit und mit möglichst wenig Übertragungen realisierbar, dass die von den Senken benötigten Informationen ausgehend von den Datenquellen übermittelt werden?

Einen umfassenden Überblick über weitere theoretische Grundlagen der Netzwerkcodierung bietet [121].

Für eine praktische Realisierbarkeit von Netzwerkcodierung in realen drahtlosen Netzwerken sind die folgenden theoretischen Arbeiten von hoher Bedeutung, da sie die Grenzen der erzielbaren Effizienz für die zugrundeliegenden Kommunikationsmuster und Topologien vorgeben.

[26] präsentiert einen einfachen Einstieg in die theoretischen Grundlagen der Netzwerkcodierung. Insbesondere beschäftigt sich die Forschungsarbeit mit der Eignung von Netzwerkcodierung für verschiedene Netzwerk- und Anwendungsklassen. Über praktische Überlegungen die zur Vereinfachung des Decodierens führen, wie die Erzeugung einer Dreiecksmatrix, die Größe von Generationen (vgl. Abschnitt 3.3.4) oder Verringerung von Verzögerungen durch ein beschleunigtes Decodieren von unvollständigen Informationen wird ein kurzer Überblick gegeben.

Tabelle 3.1 Klassifikation der Optimierungs- und Lösungsmöglichkeiten von Multicast-Problemstellungen durch Netzwerkcodierung [62]

Benötigtes Codierungsschema	Anzahl Quellen	Anzahl Senken	Bereitstehende Informationen (Quellen)	Benötigte Informationen (Senken)
Triviale Codes reichen aus	1, n	1	I, D, B	I
oder keine Codierung nötig	1, n	m	I	D
Lineare Codes reichen aus	1, n	m	I, D, B	I
Auffinden linearer Codes NP-hart	1, n	m	I	B
oder nichtlineare Codes	n	m	D, B	D, B

In [62] werden die existierenden theoretischen Problemstellungen zur Optimierung des Informationsflusses anhand des Durchsatzes eines geeigneten Codierungsschemas klassifiziert (siehe Tabelle 3.1). Es wird dabei anhand der Anzahl der vorhandenen Informationsquellen n und Senken m im Netzwerk und der jeweils zur Verfügung stehenden und geforderten Informationen unterschieden.
Es werden die folgenden Fälle im Einzelnen unterschieden:

I Es stehen **identische** Nachrichten an den Quellen zur Verfügung bzw. die Senken verlangen sämtliche Nachrichten.

D Es stehen **disjunkte** Nachrichten an den Quellen zur Verfügung bzw. es werden disjunkte Nachrichtenmengen durch die Senken benötigt.

B Es werden keine Zusagen über die Verfügbarkeit der Nachrichten getroffen bzw. es werden keine besonderen Anforderungen durch die Quellen gestellt, so dass **beliebige** Nachrichten durch die Quellen akzeptiert werden.

Multi-Source-Multicast Network Coding Problem

Die in dieser Arbeit betrachtete Problemstellung des Multisource-Multisink-Multicast bzw. der Datenübertragung von mehreren Anwendungen mit n unabhängigen Quellen und m Senken ist nach [62] ein (n,m,D,D)-Problem, da die an den Quellen erzeugten Informationen individuell von einzelnen Quellen angefordert werden können, ohne dass eine Abhängigkeit der Informationen zwischen unterschiedlichen Quellen oder Senken vorausgesetzt werden kann. Das vorliegende Multisource-Multisink-Multicast-Problem ist daher nur durch ein nichtlineares Codierungsschema per Netzwerkcodierung optimal lösbar.

Ob eine gegebene Topologie eine Lösung durch lineares Network Coding besitzt ist nach [75] im Allgemeinen in polynomialer Zeit berechenbar. Anders ist die Situation für die Berechenbarkeit eines optimalen nichtlinearen Codierungsschemas. Die Berechnung der Lösung ist für den nichtlinearen Fall ein NP-hartes Problem. Zusätzlich steigt der Rechenaufwand des Problems in Abhängigkeit der Netzgröße. Sind die Topologie des Netzes und die einzelnen zu übertragenen Datenströme im Vorhinein nicht vollständig bekannt, so empfiehlt sich für eine praktische Lösung stattdessen Nutzung einer Heuristik auf Grundlage von XOR als Codierungsschema. Insbesondere ist noch kein algorithmischer Ansatz bekannt, um für mehrere überlagernde Multicast Datenübertragungen aus unterschiedlichen Quellen ein optimales Codierungsschema zu berechnen ist.

Multirate Network Coding Problem

Das *Multirate Network Coding Problem* beschreibt die Herausforderung, unterschiedliche Nachrichtenströme, die mit unterschiedlichen Datenraten übermittelt werden müssen durch Netzwerkcodierung zusammenzufassen.

Zudem kann eine Priorisierung von einzelnen Datenströmen gewünscht sein, so dass Daten eines bestimmten Datenstroms auch ohne Informationen eines weiteren Datenstroms decodierbar bleiben müssen.

Lösungen des Multirate Network Coding Problems werden in [93] und [53] vorgestellt. Werden unterschiedlich viele Nachrichten in einem codierten Paket kombiniert, so lässt sich die Priorisierung der Datenströme und der Aufwand zur Decodierung dynamisch anpassen. Dennoch stellt die Maximierung der Übertragungsrate für Multicast-Übertragungen in einem Multirate-Szenario ein NP-hartes Problem dar und sollte daher bevorzugt per Heuristik gelöst werden.

3.2.1 Begriffsdefinitionen

Die folgenden Begriffsdefinitionen mit Bezug zur Netzwerkcodierung werden im weiteren Verlauf der Arbeit ausgiebig genutzt und werden daher hier zunächst definiert.

Aus Perspektive der Codierungstheorie wird der Begriff *Quellblock* synonym zu den Begriffen des *Quell-* und *Informationswortes* verwendet. Im weiteren Kontext dieser Arbeit entspricht auch eine von einer Datenquelle zu übertragende Nachricht jeweils einem Quellblock und unterscheidet sich somit von einem Paket, das mehrere kombinierte Nachrichten als Nutzdaten enthalten kann. Diese werden mittels eines Paketes jedoch in codierter Form übertragen. Der entsprechende repräsentierende Begriff im Rahmen eines Codierungsschemas ist der *Ausgabeblock* und entspricht in der Codierungstheorie auch einem sogenannten *Codewort*. Die Möglichkeit, eine logisch zusammenhängende Nachricht auf einer höheren Abstraktionsebene, zum Beispiel durch eine Anwendung, in mehrere Datenblöcke zu fragmentieren, die dann logischen Quellblöcken entsprechen, bleibt somit unberührt.

Definition 3. ***Codierungsschema:*** *Das Codierungsschema vereinigt ein Codierungsverfahren (siehe Kapitel 3.3) und eine Abbildung der Codierungsoperationen auf die logischen Knoten eines Netzwerkes.*

Dazu gibt das Codierungsschema zum Beispiel vor, auf welchen Knoten welche Wahrscheinlichkeitsverteilungen oder Koeffizienten genutzt werden. Darüber hinaus legt das Codierungsschema auch die Zuteilung der Knotenrollen anhand der logischen Positionen im Netzwerk fest. Dazu muss jedem Knoten eine Menge an Eingabeknoten und Ausgabeknoten zugeteilt werden, sowie die Zugehörigkeiten des Knotens zu den Gruppen der codierenden und decodierenden Knoten. Um späteres Decodieren zu ermöglichen, muss ein Codierungsschema dafür Sorge tragen, dass die in einem Paket enthaltenen Nachrichten eindeutig identifiziert werden können. Kann die Identifizierung nicht implizit erfolgen, etwa durch eine feste Codierung, so muss die Identifikation der enthaltenen Nachrichten beispielsweise durch Zusatzinformationen im Paketkopf ermöglicht werden.

Definition 4. ***Codierungskomplexität:*** *Die Codierungskomplexität c dient als Maß für die benötigten Operationen die zum Codieren oder Decodieren eines codierten Blocks oder Paketes notwendig sind.*

Im Allgemeinen entspricht dies der Menge der kombinierten Quellblöcke bzw. Originalnachrichten in einem Ausgangsblock und lässt die Komplexität des verwendeten Codierungsverfahrens unberücksichtigt. Somit eignet sich die Codierungskomplexität, um die Menge der erforderlichen Resolutionsschritte anzugeben oder die Menge der beim Decodieren zu eliminierenden Koeffizienten bei *Zufälligen Linearen Codes* (siehe Kapitel 3.3.4).

Definition 5. ***Resolution:*** *Als Resolution wird das fortgesetzte Kombinieren von Paketen oder Ausgabeblöcken mit dem Ziel, die Codierungskomplexität schrittweise zu verringern, bezeichnet. Dabei werden identische Rechenoperationen wie bei den Codierungsschritten eingesetzt.*

Somit unterscheidet sich die Resolution von Codierungsverfahren, die beim Decodieren eine andere Vorgehensweise verlangen als beim Codieren, wie beispielsweise die Gauss-Elimination der Koeffizienten bei Zufälligen Linearen Codes.

Definition 6. ***Atomares Paket:*** *Atomare Pakete sind Pakete der Codierungskomplexität* $c = 1$.

Daraus kann abgeleitet werden, dass atomare Pakete genau eine Nachricht bzw. einen Quellblock enthalten und durch weiteres Decodieren keine weitere Reduktion der Codierungskomplexität möglich ist. Das Umwandeln des Paketes in eine Nachricht kann also unmittelbar erfolgen, ohne auf das Eintreffen weiterer Pakete warten zu müssen. Das Atomare Paket mit ggf. enthaltenem Koeffizienten bildet somit einen Basisvektor für codierte Pakete und ist linear unabhängig zu anderen atomaren Paketen.

Definition 7. ***Vollständige Decodierung:*** *Ein Paket oder Ausgabeblock ist genau dann vollständig decodiert, wenn alle enthaltenen Nachrichten bzw. Quellblöcke einzeln vorliegen. Werden mittels Netzwerkcodierung k Quellblöcke* $q_1, \ldots, q_k \in \mathcal{Q}$ *aus einer Menge von Quellen in l Ausgabeblöcke* $a_1, \ldots, a_l \in \mathcal{A}$ *codiert, so ist es einer Senke möglich diese*

Quellblöcke vollständig zu decodieren, sobald k linear unabhängige codierte Ausgabeblöcke $a_{l+1}, \ldots, a_{l+k}$ vorliegen. Die codierten Ausgabeblöcke müssen dazu aus $a_1, \ldots, a_l$ linear kombiniert worden sein oder identisch mit diesen sein.

Das Decodieren einzelner Quellblöcke kann bereits zuvor möglich sein, wenn die vorliegenden Ausgabeblöcke eine geeignete Form bzw. einzelne Ausgabeblöcke eine niedrige Codierungskomplexität aufweisen.

Innovative Pakete und Nachrichten

Ein Paket wird als Kombination von Nachrichten als *innovativ* für einen Knoten bezeichnet, wenn es aktiv zum Decodieren beitragen kann. Das bedeutet, dass ein Paket mit anderen Paketen oder Einzelnachrichten derart kombiniert werden kann, so dass die enthaltenen Nachrichten anschließend in nicht-codierter Form vorliegen. Dies kann je nach Codierungsschema beispielsweise durch Resolution oder Gauss-Elimination erfolgen.

Definition 8. ***Innovative Pakete:***
Ein Paket P_j ist als Kombination von Nachrichten dann innovativ*, wenn die Matrix aus Codierungsvektor die gleiche Nachrichten enthalten, durch Hinzunahme des Paketes einen höheren Rang hat, als die gleiche Matrix ohne den Vektor von P_j. Somit wird die Reihe j bei der Gaus-Elimination nicht auf Nullen reduziert [26]. Dabei haben Pakete ohne die Nachricht M_i den Codierungsvektor $C_i = 0$ an der Stelle $a_{i,j}$.*

Ob ein sofortiges erfolgreiches Decodieren auf dem Knoten möglich ist, hat für die Definition des Begriffs jedoch keine Bedeutung.

Definition 9. ***Innovative Nachrichten:*** *Eine einzelne Nachricht in einem Paket ist dann für einen Knoten* innovativ*, wenn sie nicht Bestandteil einer Nachrichtenkombination ist, aber zugleich essentiell für den Decodierungsvorgang ist, da ihr Codierungsvektor linear unabhängig zu dem Codierungsvektor von anderen Nachrichten ist.*

3.2.2 Netzwerkcodierung als Routingalternative

Da Netzwerkcodierung den Fluss eines kapazitätsbeschränkten Netzwerks mit mehreren Senken oder Quellen optimieren kann, bildet es eine Lösung für das allgemeine Routing-Problem. Routing stellt sich nur als Teilproblem der Flussoptimierung dar, da lediglich eine einzelne Nachricht von einer Quelle zu einer Senke übermittelt werden soll. Die Lösung des Routing-Problems präsentiert einen Pfad durch das Netzwerk, der die erforderliche Kapazität aufweist. Um den optimalen Durchsatz per Routing zu ermitteln werden NP-vollständige Verfahren benötigt. Zugleich lässt sich der maximale Durchsatz per Netzwerkcodierung bereits in polynomialer Zeit ermitteln [26, 116].

Hier wird nun ein kurzer Überblick über Forschungsansätze gegeben, die sich mit der Nutzung von Netzwerkcodierung an Stelle von klassischen Routingverfahren beschäftigt haben.

Die Grundlagen für Netzwerkcodierung als Routingalternative konnten in [38] und [39] gelegt werden. Die theoretischen Forschungsarbeiten zeigen, dass Netzwerkcodierung den Durchsatz von Routing übertreffen kann. Dazu wird die theorethische Schranke für die Decodierbarkeit mittels Zufälligen Linearen Codes siehe Abschnitt 3.3.4 in einem Multisource-Netzwerk mit Gittertopologie bewiesen. Es konnte der Zusammenhang der Wahrscheinlichkeit von Decodingfehlern bei zufälliger Wahl eines identischen Koeffizienten durch zwei Knoten und der Länge des Galois-Felds gezeigt werden. Zusätzlich wurde gezeigt, dass die Wahrscheinlichkeit, den optimalen Durchsatz zu erreichen, sich exponentiell 1 nähert, wenn die Bitbreite des Galois-Felds wächst.

Die Ergebnisse aus [116] zeigen, dass Netzwerkcodierung auch effizienter als probabilistisches Routing eingesetzt werden kann, wenn Daten an alle Knoten verteilt werden sollen. Die Forschungsarbeit untersucht explizit die Nutzbarkeit von Netzwerkcodierung als Routingalternative in drahtlosen Netzen unter extremen Einsatzbedingungen, wie beschränktem Speicher und niedriger Rechenleistung, Mobilität und dünn besiedelten Netzen und Sensornetzen. Kernpunkte der Forschungsarbeit sind die Betrachtung von Zustellraten und die Verwaltung von Generationsgrößen. Es konnte dabei festgestellt werden, dass die Mobilität von Knoten in Kommunikationsszenarien mit Netzwerkcodierung vorteilhaft genutzt werden kann und sich auf Sensornetzen auch mit kleinen Generationsgrößen schon Vorteile gegenüber probabilistischem Routing einstellen.

In [75] wird das Problem untersucht, dass Routing dazu tendiert, Eigenschaften der drahtlosen Kommunikation zu vernachlässigen wie z. B. Broadcasting, Interferenz, Fading und Mobilität der Knoten. Ein weiteres Problem existierender Routingansätze ist insbesondere die Betrachtungsweise von Verbindungen zwischen Knoten als reine Unicast-Verbindungen. Die Forschungsarbeit präsentiert Auswahlmechanismen zum Auffinden von geeigneten Subgraphen für die gemeinsame Codierung von Daten aus unterschiedlichen Quellen. Wird Netzwerkcodierung als Übermenge von Routing begriffen, so kann für eine vorgegebene Topologie ein optimales Codierungsschema ermittelt werden. In drahtlosen Netzwerken eignen sich stattdessen häufig Heuristiken besser zur Codierung, um die vorherrschende Netzwerkdynamik einzubeziehen. Im Zusammenspiel mit Medienzugriffsverfahren ist eine Heuristik für die Topologieauswahl und die Weiterleitung beispielsweise nötig, da die Anzahl der möglichen Subgraphen und möglichen Codierungsschemas exponentiell mit der Knotenanzahl und der möglichen Reichweiten der Knoten durch Regulierung der Sendeleistung steigt.

Der ROCX-Ansatz in [81] analysiert das Potential für Netzwerkcodierung bei Routing-Strategien, die während dem Routing das Codierungsschema berücksichtigen. Die Autoren führen die *ECX*-Metrik ein, um die Anzahl der benötigten Übertragungen für den Austausch eines Paket bei fehlerbehafteten drahtlosen Links zu ermitteln. Es kann mittels theoretischer Analyse auf gemessenen Daten eines Testbetts mit 38 Knoten, eine marginale Verringerung der benötigten Pakete um wenige Prozent gegenüber einem Routing auf dem kürzesten Pfad nachgewiesen werden. Ein verteilt einsetzbares Routing-Verfahren, das von diesen

opportunistischer Netzwerkcodierung profitieren könnte, kann im Rahmen des Ansatzes jedoch nicht entworfen werden.

Die Arbeiten [71] und [60] versuchen den Vorteil von Netzwerkcodierung gegenüber Routing hinsichtlich des erreichbaren Durchsatzes für unterschiedliche Routingtopologien zu ermitteln. Für ausgewählte Topologien mit einer starken Vernetzung der Knoten durch viele ungerichtete Einzelverbindungen konnte dabei gezeigt werden, dass Netzwerkcodierung einen bis zu Faktor 3 höheren Durchsatz erzielen kann als Routing über einzelne Unicast-Verbindungen.

3.3 Codierungsverfahren und Aufwand

Im diesem Abschnitt werden existierende Codierungsverfahren auf ihre Nutzbarkeit für Netzwerkcodierung in drahtlosen Sensornetzen hin untersucht. Neben dem Aufwand werden die praktischen Grenzen der Codierungsverfahren diskutiert.

Tabelle 3.2 gibt eine Übersicht über die in diesem Abschnitt diskutierten Codierungsverfahren, ihre Eigenschaften, sowie den benötigten Rechenaufwand für die Netzwerkcodierung. Dabei gibt n die Anzahl der verwendeten Nachrichten zur Codierung und Decodierung an. Im Anschluss werden nun die typischen Eigenschaften der Codes im Details erläutert.

Tabelle 3.2 Klassifikation der geeigneten Codierungsverfahren zur Netzwerkcodierung

Codierungsverfahren	Ratenlos	Systematisch	Linear	Zufällig	Komplexität Codierung	Komplexität Decodierung
Exklusives Oder	-	+	+	①	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$
Luby-Transform Codes	+	②	+	+	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$
Zufällige Lineare Codes	+	②	+	+	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
Raptor Codes	-	③	(+)	(+)	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Anmerkung 1. ***zu Tabelle 3.2***
① *Sowohl zufällige als auch feste Codierung möglich*
② *Systematische Codes möglich, aber nicht erforderlich*
③ *Nur wenn innerer Code kein LT-Code ist*

3.3.1 Schematische Einordnung

In Hinblick auf geeignete Codierungsverfahren für die Netzwerkcodierung sind die folgenden drei Eigenschaften eines Codes aus der Codierungstheorie von besonderer Bedeutung (vgl. [40]):

Definition 10. ***Ratenlose Codes:*** *Ratenlose Codes haben die Eigenschaft, dass aus einer beschränkten Menge von Quellblöcken eine unbeschränkte Menge an Ausgabeblöcken codiert werden können.*

Ohne eine feste Coderate kann der Grad der Redundanz bei der Codierung angepasst werden. Das ist dann vorteilhaft, wenn die Codierung an zu erwartende Übertragungsverluste auf bestimmten Verbindungen oder die vorherrschende Kanalqualität des Mediums angepasst werden soll. Daher eignen sich ratenlose Codes auch in drahtlosen Sensornetzen, um den Mehraufwand von ARQ-Verfahren mittels zusätzlicher Quittungen bei stark fehlerbehafteten Übertragungssituationen zu reduzieren [40]. Ein weiterer Vorteil ist, dass zusätzliche Ausgabeblöcke jederzeit generiert werden können, etwa falls Paketverluste auftreten oder wenn der Empfänger signalisiert, dass ein Decodieren mit den bisher erhaltenen Informationen noch nicht möglich ist.

Definition 11. ***Systematische Codes:*** *Systematische Codes codieren die Quellblöcke derart, dass jedem Quellblock ein Ausgangsblock zugeordnet wird, der den Inhalt des Quellblocks enthält und lediglich um Prüfdaten ergänzt.*

Somit können die enthaltenen Informationen in einem Codierungsschema welches Systematische Codes verwendet oft besonders leicht decodiert werden.

Definition 12. ***Lineare Codes:*** *Ein Code ist dann ein linearer Code, wenn die Kombinierbarkeit von zwei Ausgabeblöcken zu einem weiteren gültigen Ausgabeblock gegeben ist. Dazu wird die Modulo-x Summe aus den Ausgabeblöcken gebildet und überprüft ob das Ergebnis in der Menge der Ausgabeblöcke $\mathcal{A}$ enthalten ist. x entspricht dabei der Größe des Ausgabealphabetes, so dass bei binären Codes Exklusives Oder genutzt wird. Es gilt also:*

$$a \in \mathcal{A}, b \in \mathcal{A} \Rightarrow a \oplus b \in \mathcal{A}$$

Insbesondere bei Codierungsverfahren mit Ausgabeblöcken fester Länge ist lineare Codierung eine gewünschte Eigenschaft, da beim Kombinieren von codierten Informationen keine längeren Blocks entstehen, die zusätzliche Bits zur Codierung benötigen. Bei der Übertragung von Paketen kann bei linearen Codes also auch die Paketgröße gezielt kontrolliert werden. Bei der Netzwerkcodierung auf Zwischenknoten fallen für die Weiterleitung also nicht mehr Daten an, als zuvor empfangen wurden. Die Eingangsdaten werden hier linear zu Ausgangsdaten kombiniert.

Definition 13. ***Zufällige und feste Codierung:*** *Von* zufälliger Codierung *wird im Gegensatz zu* fester Codierung *dann gesprochen, wenn die Quellblöcke mittels Zufallskombinationen, einer Zufallsverteilung oder zufällig gewählten Koeffizienten zu Ausgabeblöcken kombiniert werden.*

Insbesondere ratenlosen Codes liegt häufig eine zufällige Codierung zu Grunde. Ändern sich Übertragungsfehlerwahrscheinlichkeiten zwischen Sendern und Empfängern, so können mit zufälliger Codierung einfach weitere Pakete codiert werden. Dabei ist jedoch neben den eigentlichen Nutzinformationen in codierter

Form auch eine Übertragung von Metainformationen erforderlich. In Abhängigkeit vom gewählten Codierungsverfahren werden in den Metainformationen die Parameter angegeben, die benötigt werden um die jeweils generierte Codierung des Ausgabeblocks zu identifizieren.

Im Gegensatz dazu kann eine feste Codierung sehr effizient realisiert werden, da Speicherbedarf und bei Codierung induzierte Verzögerungen bereits zum Entwurfszeitpunkt feststehen. Es müssen jedoch auch bereits Annahmen zur Häufigkeit der Übertragungsfehler vorliegen, wenn auf zusätzliche Redundanz als Spielraum für zuverlässige Übertragungen weitgehend verzichtet werden soll.

Die Berechnung der Ausgabeblöcke findet in den folgenden Codierungsverfahren im Allgemeinen unter Zuhilfenahme eines Galois-Feldes statt. Aus Effizienzgründen sind Galois-Felder der Form $GF(2^n)$ auf Rechnersystemen vorzuziehen. Ein Galois-Feld der Form $GF(2^n)$ ist dabei ein endlicher Körper mit 2^n unterscheidbaren Elementen und $n \in \mathbb{N}$. Für Rechenoperationen in $GF(2^n)$ stehen Addition und Multiplikation in angepasster Form zur Verfügung. Für eine effiziente Decodierung werden die benötigten Operationen üblicherweise in Tabellenform im Speicher vorgehalten. Während die wachsende Anzahl unterschiedlicher Codierungssymbole mit zunehmender Feldgröße eine effizientere Codierung der Ausgabeblöcke und in Folge auch einen höheren Durchsatz erlaubt, wächst zugleich auch der benötigte Speicherplatz für die Bereitstellung der Operationstabellen. Gängige Galois-Felder für Codierungsverfahren auf mobilen und eingebetteten Rechnersystemen überschreiten daher selten Größen von $GF(2^8)$. Neben dem relativ langsamen Nachschlagen in Speichertabellen werden aber bereits parallele und hardwarebeschleunigte Optimierungen diskutiert, wie z. B. in [29] und [17].

3.3.2 Exklusives Oder

Exklusives Oder (XOR) wird häufig nicht als eigenständiges Codierungsverfahren betrachtet, da es als Operation in komplexeren Verfahren Verwendung findet. Dennoch entspricht XOR der Addition bei Verwendung des kleinsten nutzbaren Galois-Feldes $GF(2)$ und kann daher für die Codierung kleiner Datenmengen bereits zweckmäßig eingesetzt werden. Für eine Feldgröße von 2 ist es zugleich das effizienteste bekannte Verfahren [112].

Die Quellblöcke können mittels Exklusivem Oder logisch mit Nullböcken zu Ausgabeblöcken kombiniert werden. Dann kann Exklusives Oder auch als systematischer linearer Code eingesetzt werden, wie bereits in Abbildung 3.2 zu sehen war. Somit können codierte Daten auch ohne vorheriges Decodieren durch erneutes XOR gebildet werden. Dies kann für eine schnelle Verarbeitung auf Zwischenknoten eingesetzt werden, da die schnelle Weiterleitung kombinierter Daten ohne vorheriges Decodieren stets gültige Informationen des Codierungsschemas enthält.

Vorteilhaft ist außerdem, dass kein zusätzlicher Rechenaufwand für zur Decodierung notwendige Umkehroperationen anfällt. Exklusives Oder ist auf gängigen Mikrocontroller-Hardwareplattformen bereits Kernbestandteil des Instruktionssatzes und direkt in der Hardwarearchitektur realisiert. Daher lässt sich XOR effizient

für die verfügbaren Datentypen berechnen und der sonst erforderliche Speicher für Operationstabellen kann eingespart werden.

Beispielsweise bieten 8-Bit Atmega Mikrocontroller eine *EOR*-Operation, um zwei in Registern gespeicherte 32-Bit-Werte innerhalb eines Rechenzyklus mittel Exklusiven Oders zu verknüpfen [10].

Das schrittweise und rechenaufwändige Auflösen eines Linearen Gleichungssystems beim Decodieren, wie es in anderen Codierungsverfahren notwendig ist, entfällt bei Exklusivem Oder. Stattdessen müssen zur Resolution jeweils kombinierbare Ausgabeblöcke gefunden werden, die sich durch erneute Kombination zu Quellblöcken auflösen lassen.

3.3.3 Ratenlose Fountain-Codes

Ratenlose Fountain-Codes gehören zur Klasse der binären Auslöschungscodes (auch Erasure Codes), genauer zur Klasse der irregulären LDPC-Codes (Low Density Parity Check Codes) [40]. Als Verallgemeinerung der XOR-Codierung werden diese Codes typischerweise auch zur Vorwärtsfehlerkorrektur bei Ende-zu-Ende-Kommunikation eingesetzt. In Verbindung mit Zwischenknoten, die selber Kombinationen aus empfangenen Paketen bilden, lassen sich aber auch Hop-zu-Hop ARQ-Mechanismen bei gleichem Durchsatz realisieren. Die Anzahl erzeugter redundanter Ausgabeblöcke kann bei ratenlosen Fountain-Codes mittels Wahl der Verteilung Ω an die Wahrscheinlichkeit von Übertragungsfehlern angepasst werden.

Die bekannteste Realisierung von ratenlosen Fountain-Codes sind *Luby-Transform-Codes (LT)*. Zur Codierung wird bei LT-Codes ein Ausgabeblock jeweils aus einer Reihe von Quellblöcken erzeugt, die per XOR kombiniert werden. Dazu werden jeweils k Quellblöcke zufällig ermittelt. Die Anzahl k der zu kombinierenden Ausgabeblöcke gibt eine Zufallsverteilung Ω getrennt für jeden einzelnen Quellblock vor. Die Anzahl k der für die codierte Ausgabe verwendeten Quellblöcke beschreibt hier die Codierungskomplexität.

Zum Decodieren muss die Kombination der Quellblöcke jedoch dokumentiert oder durch einen identischen Pseudozufallsgenerator erneut erzeugt werden können. Während für das Codieren mit LT-Codes linearer Aufwand nötig ist, erfolgt das Decodieren in $\mathcal{O}(n \log n)$ [14]. Dazu werden empfangene Pakete in einem Puffer anhand ihrer Codierungskomplexität miteinander verglichen. Pakete mit geringer Codierungskomplexität werden bevorzugt durch Resolution aufgelöst. Die decodierten Blöcke können dann iterativ zum Auflösen komplexer Pakete verwendet werden.

Beispielhafte Arbeiten, die eine LT-Codierung in drahtlosen Sensornetzen realisieren sind die Arbeiten [7] und [13] zur verteilten Datenspeicherung.

3.3.4 Zufällige Lineare Codes

In [38] führen die Autoren *zufällige lineare Codes* (Random Linear Coding, RLC) zur Verwendung mit der Netzwerkcodierung ein. Dabei handelt es sich bei Zufälligen

Linearen Codes um eine Variante der ratenlosen Fountain-Codes. Im Unterschied zum bereits vorgestellten zufälligen Wählen der Quellblöcke werden hier Koeffizienten aus dem verwendeten Galois-Feld für die einzelnen Blöcke zufällig gewählt. Für die Quellblöcke $q_i \in \mathcal{Q}$ mit $i = 1 \ldots k$ und Koeffizienten $\beta_{i,j} \in GF(2^n)$ wird anschließend eine Linearkombination über dem Galois-Feld mit folgender Form gebildet, um den Ausgabeblock $a_j \in \mathcal{A}$ zu erstellen:

$$a_j = \sum_{i=1}^{k} \beta_{i,j} q_j$$

Bei dieser Art der Codierung werden üblicherweise neben dem Ausgabeblock auch die Liste der verwendeten Koeffizienten als Vektor übertragen. So kann die Kombination zweier codierter Pakete, gemäß der Eigenschaft linearer Codes, durch simple Addition der Vektoren und Ausgabeblöcke zu einem neuen gültigen codierten Paket erfolgen.

Zum Decodieren ist die Lösung eines linearen Gleichungssystems auf Empfängerseite notwendig. Dazu werden die empfangenen Koeffizientenvektoren mit den Ausgabeblöcken als Zeile in eine Decodingmatrix aufgenommen. Im Anschluss kann der Rang der Matrix überprüft werden, um herauszufinden, ob ausreichend viele linear unabhängige Pakete gesammelt werden konnten, um die Daten vollständig zu Decodieren. Das eigentliche Decodieren erfolgt beispielsweise per Gauss-Elimination im Galois-Feld mit einer typischen Laufzeit- und Speicherkomplexität von $\mathcal{O}(n^3)$ (vgl. [21]). In [17] finden sich darüber hinaus Optimierungen durch Parallelisierung für geeignete Hardwareplattformen. Weitere Optimierungen durch Nachschlagen in vorberechneten Tabellen und mit Hilfe des Generatorpolynoms des Galois-Felds sind für kleine Größen ($n \leq 8$) möglich [26, 116]. Es wir aber zusätzlicher Speicher von 510 Bytes benötigt.

Um die Größe der Decodingmatrix zu reduzieren, wird häufig mit sogenannten *Generationen* gearbeitet. Daten die von der Menge der Quellknoten in einem Durchgang gemeinsam codiert wurden, stellen dabei eine Generation dar. Auf ein Codierungsschema, das fortlaufend neue Daten codieren und decodieren kann, wird dabei verzichtet.

Der große Vorteil von zufälligen linearen Codes liegt in ihrer dezentralen Handhabbarkeit und ihrer Unabhängigkeit von Kenntnissen der tatsächlichen Netzwerktopologie bei Netzwerkcodierung.

Neben dem hohen Speicher- und Decodierungsaufwand bestehen bei zufälligen linearen Codes aber weitere Nachteile. So kann etwa der Fall eintreten, dass mehrere Knoten die gleichen zufälligen Koeffizienten wählen oder linear abhängige Kombinationen der Koeffizienten wählen. In diesem Fall sind die erzeugten Ausgabeblöcke nicht innovativ und tragen nicht zum Decodieren bei. Somit leidet hier voranging die Effizienz. Die Wahrscheinlichkeit, dass solche Koeffizienten gewählt werden, sinkt mit zunehmender Größe des Galois-Feld. [75] empfiehlt die Größe des Galois-Feldes mindestens entsprechend der Anzahl der decodierenden

Empfänger zu wählen. In Simulationen konnte aber gezeigt werden, dass ab Größen von $n = 8$ die Wahrscheinlichkeit für das Erzeugen von linear abhängigen Ausgabeblöcke praktisch vernachlässigbar ist [26].

Um den Mehraufwand durch das Mitsenden des Codierungsvektors mit den verwendeten Koeffizienten einzusparen, können die Koeffizienten der Kodierenden Knoten auch vorverteilt oder fest vergeben werden. Während so zahlreiche Optimierungen bei Codierung, Decodierung und Speicherbedarf möglich sind, ist die feste Codierung nicht länger ratenlos und kann weder adaptiv zur Anpassung an Übertragungsfehler, noch zur effizienten Verteilung in unbekannten Topologien verwendet werden.

3.3.5 Rapid Tornardo- (Raptor-) Codes

Rapid Tornado- (auch *Raptor-*) Codes [94] stellen die heute effizienteste bekannte Realisierung von Fountain Codes dar. Raptor-Codes basieren dabei grundlegend auf einer Fortentwicklung von LT-Codes um ein zwei- oder mehrstufiges Verfahren zur Vorcodierung.

Dabei dient typischer Weise ein systematischer *Tornardo-Code* mit fester Rate in den ersten Stufen zur Vorcodierung. Für den Übergang zwischen den einzelnen Codierungsstufen wird eine Verteilung Ω in $GF(2^n)$ benötigt. Mittels der Verteilung wird zur Codierung festlegt, wie viele zufällig gewählte codierte Blocks als Ausgänge der Vorgängerstufe jeweils in einen codierten Block der Nachfolgestufe per XOR kombiniert werden. Die Vorstufen bilden somit einen irregulären Graphen. Als innerer Code findet dabei üblicherweise ein Luby-Transform-, Reed-Solomon- oder LDPC-Code Verwendung [40].

Raptor-Codes ermöglichen sowohl das Codieren als auch das Decodieren mit linearem Aufwand $\mathcal{O}(n)$. Insbesondere in Kombination mit festen Verteilungen lassen sich dabei in Hinblick auf Speicher und Verzögerung effiziente Codierungsschemata realisieren [27]. Da Raptor-Codes zu den Verfahren mit fester Codierungsrate zählen, können bei Multi- und Broadcastnetzen aber Probleme auftreten, wenn aufgrund von Kanalfehlern nicht ausreichend Daten zum Decodieren der Quelldaten beim Empfänger vorliegen.

Raptor-Codes sind aktuell ein populäres Forschungsgebiet. Zugleich stehen bereits eine Vielzahl von gebräuchlichen Raptor-Codes unter Patentschutz.

Neben Raptor-Codes als Bestandteil eines Netzwerkcodierungs-Schemas [101], finden sie in drahtlosen, ressourcenbeschränkten Netzwerken, beispielsweise Verwendung als Ersatz für ARQ-Verfahren, zwischen direkten Kommunikationspartnern [1] oder zur Verteilten Speicherung von Sensordaten aus unterschiedlichen Quellen [6].

3.3.6 Triangular Network Coding

Bei *Triangular Network Coding* (TNC) [83, 88] steht im Vordergrund, die Rechen- und Speicherkomplexität des Codierens und Decodierens von linearen Codes zu senken. Dies wird erreicht, indem nur bestimmte Ausgabeblöcke erzeugt werden,

die in der Decodingmatrix in einer Dreiecksform dargestellt werden können. Dies kann etwa durch schrittweises Verschieben per Shift-Operation und Ergänzen von redundanten Nullen erreicht werden. Durch diese Optimierungen lässt sich das Codieren auf XOR-Operationen zurückführen und die Komplexität des Decodierens auf $\mathcal{O}(n^2)$ reduzieren. Durchsatz und Coderate ändern sich dabei nicht. Allerdings profitieren von Triangular Network Coding lediglich Codierungsschemata, bei denen eine Quelle mit einer Menge von Senken kommuniziert. Für die Koordination zusätzlicher Quellknoten ist hier erheblicher zusätzlicher Aufwand nötig, der keinen Vorteil gegenüber linearer Netzwerkcodierung erwarten lässt.

3.4 Stand der Technik

Die folgenden Forschungsansätze zeigen, wie sich die Netzwerkcodierung praktisch in drahtlosen Netzwerken einsetzen lässt und welche Hemmnisse und Schwierigkeiten dabei zu überwinden sind. Einen Schwerpunkt der diskutierten Forschungsarbeiten bilden Arbeiten, die die Nutzung von drahtlosen Sensornetzen und ähnlichen leistungs- und speicherbeschränkten Systemen behandeln.

Die Forschungsarbeit [108] beschäftigt sich mit der Frage, in welchen Fällen Netzwerkcodierung im Bereich der drahtlosen Sensornetze überhaupt vorteilhaft ist.

Die Fähigkeit der Netzwerkcodierung, die Zuverlässigkeit im Fall von Knotenausfällen zu erhöhen, wird in [111] beschrieben. Darüber hinaus beschäftigt sich die Arbeit mit der Reduzierung der Redundanz transportierter Nutzdaten und Übertragungswiederholungen, während zugleich auch die Anzahl der Empfangsbestätigungen reduziert wird. Dennoch sind die Einsatzmöglichkeiten für die Netzwerkcodierung in drahtlosen Sensornetzen insbesondere durch die geringen Mengen verfügbaren Speichers beschränkt. Die Beschränkung wirkt sich vor allem beim Decodieren empfangener Daten aus, da nur eine kleine Menge an empfangenen Paketen zwischengespeichert werden kann.

Der Zusammenhang zwischen Netzwerkcodierung und Energieeffizienz wird in [112] in ersten Ansätzen untersucht. Die Arbeit konzentriert sich dabei auf die Betrachtung eines einzelnen Nachrichtenstrom bei vorgegebener Topologie und schlägt eine Codierungsschema vor, dass die Topologie in ein Netzwerkmodell mit fünf Schichten unterteilt. Die Unterteilung ermöglicht es, zwischen Quelle und Senken auf unterschiedlichen Pfaden mehrere Datenraten zu realisieren und dabei die Anzahl der Knoten zu reduzieren, die Berechnungen zur Codierung im Netzwerk vornehmen müssen. Die Ergebnisse sind jedoch nicht auf Kommunikationsszenarien übertragbar, wo mehrere überlappende Übertragungen oder Datenströme im gleichen Netzwerk aktiv sind und ihr Verhalten gegenseitig beeinflussen können oder die Topologie dynamisch ist.

COPE

Eine grundlegende Arbeit für die praktische Nutzung von Netzwerkcodierung in drahtlosen Netzen bildet die *COPE*-Architektur [49]. Dort wird erstmals die erreichbare Verbesserung des Durchsatzes in einem echten drahtlosen Netzwerk mit 20 Knoten untersucht.

Das zugrunde liegende Kommunikationsszenario von COPE bildet eine Unicast-Topologie mit vorgegebener Routing-Topologie.

COPE bildet eine Zwischenschicht zwischen einer IP-basierten Netzwerkschicht und der MAC-Schicht. Daher wird die Netzwerkcodierung nicht als Routing-Alternative verwendet. Vielmehr versucht COPE Codierungsmöglichkeiten bei der Weiterleitung von Paketen zu erkennen, indem statt regulärer Pakete codierte Pakete an mehrere Knoten des nächsten Hop geschickt werden, die jeweils unterschiedliche gewünschte Informationen daraus decodieren können. Diese Abhängigkeit von der unterliegenden Netzwerktopologie schränkt bei COPE jedoch die Anzahl der Codierungsmöglichkeiten deutlich ein und führt dazu, dass die erreichbaren Verbesserungen des Durchsatzes ebenfalls stark topologieabhängig sind und zwischen 5% und 70% schwanken. Darüber hinaus beschränkt COPE die Analyse der Codierungsmöglichkeiten auf eine 2-Hop-Nachbarschaft.

Um Informationen über die empfangenen Pakete zwischen benachbarten Knoten auszutauschen, nutzt COPE den Broadcast von sogenannten *reception reports*. Durch ständiges Mithören der Knoten auf dem gemeinsamen drahtlosen Medium können Knoten vor dem Weiterleiten von neuen Informationen den Kenntnisstand der Knoten für den nächsten Hop ermitteln. Anhand des Kenntnisstands lässt sich feststellen, welche Informationen in ein gemeinsames Paket codiert werden dürfen, um von mehreren Knoten des nächsten Hops erfolgreich decodiert werden zu können. Jeder Knoten kennt somit die Informationen, die den Nachbarknoten vorliegen.

Zur Codierung sieht COPE virtuelle FIFO-Warteschlangen vor, um die Informationen entsprechend ihrer Ankunftsreihenfolge zu codieren. Zugleich wird versucht, Informationen unterschiedlicher Länge in ein gemeinsames Paket zu codieren. Dies dient dazu, späteres Decodieren weiter zu vereinfachen.

In Stausituationen konnte der Durchsatz mit COPE um Faktor 3-4 gegenüber einem Datentransport mit UDP-Datenströmen gesteigert werden. Gegenüber einer klassischen Weiterleitung per Store-and-Forward-Strategie (siehe Kapitel 5.5) sind die Anforderungen an den Speicherbedarf bei COPE aber leicht erhöht, da die weitergeleiteten Pakete für weitere Decodierungsvorgänge und Fehlerbehandlung zwischen Nachbarknoten länger gespeichert werden müssen.

COPE behandelt zwar nicht explizit die Kommunikation in drahtlosen Sensornetzen, die folgenden Beiträge können aber ohne Einschränkungen auch in drahtlosen Sensornetzen vorteilhaft genutzt werden:

- mischen und zusammensetzen der Paketinhalte auf Zwischenknoten
- Codierung und Decodierung ist nicht auf Endsysteme beschränkt, sondern kann auf Zwischenknoten im Netzwerk erfolgen
- Ausnutzung der Broadcast-Eigenschaften des drahtlosen Kommunikationsmediums

- Ermittlung des Zustands der Nachbarschaft aus Reception Reports und Anpassung der eigenen Codierung
- einfache XOR-Codierung von Informationen ähnlicher oder identischer Größe
- Vermeidung von Problemen bei der Decodierung durch Vermeidung der Weiterleitung von mehreren Informationen für den gleichen nächsten Hop

Weitere Verbesserungen zu COPE hinsichtlich des erreichbaren Durchsatzes finden sich in der Forschungsarbeiten [122]. Dort wird das Zusammenspiel von COPE mit dem Medienzugriff betrachtet und die Nutzung von weiteren virtuellen Warteschlangen empfohlen, um codierte Pakete beim Senden gegenüber Informationen in nicht-codierten Paketen zu priorisieren.

Auch [61] nimmt Verbesserungen an COPE vor, indem die Erkennung von Pfaden zwischen Quelle und Senke optimiert wird. Bei *Distributed Coding Aware Routing* (DCAR) wird die Warteschlangenlänge als ergänzende Routingmetrik verwendet. Die Erkennung von Codierungsmöglichkeiten beschränkt sich bei DCAR zudem nicht länger auf die zwei nachfolgenden Hops. Bei einem Pfad wird in DCAR zunächst bewertet, ob er Codierungsmöglichkeiten aufweisen könnte oder nicht. Dies schränkt Pfadwahl auf vielversprechendere Kandidaten ein und kann den Durchsatz von COPE weiter verbessern. Dazu wurden Simulationen mit dem *ns-2*-Simulator durchgeführt. Als problematisch haben sich aber Kommunikationsszenarien herausgestellt, bei denen sich mehrere codierte Datenflüsse auf einem Knoten überlagern.

AdapCode

Das in [44] eingeführte Protokoll *AdapCode* dient vorrangig zur effizienten Verteilung von Anwendungscode-Updates in einem Sensornetz. Dazu wird Netzwerkcodierung mit einem Codierungsschema eingesetzt, das sich an die aktuelle Verbindungsqualität zwischen den Knoten anpasst, um den Durchsatz zu optimieren. AdapCode ist zugleich die erste Anwendung, die Netzwerkcodierung in vorteilhafter Weise in einem Sensornetz nutzen konnte. Auch auf die Beschränkung des verfügbaren Speichers in Sensornetzen nimmt AdapCode entsprechend Rücksicht.

Durch die Gruppierung der zu übertragenden Daten in Datenrahmen können durch AdapCode negative Empfangsbestätigungsbestätigungen (NACKs) eingesetzt werden, um benachbarten Knoten fehlende Daten eines Rahmens anzuzeigen und zuverlässige Übertragungen zu gewährleisten. Genau wie bei Datenströmen ist somit die Fehlerbehebung fehlender Daten durch Nachbarn möglich. Die Kehrseite ist der benötigte Speicherbedarf von 1500 Bytes pro Nachrichtenquelle, damit die Daten eines Datenrahmens in einer Matrix zur Decodierung mittels Gauss-Elimination gesammelt werden können. Zur Einführung der Netzwerkcodierung werden pro Paket 3 zusätzliche Bytes im Paketkopf benötigt, um die Codierungsvektoren der codierten Nachrichten zu kennzeichnen. AdapCode kann

im Versuch bis zu 40% der Pakete einsparen, die ohne Netzwerkcodieren zur Verteilung nötig wären.

In [95] wird ein Neighborhood-Discovery-Protokoll als Ergänzung zu AdapCode entworfen, das mehr Codierungsmöglichkeiten schaffen und somit weniger Pakete zur Übertragung erfordern soll.

CodeDrip

Ein weiteres Protokoll zur Datenverteilung mittels Netzwerkcodierung ist *CodeDrip* [21]. Anders als AdapCode konzentriert sich CodeDrip nicht auf die Verteilung von größeren Datenmengen in einem Datenrahmen, sondern auf die Verteilung von kleineren Datenmengen in Einzelnachrichten. Diese können beispielsweise eine Abfrage eines Sensors repräsentieren oder zur Konfiguration des Netzwerks dienen. Dazu kombiniert CodeDrip den Trickle Timer von DIP mit dem Suppression-Verfahren von Drip zur Reduktion von Duplikaten, die mehrfach aus der Nachbarschaft empfangen wurden und einem XOR-Schema zur Codierung der Nachrichten.

CodeDrip kann gegenüber der Verteilungsprotokolle DIP und Drip (siehe Kapitel 4.3.1) Verbesserungen hinsichtlich der Anzahl benötigter Pakete, der erreichten Informationsabdeckung (vgl.Kapitel 4.2) und der Paketverlustrate erzielen.

Der Ansatz in [111] leitet ein Codierungsschema ab, bei dem die Knoten zwischen einer einzelnen Quelle und mehreren Senkenknoten in mehrere Schichten unterteilt werden. Die Forschungsarbeit zielt dabei auf eine Verbesserung der Energieeffizienz in drahtlosen Sensornetzen durch ein Codierungsschema, das Daten über eine möglichst kleine Anzahl von beteiligten Verbindungen schickt. Dazu wird durch den Quellknoten der Max-Flow zu den Empfängern berechnet. Der Ansatz ist jedoch nur für Einzelquellen nutzbar und kann nicht mit dynamischen Topologieänderungen und Multi-Source Multicast-Netzen umgehen. Das Zusammenspiel zwischen Netzwerkcodierung und dem MAC-Layer hinsichtlich der Energieeffizienz wurde hier ignoriert, hat aber einen großen Einfluss, wie in [15] zu sehen.

3.5 Kernproblematiken beim Einsatz in drahtlosen Sensornetzen

In Hinblick auf effiziente Netzwerkcodierung ist in realen drahtlosen Netzwerken insbesondere die meistens fehlende Schleifenfreiheit der Topologie schwierig. Obwohl dieses Kriterium Netzwerkcodierung nicht generell verhindert, ist bei fehlenden Maßnahmen zur zielgerichteten Weiterleitung mit Effizienzeinbußen zu rechnen. Darüber hinaus muss ausgeschlossen werden, dass ein Knoten durch eine Schleife Daten zur Codierung als Quellblock verwendet, die vorher durch diesen Knoten als Ausgabeblock erzeugt wurden. Ansonsten ist eine Mehrfachcodierung möglich, die je nach Codierungsschema bis hin zu einem logischen Datenverlust führen kann.

3.5.1 Optimale Codierung

Für eine Auswahlentscheidung für möglichst ein effizientes Codierungsschema für Netzwerkcodierung in drahtlosen Sensornetzen muss die unterliegende Routing-Topologie dazu im voraus bekannt sein. Damit diese Informationen genutzt werden können, müssen üblicher Weise zusätzliche Nachrichten mit Routing-Informationen und zur Etablierung von Pfaden ausgetauscht werden.

3.5.2 Einsatz auf speicherbeschränkten Systemen

Um zu ermitteln, welche Generationsgrößen in Netzen mit beschränktem Speicherkapazitäten realisierbar sind, experimentiert [116] mit Generationsgrößen zwischen 4 und 32 für Codierungsvektoren in $GF(2^8)$ auf MICA2 Sensorknoten. Der entsprechende Speicherbedarf der Decodierungsmatrix liegt zwischen 516 und 4128 Bytes pro Generation. Zusätzlich werden für zwei Codierungstabellen mit vorberechneten Rechenoperationen für das Galois-Feld benötigt, die jeweils 255 Byte Speicher benötigen. Eine Abschätzung der Gesamtgröße ergibt einen Speicherbedarf im Bereich von 3 und 42 Kilobyte Speicheraufwand pro Knoten, so dass von hohen Generationsgrößen auf drahtlosen Sensorknoten in der Regel abgesehen werden muss.

[123] präsentiert eine Strategie zur Auswahl von Paketen, um entscheiden zu können, welche Pakete Decoding-Möglichkeiten bieten und welche Pakete verworfen werden können, um knappen Speicherplatz zu sparen. Leider konzentrieren sich diese Arbeiten auf klassische Unicast-Netzwerke und setzen voraus, dass ein Knoten seine 2-Hop Nachbarschaft kennt. Darüber hinaus benötigt dieser Ansatz Knoten, die in der Lage sind, Routinginformationen und den Zustand ihres Zwischenspeichers zu teilen. Pakete beinhalten jeweils maximal 2 per XOR codierte Originalnachrichten.

3.6 Vereinbarkeit mit energiebedarfsoptimierten Medienzugriffsverfahren

Energieeffiziente Netzwerkcodierung in drahtlosen Netzwerken erfordert ein Medienzugriffsverfahren, dass ein Mithören der Nachrichten fremder Knoten in Reichweite ermöglicht. Zeitgleich sollte das Medienzugriffsverfahren die Nutzung eines periodischen Zyklus aus Aktivitätsphasen und Inaktivitätsphasen mit geringem Energiebedarf (sogenanntes *Duty-Cycling*) unterstützen. Mithören kann prinzipiell von jedem Medienzugriffsverfahren eingesetzt werden, das Broadcasting nutzt. Dazu muss ein Knoten nur ausreichend lange aktiv bleiben und unabhängig von der Adressierung alle Nachrichten auf dem drahtlosen Medium empfangen.

3.6.1 Problematik Duty-Cycling

Zur energieeffizienten Nutzung eines Knotens sollte im Allgemeinen ein Duty-Cycle mit möglichst langer Inaktivitätsphase gewählt werden. Um so höher der Anteil der Inaktivität pro Duty-Cycle ist, um so geringer ist jedoch der Zeitraum in dem aktive Kommunikation stattfinden kann. Somit wird auch die Anzahl der

Nachrichten, die in einem Zyklus gesendet werden können durch den Duty-Cycle bzw. den zu erreichenden Grad an Energieeffizienz beschränkt. Werden durch die Anwendung des Knotens oder durch die gewählte Codierung innerhalb eines Duty-Cycle mehr Nachrichten erzeugt als gesendet werden können, so müssen Nachrichten bis zur nächsten Aktivitätsphase des Knotens zurückgehalten werden.

Neben den induzierten Verzögerungen durch das Zurückhalten von Nachrichten, entstehen weitere Verzögerungen bei der Weiterleitung von Nachrichten anderer Knoten durch Duty-Cycling. Für die Ende-zu-Ende-Zustellung bei zunehmender Netzwerkgröße sind daher oft ergänzende Vorkehrungen bei Duty-Cycling zu treffen, um die Latenzen zu reduzieren.

Besonders in Kombination mit der Nutzung von ratenlosen Codes (siehe Abschnitt 3.3.1) können sich die durch Duty-Cycling erzeugten Verzögerungen als problematisch erweisen, da die gewünschte Redundanz über die Anzahl der erzeugten Nachrichten gesteuert wird.

Während sich Duty-Cycling bei paarweiser Unicast-Kommunikation zwischen zwei Knoten gut eignet, ist aber mit erhöhtem Synchronisationsaufwand für Broadcasting-Kommunikation mittels sogenannter *Beacons* zu rechnen. Diese können beispielsweise genutzt werden, um die Mitte der Aktivitätsphase der Knoten zu kennzeichnen. Die Kennzeichnung durch ein Beacon bietet den Knoten dann eine einfache Methode zur Synchronisation mit ihren Nachbarknoten. Sie können ihre eigene Aktivitätsphase dann entsprechend anpassen, indem die Mitte der eigenen Aktivitätsphase an einen Zeitpunkt in der Nähe der Mehrheit der anderen Beacons verschoben wird.

Multicasting ist im Zusammenhang mit Duty-Cycling hier besonders schwierig zu koordinieren. Durch die Überlagerung von mehreren Multicast-Bäumen kann in einem Multisource-Multisink-Multicast-Szenario schnell eine Ausgangslage entstehen, in der Duty-Cycling nicht länger zweckmäßig eingesetzt werden kann.

Ist ein Mitlauschen zur Realisierung von Mehrwegeausbreitung erforderlich, kann ein Knoten nicht länger die typischen Inaktivitätsphasen des Duty-Cycling nutzen.

Hier eignen sich alternativ zentrale Verfahren zum Medienzugriff, die eine Koordination auf Grundlage von vorausberechneten Zeitpunkten vornehmen, wie [75, 110] oder dezentrale Zeitmultiplexverfahren mit zusätzlichen Verfahren zur dezentralen Zeitsynchronisation zum Medienzugriff.

3.6.2 Lösungsansätze zur Vereinbarkeit

In [15] wurde eine Medienzugriffsschicht präsentiert, die sowohl Duty-Cycling, als auch Netzwerkcodierung unterstützt. Die Kernidee sieht jedoch vor, die übermittelten Nachrichten einem Nachrichtenstrom zuzuordnen. Die Inaktivitätsphasen lassen sich dann entsprechend der Länge eines Datenstroms anpassen. Die Ergebnisse der Arbeit zeigen bei der Übermittlung von Datenströmen mit Netzwerkcodierung, allerdings ohne Verwendung von Duty-Cycling, ein Einsparpotential von 20%-30% der benötigten Energie. Die Ergebnisse lassen sich jedoch nicht auf die Übertragung von Einzelnachrichten abbilden, da die Medienzugriffsschicht die Nutzung von

negativen Empfangsbestätigungen (NACKs) im Fall von Paketverlusten vorsieht. Bei Einzelpaketen kann der Empfänger den Verlust jedoch nicht vorhersehen, da keine Nachfolgepakete detektiert werden.

YA-MAC [120] stellt dagegen einen anderen Ansatz für drahtlose Sensornetze vor, der den Medienzugriff ausgehend vom Empfänger einer Nachricht koordiniert. Mit dem Ansatz kann Broadcasting parallel zu einem aktiven Duty-Cycling durchgeführt werden, um die Energieeffizienz eines Sensornetzes zu erhöhen. Dazu wird ergänzend zur Synchronisation mit Beacons das EBS-Verfahren [119] angewendet, um den Zeitpunkt der nächsten Aktivitätsphase eines Senders beim Empfänger abzuschätzen und die Dauer der Aktivitätsintervalle zu adaptieren. Ein zusätzliches Verfahren zur Zeitsynchronisation ist dabei nicht erforderlich, sondern erfolgt durch Beobachtung der Beacons von Nachbarknoten.

Eine Verbesserung für MAC-Verfahren zur Verwendung mit Netzwerkcodierung wird in [8] vorgestellt. Dabei wird die Paket-Reihenfolge im Sendepuffer eines drahtlosen IEEE802.11-Netzes so optimiert, dass das Decodieren der Pakete bei einem potentiellen Empfänger schon vor dem Senden gewährleistet werden kann. Zusätzlich kommen virtuelle Puffer für mitgehörte Pakete und opportunistische Empfangsbestätigungen zum Einsatz, die es ermöglichen lokale Informationen über die empfangenen Pakete der Nachbarn vorrätig zu halten und somit effizienteres Codieren von zukünftigen Paketen zu ermöglichen. Die Effizienz des Verfahrens ist primär von der Größe des Sendepuffers der Medienzugriffsschicht abhängig und kann daher in Sensornetzen nur in beschränktem Umfang genutzt werden.

3.7 Zusammenfassung

In diesem Kapitel wurden die theoretischen und praktischen Grundlagen von Netzwerkcodierung eingeführt. Im Anschluss wurden mögliche Codierungsverfahren und der benötigte Aufwand für die Nutzung von Netzwerkcodierung in drahtlosen Sensornetzen diskutiert. In einem weiteren Teil des Kapitels wurden ausgewählte Protokolle aus relevanten Forschungsarbeiten zur Netzwerkcodierung in drahtlosen Sensornetzen vorgestellt. Abschließend wurden die wichtigsten Kernproblematiken bei der Verwendung von Netzwerkcodierung in drahtlosen Sensornetzen erläutert. Eine besondere Rolle spielt hier die Vereinbarkeit des Codierungsschemas mit dem genutzten Protokoll zum Medienzugriff.

4. Opportunistische Kommunikation

Üblicher Weise wird versucht, die Kommunikation in Netzwerken möglichst effizient und deterministisch zu gestalten, um exakte Aussagen über die Leistungsfähigkeit der Kommunikation treffen zu können. Insbesondere die Ressourcenbeschränkungen von drahtlosen Sensorknoten legen es nahe, die Kommunikation zwischen den Teilnehmern möglichst deterministisch zu gestalten. Dazu steht jedoch die hohe Netzwerkdynamik im Widerspruch. Durch häufige Änderungen von Knotenpositionen, Aktivitätsphasen oder Knotenzuständen kann oft nicht sichergestellt werden, dass die erforderliche Kommunikation zum gewünschten Zeitpunkt möglich ist. Beispielsweise muss je nach Zeitpunkt eine unterschiedliche Route zwischen zwei Endsystemen gewählt werden, da die Energiereserven eines beteiligten Knotens im Lauf der Zeit so stark reduziert wurden, dass dieser das Weiterleiten von Kommunikationsdaten auf der gewünschten Route vermeidet.

Wird nun an Stelle einer starren und im voraus geplanten Kommunikationsform, eine Kommunikationsform verwendet, die in starkem Maße den Netzwerkzustand zum aktuellen Zeitpunkt und weitere Kontexteigenschaften, wie zum Beispiel den jeweiligen Ort zum Kommunikationszeitpunkt einfließen lässt, so wird diese Form als *opportunistische Kommunikation* bezeichnet. Sie ermöglicht beispielsweise, Datenpakete mit gleicher Quelle und gleichem Ziel bei spontanen Änderungen der Netzwerktopologie auf anderen Routen zuzustellen, anstatt die Übertragung neu initiieren zu müssen.

Durch den starken Situationsbezug können aber exakte Aussagen über die Leistungsfähigkeit oder Kommunikationseigenschaften wie *Zuverlässigkeit der Übertragungen* häufig nicht mehr garantiert werden. Daher wird versucht die gewünschten Eigenschaften mittels *Best Effort* bzw. durch Erzielen einer hohen *probabilistischen Effizienz* zu erreichen.

Definition 14. ***Probabilistische Effizienz:*** *Als probabilistische Effizienz wird in diesem Zusammenhang das Verhältnis vom Ausmaß der gewünschten Kommunikationseigenschaft in Relation zum Aufwand für Berechnung, Kommunikation und Speicherung bezeichnet. Dabei ist eine probabilistische Effizienz nur erreicht, wenn sich das effiziente Verhältnis in einem statistisch bedeutsamen Anteil der betrachteten Fälle einstellt.*

Ist beispielsweise eine hohe Informationsabdeckung die gewünschte Kommunikationseigenschaft, so wird eine hohe Effizienz erreicht, indem möglichst wenige Sende- und Empfangsvorgänge für die Übertragung der verteilten Nachrichten und die Signalisierung des Knotenzustands benötigt werden. Hinsichtlich des Speicheraufwands ist die probabilistische Effizienz hoch, wenn der Speicherbedarf für zwischengespeicherte Nachrichten in den meisten Anwendungsfällen niedrig ausfällt.

Im Gegensatz zu klassischer Kommunikation sind die folgenden Punkte Merkmale opportunistischer Kommunikation:

- eine stärker auf lokale Knoteneigenschaften ausgerichtete Kommunikation
- die bevorzugt lokale Optimierung statt globaler Optimierung
- oft mit höherer Latenz behaftete Kommunikation, da diese erst zu einem günstigen Zeitpunkt ausgeführt wird und nicht sofort
- eine probabilistische Vorgehensweise zum Erreichen möglichst hoher Effizienz, statt Leistungsgarantien
- bessere Adaptionsmöglichkeiten an die vorherrschende Netzwerksituation und vorhandene Ressourcen

Im Rahmen dieses Kapitels werden die Vorteile und Möglichkeiten opportunistischer Kommunikation in drahtlosen Sensornetzen erläutert.
Den Schwerpunkt des Kapitels bildet die Nutzung opportunistischer Methoden zum Zweck der Weiterleitung von Nachrichten anhand knoten-lokaler Entscheidungsmetriken. Darauf folgt eine Betrachtung zum Stand der Technik und bereits existierende Verfahren und Protokolle. Abschließend wird in diesem Kapitel gezeigt, wie die opportunistische Weiterleitung die Netzwerkcodierung ergänzen kann.

4.1 Einsatzbereiche

Opportunistische Kommunikation kann in drahtlosen Sensornetzen mit unterschiedlicher Zielsetzung genutzt werden. Entsprechend eignen sich je nach Zielsetzung verschiedene Herangehensweisen:

- Opportunistische Datensammlung umfasst die Adaption der Datenerfassung an die Netzwerksituation und den Knotenzustand. Dazu kann beispielsweise das erfasste und übermittelte Datenvolumen angepasst werden (siehe

[20]) oder die erfassten Daten situationsabhängig gewichtet werden. Ebenso können mobile Knoten genutzt werden, um einen größeren Messbereich abzudecken oder verschiedenartige Sensoren kooperieren, um Messdaten zusätzlicher Messgrößen zu schlussfolgern (vgl. [24]).

- Opportunistisches Routing nutzt Alternativpfade, die ggf. nur temporär zur Verfügung stehen, um die Zuverlässigkeit der Zustellung zu erhöhen oder die Kommunikationskosten auf dem Routingpfad zu reduzieren. Darüber hinaus kann auch eine beschleunigte Zustellung oder eine Zustellung mit reduzierter Latenz im Vordergrund für opportunistisches Routing stehen.

- Opportunistische Zustellung ist im Unterschied zu opportunistischem Routing dann erforderlich, wenn die Mobilität von Knoten ausgenutzt wird, um eine Weiterleitung und Zustellung von Paketen an einen Zielknoten trotz beschränkter Sendereichweite zu erreichen. Dies dient insbesondere dem Überbrücken von Netzwerkregionen, die eine schlechte Funkabdeckung aufweisen und in denen keine dauerhaften Kommunikationsbeziehungen zwischen den Knoten aufgebaut werden. Ein Beispiel für opportunistische Zustellung und opportunistisches Routing in Sensornetzen ist der Routingansatz *CHARON* [96].

- Opportunistisches Weiterleiten nutzt eine Bewertung der lokalen Situation, um einzuschätzen, ob die Weiterleitung empfangener Pakete einer allgemeinen Verbesserung der Informationsabdeckung oder dem Erreichen des Zielknotens nützt.

- Opportunistische Datenverteilung fasst opportunistisches Routing und die Weiterleitung zusammen, um möglichst hohe Zustellraten zu erreichen und nutzt dabei Kommunikationsbeziehungen mit geringen Kommunikations- oder Energiekosten aktiv aus.

Opportunistische Weiterleitung wird in drahtlosen Sensornetzen verwendet, um Mehrwege-Weiterleitung in Multipath-Topologien auf eine effiziente Art zu realisieren. Dies ermöglicht zudem, Mechanismen für schnelle Fehler-Beseitigung einzuführen, indem Nachbarknoten zur Korrektur eingebunden werden.

4.2 Grundlagen und Optimierungskriterien

Einige elementare Vorgehensweisen sind verbreitet in den betrachteten Verfahren zur opportunistischen Weiterleitung zu finden. Im Folgenden werden diese kurz diskutiert und unterschiedliche Ausprägungen vorgestellt.

Dazu gehören:

- die Weiterleitungsentscheidung und Auswahl der Zielknoten

- die Auswahl des Zeitpunktes zur Weiterleitung

- die Berücksichtigung des Zustands der Nachbarknoten
- die Beobachtung der Kommunikation in der eigenen Nachbarschaft
- die Unterdrückung von weitergeleiteten Duplikaten

Bei der Weiterleitung ist im Allgemeinen zu unterscheiden, ob die Verteilung von Einzelnachrichten oder Nachrichtenströmen im Vordergrund steht. Wird nur eine Einzelnachricht verteilt, so beschränkt sich die Zustandshaltung der Sensorknoten für die Nachrichtenverteilung auf die Identifikation bekannter Nachrichten und ein Zwischenspeichern von empfangenen oder weitergeleiteten Nachrichten für etwaige Sendewiederholungen. Nachrichtenströme erfordern typischerweise größere Zwischenspeicher und ermöglichen das Identifizieren oder Gruppieren zusammengehöriger Nachrichten. Dies erlaubt beispielsweise effizientere Verfahren zur Aggregation oder das Erkennen von fehlenden Nachrichten, sowie eine anschließende Rekonstruktion unter Einbeziehung weiterer Nachbarknoten.

Ein weiteres Unterscheidungsmerkmale ist die Adressierung bei der Weiterleitung und Verteilung. Hier stehen sich Verfahren auf Grundlage der Position, Region, Zieladresse, Zielgruppe, benötigten Sensorik oder des gewünschten Knotenzustands, wie zum Beispiel räumliche Nähe zum sensorisch erfassten Untersuchungsobjekt, den Daten-zentrierten Ansätzen gegenüber.

Definition 15. ***Daten-zentrierte Weiterleitung:*** *Daten-zentrierte Weiterleitung nutzt, statt einer festgelegten Adressierung, Methoden zur Filterung der im Netzwerk verteilten Daten. Dies kann üblicherweise anhand von Metadaten geschehen, die separat oder mit den Nutzdaten übertragen werden.*

Daher basieren daten-zentrierte Ansätze häufig entweder auf Flooding und Gossiping (siehe Kapitel 4.2.2 und Kapitel 4.2.3) oder die Knoten verteilen wie in [46] zunächst eigene Informationen in Form von sogenannten *Interests* oder *Subscriptions* im Netzwerk, um zu signalisieren, an welchen Informationen sie interessiert sind. Werden anschließend die gewünschten Informationen verteilt und treffen auf die vorverteilten Interests, dann dienen diese der Filterung und Steuerung der Weiterleitung.

Definition 16. ***Zielknoten:*** *Eine Weiterleitungsentscheidung umfasst bei Broadcast-Kommunikation typischerweise nur die Entscheidung, ob ein Knoten eine Nachricht weiterleitet oder unterdrückt. Wird stattdessen Unicast-Kommunikation verwendet oder kann beispielsweise anhand des Sendezeitpunktes bei entsprechender Netzwerkdynamik eine Selektion der Empfänger einer weitergeleiteten Nachricht getroffen werden, so werden diese Knoten als Zielknoten bezeichnet. Zielknoten sind daher Nachbarknoten die über einen einzelnen Hop erreicht werden können. Sie unterscheiden sich von adressierten Knoten, als dass sie nicht notwendiger Weise Knoten sind, die Informationsquellen oder -senken darstellen.*

Zur Verteilung von Informationen in drahtlosen Sensornetzen wurden eine Reihe von Strategien, Verfahren und Protokollen entwickelt und in Hinblick auf unterschiedliche Kriterien optimiert:

- Informationsabdeckung – Anteil erreichter Knoten
- Zuverlässigkeit der Zustellung – Wahrscheinlichkeit der Informationsabdeckung eines ausgewählten Knotens
- Speicherbedarf – Puffergrößen zur Zwischenspeicherung
- zur Verteilung benötigte Zeit – Ende-zu-Ende-Latenz
- Fairness, Lastverteilung und Energieverbrauch – ausgewogene Auslastung vorhandener Ressourcen, um Netzwerkverhalten und Verfügbarkeit zu optimieren
- Vermeidung von Stausituationen – Adaption an Netzwerkdichte und Nachrichtenvolumen
- Kommunikationsaufwand und Effizienz – Optimierung der Anzahl gesendeter oder empfangener Pakete

Definition 17. ***Informationsabdeckung:*** *Die Informationsabdeckung (auch Zustellrate) bezeichnet den Anteil der adressierten bzw. interessierten Knoten, die nach Durchführung des Weiterleitungsverfahrens die zu verteilende Nachricht fehlerfrei empfangen haben und somit über aktuelle Informationen verfügen. Knoten die weder adressiert noch interessiert sind, die verteilte Nachricht zu erhalten, werden nicht zur Auswertung der Informationsabdeckung herangezogen. Der mehrfache Empfang von zu verteilenden Nachrichten durch einen Knoten erhöht die Informationsabdeckung nicht. Zur Erhöhung der Informationsabdeckung kann ein adressierter bzw. interessierter Knoten aber die erhaltene Nachricht an weitere Zielknoten in der Nachbarschaft weiterleiten.*

Anmerkung 2. ***Zuverlässigkeit der Zustellung:*** *Hier wird im Unterschied zur Informationsabdeckung bewertet, mit welcher Wahrscheinlichkeit ein Knoten mit Informationen abgedeckt wird. Dazu kann beispielsweise die Verteilung über unterschiedliche Pfade beitragen, so dass ein Knoten die redundante Informationen enthält, die auch bei Störeinflüssen eines Teils der Verbindungen die Informationsabdeckung weiterhin gewährleistet.*

4.2.1 Opportunistische Entscheidung über Weiterleitung

Jeder Knoten, dem eine Nachricht zur potentiellen Weiterleitung vorliegt, hat die Entscheidung zu treffen, ob die Weiterleitung einen Mehrwert darstellt.

Ein wesentlicher Vorteil drahtloser Kommunikation ist hier die Eigenschaft, Pakete in der lokalen Nachbarschaft mithören zu können (sogenanntes *Overhearing*). Overhearing kann aktiv ausgenutzt werden, um den Datenverkehr zu koordinieren und zu reduzieren, der in lokalen Nachbarschaften bei der Weiterleitung mittels Gossiping-Techniken entsteht.

Ob ein Paket an die eigene Nachbarschaft weitergeleitet werden soll, lässt sich lokal auf einem Sensorknoten entscheiden, wenn der potentielle Informationsgewinn der Nachbarschaft durch die Weiterleitung betrachtet und mit dem eigenen Aufwand für die Weiterleitung verglichen wird. Es gilt hierbei abzuwägen, ob:

- ein Optimierungskriterium (z. B. die Informationsabdeckung) des Gesamtnetzes durch die Weiterleitung verbessert wird
- Abhängigkeiten anderer Knoten für die Weiterleitung durch die aktuelle Netzwerktopologie gegeben sind
- ausreichend Energieressourcen zur Aufrechterhaltung der Funktionsfähigkeit des Knotens gegeben sind
- eine zu erreichende Zuverlässigkeit der Zustellung für einen Knoten z. B. durch die Anwendung vorgegeben wurde
- potentielle Nachbarknoten die gleiche Aufgabe effizienter erfüllen können
- die Weiterleitung eine Stausituation hervorrufen oder verschärfen könnte
- Schleifen bei der Weiterleitung entstehen können
- unnötige Redundanz durch die Weiterleitung entsteht
- sich im Netzwerkbereich, der durch die Weiterleitung erreicht werden kann, mögliche Adressaten oder Interessenten befinden

Zur Analyse, ob das Paket zurückgestellt oder verworfen werden soll, haben sich mehrere unterschiedliche Strategien etabliert:

- lokale Metriken
- globale Optimierung oder Vorberechnung des Weiterleitungsverhaltens
- Bewertung des Informationswertes einer Nachricht

Lokale Metriken werden aus Beobachtungen des Knoten- oder Netzwerkzustands gewonnen. Sie eignen sich insbesondere für verteilte und dynamische Netzwerke, da zu ihrer Ermittlung nicht oder nur in sehr geringem Ausmaß zwischen benachbarten Knoten kommuniziert werden muss. Beispiele für lokale Metriken sind:

- die Füllrate von Zwischenspeichern
- die Häufigkeit von Paketkollisionen
- die Anzahl der Nachbarknoten
- die Verbindungsqualität zu Nachbarn
- der Anteil der quittierten Nachrichten
- die Sende- oder Empfangshäufigkeit

- lokale Energiereserven
- die verfügbaren Hardwareressourcen wie verbleibende Speicherkapazität und Rechenleistung
- die Zahl der Aufgabenstellungen die parallel durchgeführt werden müssen, z. B. regelmäßige Sensormessungen
- die Einstellungen des Knotens, z. B. zur Sendeleistung und zum Duty-Cycle

Das in [98] vorgestellte *RBP*-Protokoll (Robust Broadcast Propagation) ist exemplarisch für die Weiterleitungsentscheidung mit einer lokaler Metrik. RBP benötigt lediglich die Anzahl der Nachbarn, um daraus auf die Dichte der Topologie in der eigenen Netzwerkregion zu schließen. Zusätzlich hört jeder Knoten die Kommunikation in der Nachbarschaft mit, um mögliche Kommunikationsbeziehungen zwischen Nachbarn zu identifizieren. Mit diesen Informationen nimmt RBP schließlich eine Anpassung der Sendewiederholungen vor und unterdrückt Sendewiederholungen, wenn ein ausreichend großer Teil der Nachbarn eine Empfangsbestätigung gesendet hat.

Globale Optimierungen werden im Unterschied zu den lokalen Zustandsmessungen von zentralen Knoten, wie z. B. einer Senke vorgenommen. Diese müssen zunächst im Netzwerk propagiert werden und erfordern daher zusätzlichen Kommunikationsaufwand, so dass sich diese Vorgehensweise eher für kleine Netzwerkgrößen empfiehlt. Stau- und Lastverteilungsaufgaben lassen sich durch globale Optimierungen im Allgemeinen effizienter lösen. Eine globale Vorgehensweise wird z. B. in [105] verwendet.

Soll stattdessen der Informationswert einer Nachricht herangezogen werden, so ist die Weiterleitungsentscheidung auf Basis des Nachrichteninhaltes zu treffen. Dazu kann ein einfache Kennzeichnung der Priorität im Kopf der Nachricht durchgeführt werden oder eine vollständige Interpretation des Nachrichteninhaltes durch den weiterleitenden Knoten notwendig werden.

Ein Beispiel bietet [9]. Dort wird durch die Quelle ein Informationswert für jede Nachricht vergeben. Die Auswahl der weiterzuleitenden Nachrichten wird daraufhin mittels einem internen Schwellenwert durchgeführt, für den die verfügbaren Energiereserven und Zustandsinformationen aus dem erwarteten Weiterleitungsverhalten der Nachbarknoten herangezogen wird. Diese Vorgehensweise erlaubt die Maximierung des Informationswertes zugestellter Nachrichten an der Senke.

4.2.2 Flooding

Das *Fluten* (engl. Flooding) stellt das am einfachsten zu realisierende Verfahren zur Nachrichtenverteilung in drahtlosen Sensornetzen dar. Dabei leitet jeder einzelne Knoten die empfangenen Nachrichten an alle Nachbarn weiter, ohne Knoten-, Netzwerkzustand oder Nachrichteninhalt zu berücksichtigen. Der Aufwand zur Zustandshaltung auf einem weiterleitenden Knoten ist somit minimal. Insbesondere in Netzwerken mit einer sehr dynamischen Topologie erweist sich diese

Zustandslosigkeit von Vorteil. Mittels Broadcast kann das Weiterleiten durch einen einzelnen Sendevorgang pro Knoten erfolgen.

Der Einsatz von Flooding ist in drahtlosen Sensornetzen häufig nicht besonders effizient, da Fluten keine gezielte Adressierung von Empfängern vorsieht. So kann jede Nachricht mehrfach von unterschiedlichen Nachbarn empfangen werden, die weder Empfängerknoten sind, noch effizient in Richtung eines Empfängerknotens weiterleiten können. In Folge werden Nachrichten unnötig, ggf. auch mehrfach, durch einen Nachbarknoten weitergeleitet. Eine einfache Verbesserungen der Effizienz kann daher schon durch Zwischenspeichern und Unterdrücken der Weiterleitung mehrerer identischer Nachrichten erreicht werden. Dazu ist eine Erkennung anhand des Nachrichtenkopfes (z. B. anhand der Sequenznummer und Quelladresse) oder des Nachrichteninhalts nötig, die identische Nachrichten identifizieren kann. Flooding erzielt eine vollständige Informationsabdeckung eines nicht-partitionierten Sensornetzes. Auch im Fall von Störungen und Verbindungsfehlern kann oft noch eine sehr hohe Informationsabdeckung durch Flooding erreicht werden. Dazu reicht es, dass jeder Knoten bereits genau einmal eine empfangene Nachricht weiterleitet.

Eine weitere Verbesserung sieht das theoretische Modell in [48] vor. Indem das Netzwerk in kreisförmige Hop-Zonen um die Senke unterteilt wird, kann durch Kombination von Mithören der Kommunikation zwischen Nachbarknoten und CSMA, ein gerichteter Broadcast realisiert werden. Dafür ist jedoch ein hoher Rechenaufwand bei der Ermittlung eines geeigneten Knotens zur Weiterleitung nötig, was die Berechnung von Schnittmengen der Sendereichweite erfordert. Signifikante Verbesserungen der Effizienz zu Flooding mit Unterdrückung von identischen Nachrichten konnten daher erst in großen Netzen mit mehr als 20 Hops gezeigt werden.

Die ungerichtete Kommunikation bei Flooding erhöht darüber hinaus auch das Kollisionsrisiko von weitergeleiteten Nachrichten. Daher sollte der Zeitpunkt der Nachrichtenweiterleitung durch ein Medienzugriffsverfahren koordiniert werden, dass das Überprüfen des Mediums auf Belegung mittels sogenannter *Clear Channel Assessments* (CCA) oder das Hinzufügen einer künstlichen Verzögerung vor der Weiterleitung vorsieht (z. B. durch *Random Backoff*).

Das von Medienzugriffsverfahren zur Reduzierung des Energieverbrauchs in drahtlosen Sensornetzen eingesetzte *Duty-Cycling* ist im Zusammenspiel mit Broadcasting jedoch problematisch. Um an Nachbarknoten weiterzuleiten, die sich während des Broadcasts im Ruhezustand befinden, sind zusätzliche Sendewiederholungen zu einem späteren Zeitpunkt notwendig.

Eine detaillierte Analyse und geeignete Maßnahmen zur Koordination des Sendezeitpunktes bei Sensorknoten mit Duty-Cycling und die Auswirkungen auf die Konnektivität des Netzwerks werden in [18] betrachtet. In dieser graphentheoretischen Arbeit konnte festgestellt werden, dass zur Aufrechterhaltung der Konnektivität des Netzes beim Reduzieren der Wachphasen eine erhöhte Sendereichweite erforderlich wird. Mangels Berücksichtigung eines geeigneten Energiemodells

und der fehlenden Berücksichtigung von Kollisionen und asymmetrischen Verbindungen konnten von den Autoren aber keine Aussagen über die Effizienz und Kommunikationskosten von Broadcasting im Zusammenspiel mit Duty-Cycling getroffen werden.

Im Allgemeinen neigen Protokolle, die auf Flooding basieren, zu Stau-Situationen bei hohem Nachrichtenaufkommen. Eine weitere Problematik ist die Neigung zur Bildung von Weiterleitungsschleifen. Diese können beispielsweise auftreten, sobald eine weitergeleitete Nachricht aufgrund eines hohen Nachrichtenaufkommens oder bei langen Schleifenpfaden zwischenzeitlich aus dem Zwischenspeicher verdrängt wurde.

Wie Flooding opportunistisch zur Nachrichtenverteilung eingesetzt werden kann, zeigt das Protokoll *Cascades* [85]. Dabei sieht das Protokoll im Wesentlichen eine Baumstruktur als Topologie zum zuverlässigen Fluten vor, so dass jeder Knoten seine Nachbarn in *Eltern-* und *Kind-Knoten* unterteilt. Cascades erweitert Flooding um Mechanismen zur expliziten Bestätigung durch *Acknowledgements* (ACK) und *Negative Acknowledgements* (NACK). Zusätzlich werden implizite Bestätigungen eingesetzt. Dazu setzt Cascades stark auf das Mithören von Nachrichten der Nachbarknoten. Das *Broadcast Storm Problem* (vgl. [82] und Kapitel 5.2.2) wird in Cascades damit effektiv auf die lokale Nachbarschaft beschränkt. Bleiben implizite oder explizite Empfangsbestätigungen durch die Kind-Knoten aus, so wird die Übertragung periodisch durch die jeweiligen Eltern-Knoten wiederholt. Nachrichten, die nicht entlang des Baumes geflutet werden, können dennoch durch das Mithören opportunistisch einbezogen und genutzt werden und beschleunigen nicht nur die Nachrichtenverteilung, sondern reduzieren auch die Anzahl der versendeten Empfangsbestätigungen. Cascades benötigt teilweise nur 62,5% der in Drip (siehe Kapitel 4.3.1) gesendeten Pakete, hat im Gegenzug jedoch einen erhöhten Speicheraufwand, da unbestätigte Nachrichten zwischengespeichert werden.

4.2.3 Gossiping

Verteilungsverfahren auf Grundlage von *Gossiping* nutzen Algorithmen, die sich das epidemische Ausbreiten von biologischen Organismen zum Vorbild nehmen. Im Unterschied zu Flooding-Verfahren kontrolliert der Algorithmus die Senderate oder wählt einzelne Nachbarknoten zur Weiterleitung aus.

Die Weiterleitungsentscheidung findet nicht auf Grundlage von Adress- oder Routinginformationen statt, sondern wird anhand des Knotenzustands bzw. lokal ermittelbarer Netzwerkmetriken durchgeführt. Beispielsweise kann das aktuelle Datenverkehrsaufkommen als Metrik genutzt werden, um adaptiv die Sendehäufigkeit und Anzahl der Empfängerknoten in der Nachbarschaft für die Weiterleitung zu ermitteln.

Viele Gossiping-Verfahren nutzen zur Weiterleitung auf den Knoten eine Broadcast-Kommunikation wie beim Flooding. Dabei wird jedoch, im Unterschied zu Flooding, nicht jede eingehende Nachricht weitergeleitet.

Leiten die Knoten aktiv Nachrichten an ausgewählte oder alle Nachbarn weiter, so wird diese Vorgehensweise üblicherweise als *Push*-Strategie bezeichnet (vgl. [3]).

Bei der dazu entgegengesetzte Pull-Strategie werden die Knoten beim Ausbleiben von aktuellen Nachrichten selbst aktiv, indem diese ausgewählte Nachbarn abfragen.

Opportunistische Weiterleitungsverfahren setzen wegen der dezentralen Realisierbarkeit bevorzugt auf Gossiping-Verfahren als Grundlage. Wie bei Flooding kann durch Gossiping auch bei hoher Dynamik zur schnellen Adaption an die vorherrschende Netzwerktopologie eingesetzt werden. Im Allgemeinen ist mit einem reduziertem Nachrichtenvolumen zu rechnen, was auf die Nachrichten- und Zielknotenselektion vor dem Senden einer Nachricht zurückzuführen ist. Trotzdem ist mit Gossiping-Verfahren mit hoher Wahrscheinlichkeit eine vollständige Informationsabdeckung in Sensornetzen erreichbar, die aber nicht in jedem Einsatzszenario garantiert werden kann.

Ein verbreitetes Verteilungsverfahren, dass auf Gossiping basiert, ist Trickle [67]. Ursprünglich als Verfahren zur Verteilung von Updates des Programmcodes entworfen, findet es inzwischen als Bestandteil zahlreicher Protokolle Verwendung. Trickle verteilt neue Programmcode-Updates von Hop zu Hop mittels sogenanntem *Polite Gossiping*. Bei Polite Gossiping hören die Knoten jeweils die Kommunikation ihrer Nachbarn mit, um feststellen zu können, ob ein Nachbar ein aktuelleres Programmcode-Update verteilt, als bereits lokal zur Verfügung steht. Wird ein aktuelleres Programmcode-Update beobachtet, so übernimmt ein Knoten dieses. Im Gegenzug verteilt ein Knoten den eigenen Programmcode, wenn ein älteres Update zwischen den Nachbarn kommuniziert wird. Jeder Knoten passt während der Verteilung zudem die eigene Senderate an, um möglichst wenige unnötige Nachrichten im Netzwerk zu verteilen. Dazu ermittelt der Knoten die Häufigkeit, mit der Nachbarn das identische Programmcode-Update verteilen. Auf diese Weise ist es möglich, selbstregulierend und dezentral eine vollständige Informationsabdeckung zu erreichen, auch wenn keine Informationen zur Netzwerkdichte lokal vorliegen. Trickle bemüht sich dabei um ein Gleichgewicht zwischen schneller Verteilung der Updates, zuverlässiger Verteilung mit hoher Informationsabdeckung und Skalierbarkeit.

4.2.4 Probabilistische Weiterleitung

Eine Untergruppe der Gossiping-Verfahren sind die probabilistischen Weiterleitungsverfahren. Die Weiterleitungsentscheidung ist dabei zufallsgesteuert, so dass z. B. die Zielknoten zur Weiterleitung aus den verfügbaren Nachbarn gezogen werden. In der einfachsten Form leitet jeder Knoten die eingehenden Nachrichten mit einer festen Wahrscheinlichkeit p_{forw} an seine Nachbarn weiter. Alternativ kann eine parametrisierte Wahrscheinlichkeitsverteilung verwendet werden, um die Weiterleitung gezielter zu steuern.

Die Zufallsauswahl der Nachbarn und p_{forw} kann dazu durch eine Vielzahl möglicher Parameter beeinflusst werden. Typische Parameter sind hier:

- lokaler Verteilungsgrad – Anteil der Zielknoten aus der Menge der Nachbarknoten

- Informationsabdeckung – Gewünschtes Niveau der Informationsabdeckung nach dem Ende der Nachrichtenverteilung
- Redundanz gegen Nachrichtenverlust – Parameter zur Anpassung der Weiterleitungshäufigkeit an die zu erwartende Nachrichtenverlustrate während der Kommunikation

Ein Protokoll zur opportunistischen Zustellung in einem Netzwerk mit mobilen Knoten mittels probabilistischer Nachrichtenweiterleitung ist *Spray and Wait* [97]. Die Kernidee ist es, in der sogenannten *Spray-Phase* durch Verteilung der Nachrichten eine möglichst hohe Informationsabdeckung im Netzwerk zu erreichen. In der anschließenden *Wait-Phase*, während der keine Weiterleitung mehr stattfindet, sich die Knoten aber bewegen, soll die Nachricht dann ausreichend nah an dem gewünschten Empfängerknoten herankommen, um zugestellt zu werden. Für eine möglichst effiziente Vorgehensweise sollen dabei in der Spray-Phase möglichst wenige Nachrichten verteilt werden.

In [11] werden Weiterleitungsstrategien für ein Routingprotokoll entworfen, die eine möglichst hohe Informationsabdeckung erreichen sollen. Dazu wird die Weiterleitungswahrscheinlichkeit p_{forw} eines Knotens in Abhängigkeit von der Hop-Distanz zu Senke und Quelle angepasst. Die dafür erforderlichen Informationen über die Netzwerktopologie sind üblicherweise nicht lokal verfügbar, können aber durch schrittweisen Austausch von Zustandsinformationen im Netzwerk verteilt werden. Somit kann diese Vorgehensweise zwar nicht in Netzwerken eingesetzt werden, die einer sehr starken Dynamik unterliegen. Die Autoren konnten aber zeigen, dass sich eine gewünschte Informationsabdeckung garantieren lässt, wenn ein Kompromiss zwischen Informationsabdeckung, Netzwerklast und durch Störeinflüsse verfälschte Nachrichten toleriert wird.

Weighted Probabilistic Data Dissemination (WPDD) [22] verfolgt einen probabilistischen Ansatz zu Datenverteilung, der von biologischen Systemen mit Feedback-Schleifen inspiriert ist. Das Protokoll erlaubt die Priorisierung von Nachrichten anhand eines durch die Anwendung vorgegebenen Wertes. Die Priorität der Nachricht dient dann zur Gewichtung einer Nachricht und zur Anpassung der Weiterleitungswahrscheinlichkeit p_{forw}. Zudem erlaubt WPDD die Anpassung der weitergeleiteten Nachrichtenmenge auf Grundlage der Gewichtung, um die Staugefahr zu reduzieren und die Netzwerklast zu steuern. Dazu werden etwa aggregierte Nachrichten höher gewichtet und mit höherer Wahrscheinlichkeit weitergeleitet.

4.2.5 Verbesserungen statischer Gossipingverfahren mit konstanten Weiterleitungswahrscheinlichkeiten

Soll eine vollständige Informationsabdeckung mittels Gossiping unabhängig von der vorliegenden Topologie erreicht werden, indem alle Knoten mit konstanter Wahrscheinlichkeit p_{forw} weiterleiten, so zeigt sich, dass p_{forw} einen Wert nahe 1 besitzen muss. Daher sieht das Protokoll *Smart Gossip* [58] eine dezentrale Anpassung und Optimierung des Gossiping-Verfahrens an die unterliegende dynamische

Topologie vor. Dabei steht das Ziel im Vordergrund, eine möglichst effiziente Weiterleitung per Mehrwegeausbreitung mit wenigen Sendevorgängen zu realisieren, um den Energieverbrauch niedrig zu halten.

Die Berechnung von p_{forw} erfolgt, indem zunächst die Knotenbeziehungen für die Weiterleitung ermittelt werden. Dabei wird zwischen logischen Eltern-Kind- und Geschwisterbeziehungen unterschieden. Diese Einteilung erlaubt es zu bestimmen, welche Knoten nur über bestimmte Nachbarknoten erreichbar sind und welche Knoten über unterschiedliche Pfade erreicht werden können. Dazu werden mit der zu verteilenden Nachricht Zustandsinformationen versendet und von jedem Knoten aktualisiert, die Eltern-Knoten identifizieren und die erforderliche Weiterleitungswahrscheinlichkeit für die Eltern-Knoten ausweisen. Sind gleichzeitig mehrere Quellen im Netzwerk aktiv, so muss die Weiterleitungswahrscheinlichkeit der Knoten für jede Quelle daher neu berechnet werden. Zusätzlich wird die Übertragungsfehlerrate zwischen benachbarten Knoten bei der Weiterleitung berücksichtigt, indem ein von der Anwendung spezifizierter Zuverlässigkeitswert mit der beobachteten Fehlerrate im Netzwerk verglichen wird und auf die Weiterleitungswahrscheinlichkeit von Einzelverbindungen anteilig heruntergerechnet wird. Somit ergibt sich für jeden Sensorknoten eine andere Weiterleitungswahrscheinlichkeit.

Die simulativen Ergebnisse des Protokolls zeigen jedoch, dass der zusätzliche Mehraufwand für Smart Gossiping gegenüber topologieunabhängigem Gossiping erst amortisiert, wenn die Anwendung einen Zuverlässigkeitswert über 90% spezifiziert. Duty-Cycling wird in dieser topologieadaptiven Form des Gossipings nicht berücksichtigt. Dagegen zeigen die Autoren die Vorteile und Fallstricke bei Verwendung von ausschließlich lokalen Topologieinformationen zur Weiterleitung von Nachrichten auf.

Wie ein Medienzugriffsverfahren mit statischem Duty-Cycling bei der opportunistischen Nachrichtenverteilung berücksichtigt werden kann zeigt dagegen der Ansatz in [30]. Dabei wird ausgehend von einer einzelnen Quelle die zu verteilende Nachricht im Netzwerk geflutet. Je nach Wahl kann der Ansatz insbesondere die Anzahl der benötigten Übertragungen oder die Verzögerung während der Weiterleitung optimieren. Die Kernidee des Ansatzes ist es, den Duty-Cycling-Plan eines Knotens mit den Nachbarknoten zu teilen, so dass ein Knoten nur dann den Sendevorgang zur Weiterleitung anstößt, wenn der Empfänger zu diesem Zeitpunkt auch empfangsbereit ist. Ausgehend von der Quelle und ausgehend von der Entfernung in Hops wird zum Fluten mittels der bekannten Duty-Cycling-Pläne ein azyklischer Baum konstruiert. Somit kann die erwartete Ankunftszeit einer Nachricht schon vor der Zustellung berechnet werden. Eine opportunistische Weiterleitung erfolgt in diesem Ansatz, sobald eine Übertragung durch eine Abkürzung abseits des berechneten Baumes für eine schnellere Ankunftszeit sorgen kann, als zuvor erwartet. Potentielle Stausituationen können weiter reduziert werden, indem gezielt eine Menge potentieller Vorgängerknoten zur opportunistischen Weiterleitung anhand der Verbindungsqualität ausgewählt werden. Die Versuchsergebnisse in Simulation und Testbett zeigen, dass diese Form der opportunistischen Weiterlei-

tung mit zunehmender Anzahl an Hops an Wirksamkeit gewinnt und wirksam zur Stauvermeidung beiträgt. Aber zugleich zeigt sich ein hoher Kommunikations- und Rechenaufwand zum Austausch der Duty-Cycling-Pläne und zur Berechnung der optimierten Kommunikationspfade und -zeitpunkte. Insbesondere bei sehr dynamischen Topologien z. B. mit mobilen Knoten ist der erhöhte Aufwand daher nicht gerechtfertigt. Darüber hinaus sind hier weitere Maßnahmen erforderlich, um eine lokale Zeitsynchronisation zwischen benachbarten Knoten zu gewährleisten.

4.3 Existierende Forschungsarbeiten zu opportunistischen Weiterleitungs- und Verteilungsverfahren

In diesem Abschnitt werden die existierenden Weiterleitungs- und Verteilungsverfahren für Sensornetze näher betrachtet, die sich zur opportunistischen Weiterleitung in Multipath-Topologien einsetzen lassen und auf den zuvor vorgestellten Prinzipien basieren. Zusätzlich werden exemplarische Verfahren für drahtlose Netzwerke vorgestellt, die nicht spezifisch für Sensornetze entworfen wurden, aber die typischen Merkmale von opportunistischen Weiterleitungs- und Verteilungsverfahren aufweisen und in leicht veränderter Form für die Nutzung in Sensornetzen angepasst werden könnten.

Das ExOR-Protokoll [12] ist als opportunistisches Routing-Protokoll für drahtlose IEEE 802.11-Netze entworfen. Die zuvor diskutierten Merkmale zur opportunistischen Weiterleitung für Sensornetze sind auch hier zu finden. Das Protokoll unterteilt den Weiterleitungsprozess in drei Phasen: Zunächst erfolgt eine Auswahl möglicher Zielknoten-Kandidaten anhand ihrer Distanz zur Senke in Hops. Potentielle Zielknoten mit der niedrigsten Distanz werden entsprechend ihrer Zustellraten priorisiert und im Kopf des zur Weiterleitung verwendeten Paketes als Liste vermerkt. In der zweiten Phase werden die Pakete von den Empfängern entsprechend der Reihenfolge ihrer Priorität per ACK quittiert. Dabei enthält die Bestätigung immer die ID des bekannten Knotens mit höchster Priorität, so dass implizit auch der Empfang für näher benachbarte Knoten durchgeführt werden kann. In der letzten Phase entscheiden die Empfänger selbst ob sie die erhaltene Nachricht weiterleiten oder unterdrücken. Die Weiterleitung erfolgt nur, wenn das Paket zuvor nicht auch von einem Knoten höherer Priorität erfolgte. ExOR kann mit leichten Anpassungen auch in drahtlosen Sensornetzen verwendet werden (vgl. [92]), ist jedoch vorwiegend auf einen hohen Durchsatz optimiert und berücksichtigt den Aspekt der Energieeffizienz nicht.

Ein anderes Optimierungsziel verfolgt *ORW* (Opportunistic Routing in Wireless sensor networks) [28, 59]. Um beim Duty-Cycling möglichst lange Phasen der Inaktivität zu erzeugen und auf diese Weise die Energieeffizienz der Sensorknoten zu erhöhen, sieht das Protokoll im Zusammenspiel mit einem unterliegenden Protokoll zum Medienzugriff ein opportunistisches Routing vor, das jeweils den Knoten zur Weiterleitung bevorzugt, der zuerst aus dem Schlafzyklus aufwacht. Zur Koordination der Weiterleitung werden die Knoten in Gruppen zur Weiterleitung

unterteilt, sowie auf Empfangsbestätigungen und Mithören der Kommunikation fremder Knoten.

Auch die Zustellung von Daten zu einem garantierten Zeitpunkt kann das Optimierungsziel für opportunistische Verteilungsverfahren darstellen. Zwei Protokolle, die Zeitbeschränkungen einbeziehen sind *ORTR* [52] und *BDT* [87]. Während ORTR sich auf die garantierte Zustellung der Daten in eine geographische Zielregion unter Einhaltung von Zeitgarantien konzentriert, ist der Fokus von BDT die Einhaltung von Zeitschranken zur maximalen Verzögerung bei der Zustellung der Daten. Beide Protokolle nutzen Mechanismen der Medienzugriffsschicht, um die Energieeffizienz der beteiligten Knoten zu optimieren.

4.3.1 Nachrichtenverteilung in drahtlosen Sensornetzen

Zwei verbreitete Protokolle zur Verteilung von Informationen in drahtlosen Sensornetzen werden bereits durch das Betriebssystem TinyOS [64]bereitgestellt. Die Protokolle sind das *DIssemination Protocol* (DIP) [72] und *Drip* [106].

Beide Protokolle basieren auf dem in Kapitel 4.2.3 vorgestellten Zeitgeber des Trickle-Protokolls und erhöhen den Abstand zwischen zwei Übertragungen exponentiell, solange identische Informationen auf den Knoten vorhanden sind. Wenn stattdessen veraltete Informationen während der Kommunikation entdeckt werden, wird von Nachbarknoten ein Update der Informationen gesendet.

Die Art der verteilten Daten unterscheidet sich zwischen den Protokollen. Bei Drip soll die Verteilung und Synchronisation von mehreren jeweils unabhängigen Variablen oder Parametern bezweckt werden. Ebenso können Statusabfragen im Netzwerk verteilt werden. DIP sieht dagegen primär das Verteilen und die Synchronisation von logisch zusammenhängenden Daten oder größeren Datenblöcken vor, wie beispielsweise Programmcode. Daher werden die Datenobjekte nicht unabhängig, sondern in Datensätzen verwaltet. Während DIP daher nur einen einzelnen Trickle-Zeitgeber für den Datensatz vorsieht, werden bei Drip einzelne Zeitgeber für jedes Datum verwaltet.

Zur Erkennung der benötigten Updates nutzen beide Protokolle eine Versionsnummer pro Datum. Im Unterschied zu Drip, wo die einzelnen Daten nicht gekennzeichnet sind, nutzt DIP hier ein Tupel aus Schlüssel und Versionsnummer zur Speicherung der Daten in einem Hash-Baum.

DIP optimiert die Verteilungsgeschwindigkeit indem Maßnahmen zur effizienten Suche im Hash-Baum ergriffen werden. Dazu wird eine Abschätzung ermittelt, die angibt, an welcher Position im Hash-Baum Anpassungen durch Updates notwendig sind. Diese Abschätzung dient zugleich der Anpassung des Sendeintervalls durch den Trickle-Zeitgebers. Zur Ermittlung der Abschätzung ist ein Bloomfilter und drei Nachrichtentypen vorgesehen. Neben normalen Datennachrichten dienen hier sogenannte *Vector-* und *Summary-Nachrichten*. Diese transportieren keine Nutzdaten, sondern annoncieren lediglich Änderungen von Datenobjekten. Die Position der Änderungen im Hash-Baum kann anschließend mittels des Bloomfilters abgeschätzt werden.

Drip erreicht eine möglichst niedrige Latenz im Gegenzug zu DIP, indem die Trickle-Zeitgeber unabhängig verwaltet werden. Bei Änderungen eines einzelnen Datenobjektes können so zeitnah Updates versendet werden. Für eine möglichst hohe Effizienz fassen beide Protokolle zudem mehrere Datenobjekte in den gesendeten Nachricht zusammen.

4.4 Vereinbarkeit mit Netzwerkcodierung

Die Kombination aus Netzwerkcodierung und opportunistischer Mehrwegeausbreitung mittels Broadcasting, Gossiping oder probabilistischer Weiterleitung und Overhearing bietet das Potential, sowohl auf spontane Topologieänderungen, Fehlersituationen und Änderungen des Datenverkehrs zu reagieren, als auch die theoretischen Kapazitätsgrenzen besser auszuschöpfen, Stausituationen zu vermeiden und durch redundante Informationen die Zuverlässigkeit der Nachrichtenverteilung zu erhöhen oder verlorene Pakete durch Nachbarn zu rekonstruieren.

Wird Netzwerkcodierung in Kombination mit einem Routingprotokoll eingesetzt, so ist das Potential für Codierungsmöglichkeiten stark von den verfügbaren Routing-Pfaden abhängig (vgl. [114]).

Opportunistische Kommunikation kann dagegen ohne vorheriges Wissen über die momentane Netzwerktopologie stattfinden. Eine Änderung der Netzwerktopologie ist jederzeit möglich ohne dass der Austausch von Statusinformationen zwischen benachbarten oder weiter entfernten Knoten nötig wird.

In sehr dynamischen Netzwerken kann opportunistische Weiterleitung in Verbindung mit Netzwerkcodierung oft ein dediziertes Routingverfahren vollständig ersetzen und zugleich durch spontane Codierungsmöglichkeiten profitieren.

Insbesondere kann bei einem Verfahren dass nur auf lokalen Mechanismen zur Weiterleitung aufbaut der Mehraufwand für das Konstruieren, Erhalten und Reparieren der Pfade eingespart werden. Weiteres Einsparpotential bieten Sendebestätigungen und der Austausch von Statusinformationen zur Verbindungsqualität oder Pfadwahl, die bei Routingprotokollen oft über mehrere Hops hinweg erfolgen.

Ein neues Problem der Netzwerkcodierung ist aber der Verlust eines Paketes, welches Teile von einer Vielzahl an codierten Nachrichten enthält. Der Verlust des Paketes kann folglich dazu führen, das mehrere bereits vorhandene Pakete am Zielknoten nicht decodiert werden können. Daher werden codierte Pakete per Mehrwege-Weiterleitung bevorzugt über mehrere Pfade gesendet, so dass Störungen und Paketverluste auf einzelnen Verbindungen ausgeglichen werden können.

Im folgenden Abschnitt werden nun die bestehenden Forschungsansätze zur Vereinigung von opportunistischer Kommunikation und Netzwerkcodierung mit ihren jeweiligen Forschungsschwerpunkten kurz vorgestellt.

Ein früher Ansatz, der Netzwerkcodierung und opportunistische Nachrichtenverteilung mittels Gossiping verbindet ist *NBGossip* [74]. In dieser theoretischen Forschungsarbeit werden verschiedene Gossiping-Verfahren mit einem durch Netzwerkcodierung erweitertes Gossiping anhand der Anzahl benötigter Weiterlei-

tungsschritte verglichen. Es wird dabei lediglich eine einzelne Nachrichtenquelle angenommen und jeder Knoten der erfolgreich eine Nachricht decodieren konnte nimmt nicht mehr an dem weiteren Verlauf der Kommunikation teil, so dass die vorhandenen Informationen z. B. nicht länger dazu genutzt werden können, um Paketverluste bei Nachbarknoten auszugleichen.

In [104] werden zusätzliche Mechanismen ergänzt, um die Verteilung von codierten Paketen im Netzwerk beim Gossiping gezielt zu kontrollieren. Der Ansatz konzentriert sich dabei auf Schwierigkeiten beim Decodieren, wenn neue Nachrichten initial zur Übertragung hinzugefügt werden und auf den meisten Knoten noch keine ausreichende Redundanz zum Decodieren vorhanden ist. Eine weitere Situation, die hier berücksichtigt wird, ist das Vorliegen von Nachrichten die in hohem Maße redundant im Netzwerk verteilt wurden, so dass weitere Übertragungen keinen Informationsgewinn bewirken und daher ineffizient sind. Durch die vorgeschlagene *Fanout*-Komponente wird die Anzahl der Nachrichten die in Pakete codiert und anschließend an Zielknoten in der Nachbarschaft weitergeleitet wird, in diesen Fällen angepasst.

[116] betrachtet die Zustellung von Paketen, Latenz und Mehraufwand für probabilistisches Routing in Ad-hoc Netzwerken mit mobilen Knoten.Dabei wird die Netzwerkcodierung als Alternative zu probabilistischen Routing verwendet, indem ein zu verteilender Nachrichtenstrom zunächst zu Generationen (vgl. Kapitel 3.3.4) gruppiert wird. Die codierten Pakete werden dann ebenfalls probabilistisch weitergeleitet. Dabei jedoch nur Pakete der gleichen Generation kombiniert und decodiert. Per Simulation mit einem idealisierten Medienzugriffsverfahren und dauerhaft aktiven Knoten kann dabei gezeigt werden, dass der auf Netzwerkcodierung beruhende Ansatz wesentlich stärker von der Mobilität im Netzwerk profitieren kann und zu einer etwa 10% höheren Zustellrate führt, als der rein probabilistische Ansatz zur Weiterleitung. Bei kollisionsfreier Kommunikation lassen sich bei Netzwerkcodierung alle ursprünglichen Nachrichten rekonstruieren, aber höchstens 90% der per Gossiping versendeten Nachrichten. Das hat zur Konsequenz, dass p_{forw} bei Netzwerkcodierung niedriger angesetzt werden kann, um eine vergleichbare Informationsabdeckung zu erreichen. Als Problematik stellt sich in dieser Arbeit aber der hohe Speicherverbrauch durch die Vielzahl benötigter Generationen heraus.

4.4.1 Mechanismen zur Verbesserung der Effizienz

Einen theoretischen Ansatz zur Verbesserung der Energieeffizenz bei Broadcasting wird in [45] überprüft. Dazu wird Netzwerkcodierung hier mit einem Store and Forward-Ansatz verglichen. Das Netzwerk enthält sogenannte *Cooperating Nodes*, die in der Lage sind, ihre Sendereichweite anpassen zu können um somit Einsparungen des Energieverbrauchs vornehmen zu können. In einem verteilten Netzwerk ergeben sich hier jedoch einige zusätzliche Problemstellungen, die durch die zentrale Optimierung des Ansatzes nicht gelöst werden können. Beispielsweise müssen die Distanzen aller Knoten im Vorhinein zur Optimierung bekannt sein. Der Ansatz berücksichtigt darüber hinaus bei der Berechnung der Energieeffizienz

nur den Energiebedarf zum Senden der Nachricht. Der Energiebedarf durch den Empfänger bleibt unberücksichtigt.

Einen Ansatz um die Zuverlässigkeit der Zustellung zu erhöhen wird in [80] präsentiert. Verlorene Nachrichten werden gemeinsam per Netzwerkcodierung in ein Paket codiert, so dass mehrere Empfänger unterschiedliche verlorene Nachrichten gleichzeitig wiederherstellen können. Diese Vorgehensweise ermöglicht es in den simulierten drahtlosen IEEE 802.11 Netzwerken auf ein eigenes ARQ-Verfahren zu verzichten und lässt sich weitgehend auf Sensornetzwerke übertragen. Die theoretische Analyse und Simulation berücksichtigt dabei sowohl korrelierten, als auch nicht korrelierte Paketverluste.

CODEB [68] ist ein auf Broadcasting basierendes Protokoll für mobile Ad-hoc Netzwerke, das zur Weiterleitung aber eine explizite Wahl der Zielknoten vornimmt. Der deterministische Ansatz sieht eine Codierungsschicht zwischen der Schicht für den Medienzugriff und der Netzwerkschicht vor. Alle Knoten hören aktiv die Kommunikation in der eigenen Nachbarschaft ab. Um effizient weiterleiten zu können, pflegt jeder Knoten eine Nachbarschaftstabelle und tauscht Statusinformationen über bekannte Knoten mit seiner Nachbarschaft in einer Entfernung von zwei Hops. So lässt sich beim Versenden von codierten Paketen identifizieren, welche Nachbarn zugleich die Nachricht erhalten und decodieren können. Es werden zwei Algorithmen zur Codierung verglichen. Eine auf XOR basierende Codierung konnte im Untersuchungsszenario 45% der erforderlichen Kommunikation einsparen. Dagegen wurden bei optimalem Reed-Solomon-Coding bis zu 61% weniger Nachrichten verschickt als ohne Codierung. Die in einer statischen Topologie erzielten Ergebnisse lassen sich jedoch nicht ohne weiteres auf drahtlose Sensornetze übertragen, da der hohe Speicherbedarf zur Pflege der Nachbarschaftstabelle in drahtlosen Sensornetzen nicht zur Verfügung steht.

4.4.2 Strategien zur Mehrwege-Weiterleitung

Eine Übersicht über bestehende Protokolle und Verfahren, die Mehrwege-Weiterleitung in drahtlosen Sensornetzen einsetzen, bietet [89]. Dabei ist grundsätzlich zwischen Verfahren zu unterscheiden, die eine identische Nachricht über mehrere konkurrierende Pfade versenden (*Concurrent Multipath Routing*) und Verfahren, die zur Weiterleitung einen von mehreren alternativ verfügbaren Pfaden wählen (*Alternative Path Routing*).

Ein weiteres Unterscheidungsmerkmal, ist die Art der Pfade, die zur Weiterleitung unterschieden wird:

- Pfade mit disjunkten Knoten haben mit Ausnahme der Start- und Endknoten des Pfades keine gemeinsamen Zwischenknoten und keine gemeinsamen Verbindungskanten.
- Pfade mit disjunkten Verbindungen können gemeinsame Zwischenknoten auf dem Pfad haben auf denen sich die disjunkten Verbindungskanten schneiden.
- Teilweise disjunkte Pfade haben mindestens eine disjunkte Verbindungskante.

Diese Unterscheidung ist jedoch vorrangig bei Unicast-Kommunikation von Bedeutung und kann bei Broadcast-Kommunikation oder dem Mithören von Paketen in der Nachbarschaft nicht eindeutig vorgenommen werden. Die Einsatzgründe und Auswahlkriterien für Mehrwege-Weiterleitung und die Anzahl der genutzten Pfade sind vielfältig und umfassen:

- Verbesserung des Durchsatzes und der Netzwerkkapazität z. B. durch Vermeidung von Engstellen
- bessere Lastenverteilung und gleichmäßigere Ressourcennutzung, Optimierung der Energieeffizienz und Verfügbarkeitsdauer des Netzwerks
- fehlertolerantes Routing gegen Knotenfehler und -ausfälle oder mangelnde Energiereserven auf bevorzugtem Pfad
- zuverlässige Datenübertragung durch redundante Pakete gegen Paketfehler auf einzelnen Verbindungen

Neben dem Potential der Mehrwege-Weiterleitung für effiziente Kommunikation, sind einige zusätzliche Punkte zu berücksichtigen. Durch die Zustellung von Nachrichten über verschiedene Pfade kann die Reihenfolge der Nachrichten zwischen Quelle und Senke verändert werden. Dieses Problem trifft vor allem Nachrichtenströme mit korrelierten Nachrichten. Bei Verwendung von netzwerkcodierten Nachrichten werden die Möglichkeiten zur Decodierung jedoch nicht eingeschränkt.

Ein anderes bislang ungelöstes Problem sind zuverlässige Aussagen über Garantien zu *Quality of Service*-Eigenschaften des Netzwerks für unterschiedliche Anwendungstypen (vgl. [89]). In verstärktem Maße trifft dies auch auf die Situation zu, wenn mehrere Anwendungen im gleichen Netzwerk aktiv sind, so dass hier nur probabilistische Aussagen auf Basis von Experimenten und Simulationen möglich sind.

Ein typisches Beispiel für *Concurrent Multipath-Routing* bietet das Protokoll *ReInForM* [20]. ReInForM nutzt keine Zwischenspeicherung empfangener Pakete und dient dazu, vorgegebene Anforderungen an die zuverlässige Zustellung der Nachrichten an eine Senke einzuhalten. Dazu wird auf jedem Knoten einzeln anhand des berechneten Informationswertes einer Nachricht entschieden, ob eine Nachricht weitergeleitet wird und an wie viele Knoten bzw. über wie viele teilweise disjunkte Pfade. In diese Entscheidung fließt die gemessene Kanalfehlerrate des Knotens ein. Damit die Nachricht nur in Richtung der Senke weitergeleitet wird, benötigt jeder Knoten darüber hinaus die eigene Distanz zur Senke. Ansonsten sind die Informationen, die für die Weiterleitungsentscheidung herangezogen werden, lokal ermittelbar. ReInForM ist dabei vorwiegend für das Zustellen von einzelnen Nachrichten zwischen einer Quelle und einer Senke konzipiert und nutzt daher nicht das Potential von Netzwerkcodierung, um Nachrichten mehrerer Quellen zu kombinieren.

Das Mehrwege-Routing-Protokoll *CAMP* (Coding-Aware Multi-path) [33] legt seinen Schwerpunkt auf die Verbesserung der Zuverlässigkeit bei der Paketzustellung und auf Verringerung der nötigen Latenz. Das Protokoll greift dabei auf die *ETX-Metrik* zurück, die eine Bewertung des Pfades anhand der erwarteten Übertragungen, Sendewiederholungen, Zuverlässigkeit und Paketverlustwahrscheinlichkeit vornimmt. Die Auswahl eines Pfades geschieht durch Abwägen der ETX-Metrik mit potentiell vorhandenen Möglichkeiten zur Codierung auf dem Pfad zur Verbesserung der Effizienz. Das Protokoll ist daher ein Vertreter des Alternative Path Routing. Simulationsergebnisse zeigen für kleine Topologien mit 16 Knoten eine Verbesserung des Durchsatzes gegenüber *COPE* [49] (siehe Kapitel 3.4) von bis zu 70%, durch die aktive Auswahl der Pfade zugunsten von weiteren Codierungsmöglichkeiten.

Um adaptiv hinsichtlich Kapazitäts- und Zuverlässigkeitsanforderungen auf Änderungen im Netzwerk reagieren zu können, wird in [105] ein kombinierter Ansatz aus *Directed Diffusion* [46] und Netzwerkcodierung vorgeschlagen. Die Senke sammelt hierbei Informationen über die vorhandene Netzwerkkapazität und propagiert aktiv die Sende- und Verteilungsrate im Netzwerk. Daher eignet sich der Ansatz vorrangig für Netzwerke mit geringer Dynamik. Durch die Nutzung einer sogenannten *Braided-Mesh*-Topologie mit disjunkten Verbindungen lässt sich die Netzwerkcodierung verwenden um den erreichbaren Durchsatz des Netzwerkes entsprechend des Max-Flow Min-Cut Theorems (siehe Kapitel 3) zu optimieren. Der Ansatz nutzt hierzu Unicast-Kommunikation um gezielt Zielknoten auszuwählen. Paketverluste, Störungen der Kommunikation oder Fehler der Knoten bleiben unberücksichtigt.

In [47] wird ein Mehrwege-Weiterleitungsverfahren zur Vermeidung von Stausituationen vorgestellt. Durch Nutzung der Netzwerkcodierung soll insbesondere die Anzahl der Sendevorgänge verringert werden. Die vorhandenen Knoten werden dazu in drei Knotentypen unterteilt: Messende Sensorknoten, Knoten ausschließlich zur Weiterleitung und Aggregationsknoten. Die Codierung erfolgt nur durch Aggregationsknoten. Zur Koordination des Verfahrens ist es erforderlich, dass das Netzwerk mit einer zuvor geplanten statischen Topologie ausgebracht wird. Die Topologie wird dazu rekursiv in mehrere Ebenen unterteilt wird. Die Knoten dürfen dabei nur in einer zweidimensionalen Ebene mit äquidistanten Abständen ausgebracht werden.

In [69] wird ein Ansatz zur Verbesserung der Energieeffizienz in drahtlosen Sensornetzen mit Mehrwege-Topologien präsentiert. Die grundsätzliche Idee umfasst eine Reduktion der zur Weiterleitung verwendeten Pfade unter Aufrechterhaltung der Zuverlässigkeit zur Nachrichtenzustellung. Netzwerkcodierung wird hier eingesetzt, um die Anzahl der benötigten redundanten Pfade zu reduzieren. Die Simulationsergebnisse der Forschungsarbeit, in der ein einzelnes Paar aus Quelle und Senke Nachrichten übertragen, lassen einen Vorteil von netzwerkcodierten Übertragungen erkennen, wenn die Verbindungsqualität im Netzwerk niedrig ist, aber eine hohe Zuverlässigkeit erreicht werden soll. Berücksichtigt werden muss dabei, dass die Codierung einen Mehraufwand für die benötigte Speicherkapazität

und Rechenleistung nach sich zieht. Ebenso erhöht sich die Paketgröße und die Anzahl der zu übertragender Pakete. Dagegen sinkt das Datenvolumen, das insgesamt im Netzwerk versendet wird. Ob sich der Energieverbrauch durch diese Technik erhöht oder sinkt, ist somit stark von der verwendeten Hardwarearchitektur des Sensorknotens und dem verwendeten Medienzugiffsverfahren abhängig.

4.4.3 Verteilungszeitraum

Eine theoretische Analyse der benötigten Ausbreitungszeit eines Gossiping-Verfahrens in Kombination mit Netzwerkcodierung per Random Linear Coding in [32] zeigt, dass die zur Verteilung benötigte Zeit insbesondere von der Anzahl der zu verteilenden Nachrichten abhängt. Die Ausbreitungszeit liegt in der Komplexitätsklasse $\mathcal{O}(k + T)$ für k zu verteilende Nachrichten und die Zeit T zum Fluten einer Einzelnachricht durch das Netzwerk. Sind die n Knoten des Netzwerks zufällig verteilt, gilt mit wachsender Netzwerkgröße für T wiederum $T = \mathcal{O}(logn)$. Die Autoren argumentieren daher, dass k der beschränkende lineare Faktor für die benötigte Verteilungszeit ist und bei kontinuierlich vorliegenden Daten zur Verteilung (*Pipelining*) eine Abschätzung der Verteilungszeit auf Grundlage der Dauer für einer Einzelnachricht durchgeführt werden kann. Der Analyse liegt eine dynamische Netzwerktopologie zu Grunde, die jederzeit geändert werden kann. Zeitliche Effekte des Medienzugriffsverfahrens wurden jedoch nicht berücksichtigt.

4.5 Zusammenfassung

In diesem Kapitel wurden die Grundlagen der opportunistische Kommunikation eingeführt, sowie auf die unterschiedlichen Einsatz- und Optimierungsmöglichkeiten eingegangen. Die Kommunikationsmechanismen für opportunistische Kommunikation wurden eingeführt und anhand typischer Weiterleitungs- und Verteilungsverfahren erläutert. Neben der Diskussion von Verfahren, die auf Flooding oder Gossiping basieren wurden weitere Verfahren kurz eingeführt, die einzelne Aspekte des Weiterleitungsvorgangs optimieren.

Im zweiten Teil des Kapitels wurde anhand existierender Protokolle und Forschungsansätze herausgestellt, wie sich Netzwerkcodierung und opportunistische Verfahren ergänzen können und welche Kernpunkte bei der Kombination gesondert berücksichtigt werden müssen. Insbesondere der Verzicht auf ein eigenständiges Routingprotokoll bei gleichzeitiger Nutzung eines opportunistischen Protokolls zur Weiterleitung und Netzwerkcodierung zur Koordination von Durchsatz, Zuverlässigkeit und Redundanz, macht diese Kombination in sehr dynamischen Netzen für effiziente Kommunikation besonders attraktiv.

5. Grundlegende Konzepte des OCSSIM-Kommunikationsschemas

In diesem Kapitel werden die Grundlagen und Basiskonzepte des entworfenen OCSSIM-Kommunikationsschemas vorgestellt. Das Entwurfsziel von OCSSIM ist die Vereinigung von Netzwerkcodierung und opportunistischer Nachrichtenweiterleitung zur Realisierung von Anwendungskommunikation auf Grundlage von Multi-Source-Multicast Kommunikation in drahtlosen Sensornetzen.

Um dynamische Änderungen der Netzwerktopologie zu ermöglichen, z. B. durch das spontane Hinzufügen oder Entfernen von Knoten, muss das Kommunikationsschema bei der Integration von Knoten auf umfassende Konfigurationsmaßnahmen verzichten. Stattdessen setzt OCSSIM hier auf einen opportunistischen Mechanismus, bei dem jeder Knoten nur mit seinen lokalen Nachbarknoten in direkter Sendereichweite kommuniziert. Alle Weiterleitungsentscheidungen werden auf Basis des eigenen Knotenzustandes oder Informationen getroffen, die bei der Kommunikation von direkten Nachbarn mitgehört werden können.

Dazu setzt jeder Sensorknoten zur Informationsverarbeitung das gleiche Basisprinzip eines Pipeline-Modells auf den eingehenden Paketen der benachbarten Knoten um. Eintreffende Pakete werden entweder an eine Zielanwendung auf einem beteiligten Senken- oder Quellknoten ausgeliefert oder für das opportunistische Weiterleiten innerhalb der lokalen Nachbarschaft vorbereitet. Die Knoten wenden Netzwerkcodierung auf weitergeleiteten Daten an und kontrollieren die Menge der zu verteilenden Nachrichten in der Nachbarschaft. Außerdem werden innovative Nachrichten gezielt priorisiert.

Verteilte und kooperative Anwendungen können auf Grundlage von OCSSIM implementiert werden. Solche Anwendungen können Statusinformationen mit dem Kommunikationsschema passiv aus dem Strom der ankommenden Pakete

sammeln, die netzwerkweit verteilt werden. Dies dient dazu die Übertragungen fremder Anwendungen für die eigene Kommunikation nutzen zu können, ohne dass es zu Konkurrenzsituationen um Netzwerkressourcen kommt und beabsichtigt eine bessere Auslastung der vorhandenen Netzwerkkapazitäten, wenn mehrere Anwendungen im gleichen Netzwerk aktiv sind.

5.1 Zugrunde liegendes Kommunikationsmuster

Anders als die in Kapitel 2.3 Kommunikationsmuster für typische Sensornetzanwendungen, basiert OCSSIM auf dem im Folgenden vorgestellten Kommunikationsmuster. Dabei wird grundlegend zwischen *Nachrichten* und *Paketen* unterschieden. Nachrichten enthalten Informationen, die von einer Anwendung erzeugt und von einer Zielanwendung konsumiert werden. Nachrichten werden transportiert, indem sie in einem Paket gekapselt werden. Im Weiteren wird hier die Annahme getroffen, dass keine Fragmentierung der Nachricht und Aufteilung auf mehrere Pakete stattfindet. Dagegen können prinzipiell mehrere Nachrichten im gleichen Paket gekapselt werden. Eine transportierte Dateneinheit entspricht hier also einem Paket.

Jede betrachtete Nachricht einer Übertragung repräsentiert dabei entweder eine *Anfrage* oder eine *Rückantwort* auf eine Anfrage. Dabei stellt eine Anfrage eine gerichtete Ende-zu-Ende Übertragung von Informationen von einer Senke zu einer Quelle dar, während eine Rückantwortnachricht von einer Quelle ausgeht und Informationen in Richtung Senke überträgt. Weitere *Zwischenknoten* können an einer Übertragung teilhaben, indem die Nachrichten von diesen weitergeleitet werden.

Im Allgemeinen können komplexere Kommunikationsmuster auf Übertragungen von Anfragen und Rückantworten abgebildet werden. So kann etwa ein Nachrichtenstrom als eine Menge von Rückantworten mit festgelegter Reihenfolge interpretiert werden. Einzelnachrichten können entweder als Anfragen ohne erforderliche Rückantworten oder als proaktiv gesendete Rückantwortnachrichten interpretiert werden.

Knoten die als Senke fungieren, senden Anfragenachrichten, um Informationen von einem oder mehreren adressierten Quellknoten zu erhalten.

Dabei kann jeder Knoten, der am Kommunikationsschema teilnimmt, für unterschiedliche Anwendungen und Übertragungen jeweils unabhängig die Rolle eines Senken-, Quell- oder Zwischenknotens einnehmen. Für jede Übertragung von Informationen und jedes Paar aus Senken- und Quellknoten fungieren die restlichen Knoten des Netzwerks als Zwischenknoten. Wenn mehrere Anwendungen A, B und C gleichzeitig im Netzwerk aktiv sind und den selben Knoten zur Kommunikation nutzen, kann dies dazu führen, dass der Knoten zum Beispiel für Anwendung A die Rolle der Quelle einnimmt, während Anwendung B den Knoten als Senke nutzt und Anwendung C den Knoten als Zwischenknoten zur Informationsweiterleitung nutzt.

Da jede Übertragung zwischen zwei benachbarten Knoten stets gerichtet ist, aber die Einzelübertragungen unterschiedliche Ausgangs- und Zielknoten haben, können sich die Übertragungsrichtungen bei Betrachtung eines Einzelknotens überschneiden. Diese Überschneidungen werden jedoch nicht nur durch mehrere gleichzeitig aktive Anwendungen ausgelöst. Vielmehr überschneiden sich auch die Übertragungsrichtungen eines Anfrage- und Rückantwortpärchens. Dabei ist insbesondere zu beachten, dass eine einzelne Anfrage zahlreiche Rückantworten von unterschiedlichen Quellknoten auslösen kann. So werden bereits Rückantworten im Netzwerk übertragen, während die Anfrage in Teilbereichen des Netzwerkes noch zu weiteren Quellknoten verteilt wird. Dies führt zu der Situation, dass sich die Übertragung einer Anfrage mit der dazugehörigen Rückantwort auf einem Knoten überschneiden kann.

OCSSIM verwendet Netzwerkcodierung, um den Nachrichtenverkehr in diesen Situationen dynamisch koordinieren zu können. Verschiedene sich überschneidende Übertragungen zwischen Senken- und Quellknoten werden dazu kombiniert und die dazugehörigen Nachrichten in ein gemeinsames Paket codiert. Prinzipiell ist jeder am OCSSIM-Kommunikationsschema teilhabende Knoten in der Lage, Pakete mit dem verwendeten Codierungsschema zu codieren und decodieren, sofern ausreichend Informationen dazu auf dem Knoten zur Verfügung stehen (siehe Kapitel 3.3). Die Codierung findet dabei auch auf Zwischenknoten statt, ohne dass die ursprünglichen Quellknoten oder die adressierten Zielknoten unmittelbar berücksichtigt werden müssen. Folglich kann ein Paket codierte Informationen für unterschiedliche Übertragungsrichtungen beinhalten. Das Paket muss also beim Decodieren unterschiedlich interpretiert werden, je nach dem welche Informationen beim Empfängerknoten bereits vorhanden sind und in welcher Übertragungsrichtung sich der empfangende Knoten in Relation zum codierenden Knoten befindet.

Senken müssen zum Senden von Anfragenachrichten nicht notwendigerweise neue Pakete erzeugen. Vielmehr kann eine Anfragenachricht auch in codierter Form in ein Paket eingebracht werden, das bereits die Nachrichten anderer Knoten bzw. Übertragungen enthält.

Wenn ein Quellknoten ein Paket empfängt und Nachrichten des Paketes mit den zur Verfügung stehenden Informationen decodieren kann, muss zunächst geprüft werden, ob die decodierte Nachrichten an den Knoten adressiert ist. Adressierte Quellknoten, die fähig sind die Anfrage eines Senkenknotens zu beantworten, erzeugen eine geeignete Rückantwortnachricht. Die Rückantwort kann wiederum mit anderen Nachrichten in ein gemeinsames Paket codiert werden, das überschneidende Übertragungen enthält. Insbesondere kann das Paket sowohl die codierte Rückantwort enthalten, als auch die codierte, ursprüngliche Anfragenachricht, die an weitere Quellknoten verteilt und weitergeleitet wird. Das ist vor allem dann zweckmäßig, wenn mehrere Senken als auch mehrere Quellen an einer Übertragung teilnehmen, damit gemeinsame Informationen von Knoten geteilt oder untereinander synchronisiert werden sollen.

Im Gegensatz zu vielen, der in Kapitel 3 vorgestellten Protokolle und Kommunikationsschemata, setzt OCSSIM zur Verteilung und Weiterleitung von Nachrichten kein zusätzliches Routingverfahren ein, um das Verhalten der Netzwerkcodierung oder den Durchsatz im Netzwerk zu optimieren. Durch den Verzicht auf ein eigenständiges Routingverfahren entfällt die Notwendigkeit, Routinginformationen und Route-Discovery-Nachrichten zur Aufrechterhaltung der Routingtopologie in eigenen Nachrichten zu versenden.

Ein zielgerichtetes Routing von Paketen entlang eines einzelnen Pfades, wie es in klassischen Routingverfahren üblich ist, kann in diesem Kommunikationsszenario nicht erfolgen, da das Bündeln von unterschiedlichen Übertragungen in einem codierten Paket keine Semantik mit eindeutiger Zieladresse für Pakete zulässt.

Dennoch ist es für die zielgerichtete Weiterleitung der enthaltenen Nachrichten erforderlich, dass anhand von geeigneten Metriken Weiterleitungsentscheidungen gefällt werden können. Dazu verwendet das OCSSIM-Kommunikationsschema ausschließlich eine lokale Heuristik zur Multi-Hop-Kommunikation zwischen Nachbarknoten. Ziel der Heuristik ist ein opportunistisches Ausnutzen von Coding- und Decoding-Gelegenheiten und Weiterleitungsentscheidungen basierend auf mitgehörter Kommunikation aus der direkten Nachbarschaft oder dem Zustand der internen Nachrichtenpuffer. Somit kann sehr dynamisch auf spontane Topologieänderungen und veränderte Verkehrssituationen im Netzwerk reagiert werden. Auch Situationen mit einer variierenden Anzahl von Quell- und Senkenknoten, sowie Multi-Source- und Multicast-Kommunikationsszenarien können realisiert werden, ohne dass zusätzliche Verfahren eingesetzt werden müssen, die weitere Steuernachrichten zur Adaption erfordern.

Um das vorhandene Broadcastmedium in drahtlosen Sensornetzen auszunutzen, und aufgrund der fehlenden Adresssemantik für codierte Informationen in Paketen, nutzt OCSSIM Broadcast-Kommunikation und Overhearing zwischen den Knoten des Sensornetzes. Eine getroffene Weiterleitungsentscheidung betrifft daher vordringlich die Frage, ob ein Knoten ein vollständiges Paket überhaupt weiterleiten soll oder das Paket verworfen werden soll. Die Auswahl des Nachbarn an den ein Paket weitergeleitet wird ist in kooperativen Sensornetzen häufig weniger entscheidend, da das Paket auf mehreren alternativen Pfaden zugestellt werden kann und Nachbarn proaktiv bei Weiterleitung und Fehlerkorrekturen unterstützen können. Die Folge ist allerdings häufig, dass sich aus globaler bzw. netzwerkweiter Perspektive kein optimales Routingverhalten für Einzelnachrichten einstellt.

Ein weiterer Bestandteil der Weiterleitungsentscheidung ist die Frage, ob alternativ nur einzelne der Nachrichten, die im Paket codiert sind, weitergeleitet werden sollen. Das Verwerfen von Paketen und Nachrichten stellt dabei eine aktive Entscheidung im Weiterleitungsprozess dar, um Flussrichtung und Menge der Pakete und Nachrichten im Netzwerk zu steuern. So kann die eigene Weiterleitungsentscheidung zum Beispiel direkt davon abhängig gemacht werden, wie häufig ein Paket bereits von Nachbarn weitergeleitet wurde, um redundante Informatio-

nen im Netzwerk adaptiv anzupassen. Weitere Strategien und Kernpunkte dieser Entscheidung werden in Kapitel 5.5 diskutiert.

Ein weiterer Vorteil des Kommunikationsschema ist die inhärente Nutzung von *Multi-Path*-Strukturen der zugrunde liegenden Netzwerktopologie.

Sind mehrere disjunkte Pfade im Sensornetz zwischen zwei, an einer Übertragung beteiligten Knoten vorhanden, so werden diese Pfade zur Weiterleitung genutzt, solange die Zwischenknoten auf den disjunkten Pfaden eine positive Weiterleitungsentscheidung treffen. Ein Knoten am Ende eines disjunkten Pfades kann dann redundante Pakete aus den empfangenen Daten filtern. Diese Eigenschaft erhöht die Robustheit der Paketzustellung, insbesondere im Fall von hohen Bitfehler- und Paketverlustraten auf einzelnen Pfaden im Netzwerk, sofern die Netzwerkdichte und Topologie mehrere disjunkte Pfadabschnitte zwischen den Knoten bewirken.

5.2 Simultane Netzwerkcodierung und opportunistische Kommunikation

Opportunistische Weiterleitungsentscheidungen führen in Kombination mit den verwendeten Mechanismen für Netzwerkcodierung zu einer Reihe von Kernproblematiken, die in den folgenden Unterkapiteln analysiert werden. Zu den Kernproblematiken erfolgen im Anschluss bisherige Lösungsansätze, sowie die Diskussion geeigneter Vorgehensweisen zu Behandlung der Problemstellungen im Rahmen des OCSSIM-Kommunikationsschemas.

5.2.1 Initialisierungs- und Sättigungsverhalten

Kommunikation mit Netzwerkcodierung hat die Eigenschaft, dass Nachrichten erst bei Vorliegen einer ausreichend großen Anzahl von innovativen Paketen (siehe Kapitel 3.2.1) mit zuvor unbekannten Nachrichtenkombinationen decodiert werden können. Wenn neue Nachrichten in codierter Form im Netzwerk verteilt werden, führt dies dazu, dass ein erhöhter Anteil von Paketen nicht decodiert werden kann. Eine neue Nachricht, die in ein Paket codiert wird, erschwert das spätere Decodieren und erhöht die Codierungskomplexität. Solange sie nicht auch in Paketen mit anderen Nachrichtenkombination vorhanden ist oder keine atomaren Pakete mit der Nachricht vorliegen, ist das Decodieren eines Paketes daher nicht möglich. Bis zum Eintreffen weiterer Pakete, die das Decodieren ermöglichen, muss das Paket, das die neue Nachricht enthält, folglich zwischengespeichert werden.

Werden im Gegensatz dazu zu viele Nachrichtenkombinationen versendet, obwohl das Decodieren bereits gewährleistet werden kann, so ist neben steigender Ineffizienz auch eine erhöhte Kollisions- und Staugefahr im Netzwerk die Folge.

Initialisierungsverhalten

Das Initialisierungsverhalten der Kombination von Gossiping und Netzwerkcodierung wird in [104] analysiert. Dabei konnte festgestellt werden, dass die Decoding-Problematik zunimmt, wenn ein erheblicher Anteil von versendeten Nachrichtenkombinationen für die Empfänger nicht innovativ ist. Erfolgt die Auswahl

der Nachrichten und die Wahl der Codierungsvektoren etwa auf rein zufälliger Grundlage, so werden oft linear abhängige Nachrichtenkombinationen und Codierungsvektoren in den Paketen versendet, die nicht zum Decoding beitragen. Als Abhilfe wird das aufeinanderfolgende Senden von mehreren Nachrichtenkombinationen an Knoten aus sogenannten *Contact Sets* empfohlen. Dabei befinden sich nur Nachbarknoten in einem Contact Set, die eine vorangegangene Nachrichtenkombination mit der neuen Nachricht erhalten haben. Somit wird sicher gestellt, dass ein Decoding der nachfolgenden Nachrichten möglich ist.

Dabei werden jedoch drei wesentliche Punkte vorausgesetzt, die im allgemeineren Kommunikationsszenario des OCSSIM-Kommunikationsschema nicht gegeben sind:

- Die Paket werden jeweils per Unicast an die Empfänger weitergeleitet, so dass gezielte Empfänger aus den Contact-Sets vorliegen. Die Verteilung von innovativen Nachrichtenkombinationen über unterschiedliche Pfade ist nicht als Unterstützung des Decodierens vorgesehen.

- Nachrichten müssen stets vollständig decodiert vorliegen, bevor sie weiter versendet werden können. Fehlende Nachrichten können dabei zu erheblicher Latenz bei der Verteilung führen.

- Um den nächsten Empfänger auswählen zu können, ist pro Knoten eine Zustandshaltung der empfangenen Nachrichtenkombinationen aller Nachbarknoten erforderlich. Dazu ist es notwendig, dass die Knoten den Empfang von Paketen explizit quittieren.

Sättigungsverhalten

Ein Netzwerk ist dann *gesättigt*, wenn nur noch die gleichen, einzelnen und innovative Nachrichten auf vielen Knoten fehlen und der Netzwerkverkehr vorrangig aus nicht-innovativen Nachrichtenkombinationen besteht. Bei zunehmender Sättigung des Netzwerkes mit Paketen, die zum Decodieren einer bestimmten Nachricht erforderlich ist, wird das Senden von Paketen, die diese Nachricht enthalten, zunehmend ineffizient. In Folge eines übersättigten Netzwerkes können mit erhöhter Wahrscheinlichkeit Kollisionen und Stausituationen auftreten, die es zu vermeiden gilt.

[104] schlägt hier das Einführen eines *variablen Fanouts* zur Regulierung der Paketmenge vor, die an einen Nachbarn gesendet werden soll. Dieser ist abhängig von der Menge der fehlerfrei empfangenen Pakete des jeweiligen Nachbarn, bezieht aber weder die ganze Nachbarschaft ein, noch wird ermittelt, ob ein Decodieren der Nachrichten durch den Empfang von Paketen auf einem anderen Pfad bereits möglich ist. Zudem muss der Fanout für eine gegebene Netzwerktopologie und Paketverlustwahrscheinlichkeit durch Simulation im Vorfeld bestimmt werden, um brauchbare Optimierungen liefern zu können.

Adaptives Verhalten in OCSSIM

OCSSIM sieht Maßnahmen vor, die sowohl das Initialisierung- als auch das Sättigungsverhalten des Netzwerks positiv beeinflussen.

Zunächst wird jedes eingehende Paket heuristisch daraufhin überprüft, welchen Mehrwert es für den jeweiligen Knoten hat und welchen Wert eine Weiterleitung an die Nachbarknoten hat.

OCSSIM sieht dazu eine Reihe von aktiven und passiven Maßnahmen vor, die anhand einer Überprüfung der folgenden Kriterien ausgewählt werden:

Bekannter Ursprungsknoten: Ist ein Knoten Initiator einer neuen Nachricht, so hat er die Möglichkeit diese neue Nachricht entweder dauerhaft oder über eine vorgegebene Anzahl an Hops mit erhöhter Priorität weiterzuleiten. Dabei wird die Nachricht als atomares Paket versendet oder mit sehr geringer Codierungskomplexität mit anderen Nachrichten kombiniert. Ein interner Zähler im Paketkopf kann vorgesehen werden, um die Anzahl der Hops festzulegen, mit der die Nachricht priorisiert versendet wird, bis eine Kombination mit anderen Nachrichten erfolgt. Somit wird eine initiale Verteilung der neuen Nachricht gesichert, bis ausreichend viele Knoten zum Verteilen und Decodieren mit Paketen aus Nachrichtenkombinationen beitragen können, die auch die neue Nachricht enthalten. Für die priorisierte Weiterleitung sieht OCSSIM das *Fast-Forwading*-Verfahren vor (siehe Kapitel 5.4.2).

Codierungskomplexität: Um so mehr Nachrichten gemeinsam in einem Paket codiert versendet werden, um so eher ist dies als Merkmal zu werten, dass sich ein Nachbarknoten in gesättigtem Zustand befindet. Gleichzeitig steigt die Wahrscheinlichkeit, dass das Paket auch nicht-innovative Nachrichten beinhaltet. Deshalb ist vor der Weiterleitung ein Decodieren der Nachrichten zweckmäßig.

Abhilfe kann das Verteilen der Nachrichten über mehrere Pfade und Nachbarn bringen. Jeder Nachbar fällt dabei eine eigenständige Weiterleitungsentscheidung und erzeugt ggf. eine Neukombination der Nachrichten, die gemeinsam in einem Paket versendet werden (siehe auch Kapitel 5.4.3). Ein Empfängerknoten, der eine Nachricht von zwei unterschiedlichen Nachbarn, also über unterschiedliche Pfade erhält, bekommt die Nachricht daher selten als Bestandteil des gleichen Paketes. Diese Paketvielfalt erlaubt es dem Knoten im Sättigungsfall auch dann noch innovative Nachrichten decodieren zu können, wenn eine Vielzahl der Nachrichten im Paket nicht mehr innovativ sind.

Kenntnis des Paketes oder seiner enthaltenen Einzelnachrichten: Wird ein bekanntes Paket empfangen, so sieht OCSSIM je nach Situation das sofortige Verwerfen des Paketes beim Empfang vor. Ein Paket wird genau dann verworfen, wenn ein Knoten das Paket bereits selbst gesendet hat oder wenn der Knoten das Senden bei einer ausreichend großen Anzahl Nachbarn thr_{drop} beobachten konnte. Als zweckmäßige Vorgehensweise hat sich das Verwerfen

von Paketen, die von mindestens drei Nachbarn gehört wurden, etabliert [82]. Dabei wird die Abdeckung des Netzwerkes bei der Nachrichtenverteilung nur in Ausnahmefällen gefährdet (siehe Kapitel 5.5.1).

Beim erstmaligen Decodieren einer innovativen Nachricht wird diese gemeinsam mit anderen Nachrichten kombiniert und weitergeleitet. Werden bekannte Nachrichten aber wiederholt decodiert und sind somit nicht-innovativ, dann sieht OCSSIM das Unterdrücken der Weiterleitung für die Nachricht vor. Dies vermeidet, dass die Nachricht als Bestandteil eines erneuten Paketes versendet wird und somit die potentiell gesättigten Nachbarknoten mit nicht-innovativen Nachrichtenkombinationen versorgt.

5.2.2 Empfangsbestätigungen bei zuverlässiger Kommunikation

Die Nutzung opportunistischer Kommunikation mittels auf Broadcast und auf Multicast basierender Flooding- und Gossiping-Verfahren ist mit Blick auf zuverlässige Kommunikation problematisch. Protokolle, die diese Verfahren nutzen, können oft nur probabilistische Zuverlässigkeit gewährleisten, da auf Empfangsbestätigungen zu Gunsten der Effizienz verzichtet werden muss. Statt eine garantierte Nachrichtenabdeckung des Netzwerkes durch zuverlässige Verteilung der Nachrichten zu erreichen, wird versucht, die Wahrscheinlichkeit einer vollständigen Abdeckung auf einem möglichst hohen Niveau zu halten. Mögliche Paketverluste und Übertragungsfehler werden dabei durch redundantes Empfangen der gleichen Nachricht von mehreren Nachbarknoten ausgeglichen. Bevorzugt an Randbereichen des Netzwerks oder bei Bereichen mit geringer Knotendichte kann es aber zu Lücken in der Nachrichtenabdeckung kommen.

Grund für den Verzicht von Empfangsbestätigungen sind die als *Broadcast Storm* (vgl. [82]) und *ACK-Explosion* bekannten Problematiken in drahtlosen Netzwerken.

Zunächst wird hier die Broadcast-Storm-Problematik diskutiert, anschließend die ergänzende Problematik bei Einführung von bestätigter Kommunikation und Verwendung von Netzwerkcodierung. Abschließend werden in diesem Abschnitt existierende Lösungsansätze genannt und das Verfahren impliziter Bestätigungen vorgestellt, das in OCSSIM Verwendung findet.

Wird eine Nachricht beim Flooding-Verfahren an die Nachbarknoten gesendet, so sind diese aufgefordert, die Nachricht an benachbarte Knoten mittels eigenem Broadcast weiterzuleiten. Dabei wird üblicher Weise jede Nachricht pro Knoten nur ein einziges Mal weitergeleitet, um zyklisches Weiterleiten zwischen den Knoten zu vermeiden. Dazu muss jede Nachricht eindeutig anhand eines Identifikators erkannt werden können. Der Identifikator muss lokal gespeichert werden, bis das Fluten im Netzwerk abgeschlossen ist. Ansonsten könnten Weiterleitungsschleifen zwischen beteiligten Knoten entstehen. Diese Vorgehensweise führt dazu, dass bei einem Netzwerk der Größe n insgesamt n Broadcasting- und Speichervorgänge benötigt werden, sowie bis zu $n(n-1)$ Nachrichten von den Knoten des Netzwerks empfangen werden müssen.

Das Weiterleiten per Broadcast ist üblicherweise mit weiteren Teilproblematiken verbunden. Insbesondere bei Kombination von Flooding und Medienzugriff per CSMA/CA ist ohne RTS/CTS mit einer Erhöhung der Kollisionswahrscheinlichkeit zu rechnen, da es zu einem Synchronisationseffekt der weitergeleiteten Nachrichten zwischen benachbarten Knoten kommt. Nach dem Empfang einer weiterzuleitenden Broadcast-Nachricht eines Knotens X, benötigt der Protokollstack der zu X benachbarten Knoten N_1 und N_2 in etwa die gleiche Verarbeitungszeit, bis der Sendevorgang auf den Nachbarknoten beginnen kann. Gerade in Netzwerken hoher Dichte ist der Wettstreit der Knoten um die Weiterleitung besonders auffällig. Selbst wenn sich die Knoten nicht in gegenseitiger Sendereichweite befinden, trägt das Problem versteckter Endgeräte dann zu massiven Kollisionen bei. Zur Vermeidung des Problems ist beim Einsatz von Flooding-Verfahren die Verwendung einer zufälligen *Rückstellzeit* (Back-Off) notwendig, um den Zeitpunkt der Weiterleitung stärker zu streuen und den Medienzugriff zu optimieren.

Während eine zeitliche Streuung des Sendezeitpunktes von weitergeleiteten Nachrichten auf jedem Knoten der Nachrichtenübertragung zusätzliche Latenz hinzufügt, kann mit zufälligem Back-Off jedoch auch eine weitere Problematik der Flooding-Verfahren gelöst werden: Da jeder Knoten im Allgemeinen die gleiche Nachricht von unterschiedlichen Nachbarn mehrfach hört, ist eine hohe Ineffizienz mit dem Nachrichtenempfang verbunden, da die Redundanz nicht zum Informationsgewinn beiträgt. Wird während der Back-Off-Zeit eine identische Nachricht von einem anderen Nachbarknoten empfangen, so sinkt sowohl die Wahrscheinlichkeit, dass ein eigener Nachbar durch die weiterzuleitende Nachricht einen Informationsgewinn erhält, als auch der noch abzudeckende Bereich in Sendereichweite, indem die Nachricht zuvor nicht empfangen werden konnte. Wurde die weiterzuleitende Nachricht innerhalb der Back-Off-Zeit von allen Nachbarn empfangen, kann die geplante Weiterleitung vollständig eingestellt werden, da kein Informationsgewinn zu erzielen ist. [82] zeigt hier, dass sich der abgedeckte Bereich durch das Weiterleiten von Broadcast-Nachrichten nur noch um 0,05% erhöht, sobald die Nachricht zuvor von $k > 3$ Nachbarn empfangen wurde.

Aus Effizienzgründen sollte daher in den meisten Netzwerken schon bei Erreichen einer Schwelle von drei identischen empfangenen Nachrichten auf das Weiterleiten weiterer identischer Nachrichten verzichtet werden. Sofern keine sehr ungleichmäßige Verteilung der Knotenpositionen vorherrscht, wird der nicht abgedeckte Bereich durch Verzicht auf Weiterleitung später häufig von anderen Knoten abgedeckt, die selber eine größere Back-Off-Zeit haben oder erst nach zusätzlichen Weiterleitungsschritten erreicht wurden.

Die Einführung von Bestätigungen erfolgreich empfangener Nachrichten (*ACKs*, *Acknowledgements*) verschärft die oben genannte Broadcast Storm Problematik. Empfängt ein Knoten nun eine Nachricht per Broadcast, so muss diese nicht nur weitergeleitet werden, sondern auch eine Bestätigungsnachricht an den Sender zurück gesendet werden. Daraufhin ist zu unterscheiden, ob eine Fehlerbehandlung und Übertragungswiederholung im Netzwerk mittels ACK auf Basis einer Ende-zu-Ende Kommunikation oder pro Hop durchgeführt wird.

Bei Durchführung auf Basis der Ende-zu-Ende Kommunikation explodiert die Anzahl der benötigten Nachrichten im Netzwerk förmlich (*ACK-Explosion*), da jede Bestätigung eines Endsystems als neue weiterzuleitende Nachricht von jedem Knoten behandelt wird. Die Komplexität der zu sendenden Pakete erreicht $\mathcal{O}(n^2)$ und die zu empfangenen Pakete sind in $\mathcal{O}(n^4)$. Somit kann effiziente Skalierbarkeit des Netzwerks nicht gewährleistet werden. Außerdem ist ein drastischer Anstieg an Kollisionen und Stausituationen zu beobachten.

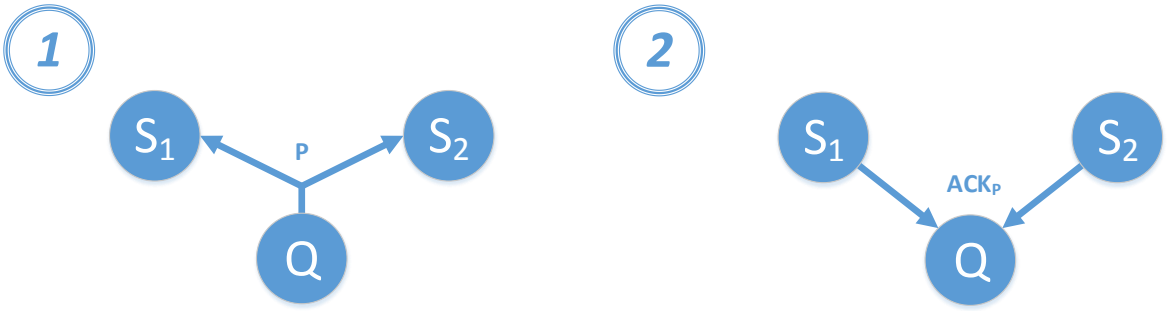

Abbildung 5.1 Versteckte Endgeräte bei Broadcasting mit ACKs

Abbildung 5.1 zeigt die Ausgangssituation mit versteckten Endgeräten, wenn Knoten Q in Schritt ① ein bestätigtes Paket an seine Nachbarknoten S_1 und $S-2$ sendet. Sind S_1 und S_2 nicht in gegenseitiger Empfangsreichweite, so können sie das explizite ACK des jeweils anderen Knotens in Schritt ② nicht erkennen. Das führt nicht nur dazu, dass jeder Knoten eine eigene Quittung senden muss, sondern führt ohne Erweiterung durch RTS/CTS bei gleichzeitigem Senden auch zur Kollision der ACKs bei Knoten Q.

In drahtlosen Sensornetzen ist mit einer hohen Paketfehlerwahrscheinlichkeit auf einzelnen Knotenverbindungen auszugehen [4]. Deshalb sind Mechanismen zur Fehlerbehandlung und Übertragungswiederholung die auf einzelnen Hops arbeiten im Allgemeinen Ende-zu-Ende-Verfahren vorzuziehen.

Dennoch sind bei einer Paketfehlerwahrscheinlichkeit p_i auf der Verbindung zu Nachbar N_i und insgesamt k Nachbarn mit einer Wahrscheinlichkeit von

$$P(SRep) = 1 - ((1 - p_1)...(1 - p_k))^2$$

eine Sendewiederholung zu erwarten. Dabei gilt die Annahme, dass auch eine Empfangsbestätigung trotz erfolgreicher Übertragung mit der Wahrscheinlichkeit p_i von Fehlern auf der Verbindung betroffen sein kann. Abbildung 5.2 visualisiert die Wahrscheinlichkeit für Sendewiederholungen für verschiedene Nachbarschaftsgrößen und bei identischer Paketfehlerwahrscheinlichkeit für alle Verbindungen zu Nachbarknoten.

Um so mehr Verbindungen ein Knoten zu Nachbarknoten hat, und um so höher die Wahrscheinlichkeit für einen einzelnen Paketfehler ist, um so eher ist auch

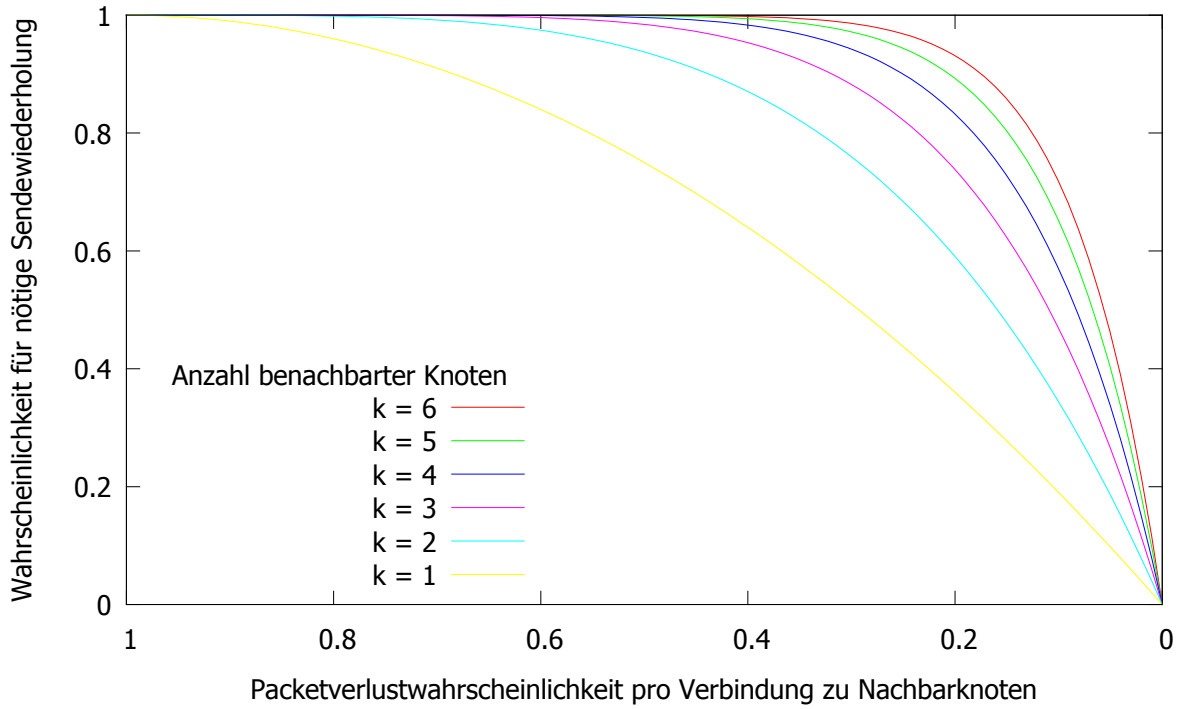

Abbildung 5.2 Sendewiederholungen bei Flooding mit Empfangsbestätigungen per Hop bei Nachbarschaftsgröße k

mit Sendewiederholungen zu rechnen. Schon bei einer Paketfehlerrate von 10% und 3 benachbarten Knoten ist für einen quittierten Broadcast über einen Hop in über 50% der Fälle eine Sendewiederholung nötig. Ebenso wie im Ende-zu-Ende-Szenario ist daher bei zunehmender Netzwerkdichte und -größe mit einer deutlich erhöhten Anzahl von Kollisionen und Stausituationen auszugehen, was keine skalierbare Übertragung mittels quittiertem Broadcast zulässt.

5.2.2.1 Empfangsbestätigungen von codierten Nachrichten

Wird Netzwerkcodierung in Kombination mit einem Verfahren zur Sendewiederholung (Automatic Repeat Request, ARQ) eingesetzt, stellt sich das Problem, wie der Empfang eines Paketes mit codierten Nachrichten zweckmäßigerweise quittiert werden kann und welche Mechanismen den Verlust eines Paketes effizient behandeln, um die enthaltene Information zu rekonstruieren. Prinzipiell können zwei elementare Vorgehensweisen unterschieden werden:

- Quittierung von ganzen Paketen ACK_P mittels eindeutigem Identifikator, z. B. anhand einer Sequenznummer und Kennung des Senders
- Einzelquittierung von erhaltenen Nachrichten ACK_{M_i}

Vor der Weiterleitung der empfangenen Pakete erfolgen auf den weiterleitenden Zwischenknoten im Allgemeinen einige Verarbeitungsschritte, die dafür sorgen, dass das Quittieren von Paketen nicht als Bestätigung für eine erfolgte Weiterleitung ausreicht. Die weiterleitenden Zwischenknoten decodieren in Abhängigkeit

vom ausgewählten Codierungsschema zunächst die enthaltenen Nachrichten des Paketes. Anschließend erfolgt eine Neucodierung, die dafür Sorgen kann, dass die enthaltene Nachrichtenkombination verändert wird. Erst dann erfolgt das Weiterleiten des neuen Paketes.

Zudem ist anhand der Quittung nicht erkennbar, ob der Empfänger Informationen aus dem empfangenen Paket erfolgreich decodieren und interpretieren konnte. Auf der Seite des Senders muss für Sendewiederholungen zusätzlicher Speicher vorgesehen werden, in dem die Kombination der codierten Nachrichten eines Paketes und der dazugehörige Codierungsvektor $C_1M_1 \oplus ... \oplus C_kM_k$ bzw. das vollständige Paket abgelegt wird. Übermittelte Nachrichten können erst lokal verworfen werden wenn sicher gestellt ist, dass diese in keinen unquittierten Paketen enthalten sind.

Im Gegensatz dazu erlaubt die Quittierung von Einzelnachrichten das unabhängige Verwerfen einzelner Nachrichten beim Sender. Je nach Quittierungszeitpunkt des Empfängers können die folgenden Vorgehensweisen für Einzelnachrichten unterschieden werden. Für den Sender signalisiert eine eingehende Quittung in diesem Kontext immer den Zeitpunkt, wann eine Nachricht für Sendewiederholungen aus Sicht dieses Empfängers nicht länger relevant ist.

Drop-when-received: Die Quittung wird vom Empfänger verschickt sobald die Nachricht eintrifft. Anhand der Informationen des Codierungsvektors, Sequenznummer und Identifikation des Senders wird die Einzelnachricht eindeutig identifizier- und quittierbar. Eine Drop-when-received-Quittung signalisiert jedoch nicht, dass der Nachrichteninhalt vom Empfängerknoten decodiert und interpretiert werden konnte.

Drop-when-decoded: Bei dieser Vorgehensweise wird die Quittung erst nach erfolgreichem Decoding des Empfängers ausgelöst. Die Identifikation erfolgt wie im Fall von Drop-when-received. Während hier zwar sicher gestellt ist, dass der Nachrichteninhalt dem Empfänger in verwertbarer Form vorliegt, kann sich der Sendezeitpunkt der Quittung als problematisch erweisen. Solange nicht ausreichend lokale Informationen zum Decodieren vorliegen, wird das Senden der Quittung hinausgezögert. Ist dem Empfänger eine Nachrichtenkombination bereits bekannt, also nicht-innovativ, so dass kein Decoding-Prozess stattfindet, wird auch keine erneute Quittung versendet.

Drop-when-seen: Die Drop-when-seen"Vorgehensweise (vgl. [100]) basiert auf der Annahme, statt unabhängiger, einzelner Nachrichten einen zusammenhängenden Nachrichtenstrom an eine definierte Gruppe von Knoten zu senden. Ein Knoten quittiert den Empfang einer Nachricht erst, sobald zum Decodieren der Nachricht nur noch Nachrichten benötigt werden, die in der Reihenfolge des Nachrichtenstroms hinter der empfangenen Nachricht folgen. Diese Vorgehensweise hat den Vorteil, dass ein Decodieren der Nachricht im weiteren zeitlichen Verlauf sichergestellt werden kann und der Sender bei Sendewiederholungen nur Nachrichten verwendet, die für mindestens einen

Nachbarn innovativ sind. Für den Einzeltransport von Nachrichten ist diese Vorgehensweise jedoch nicht geeignet.

ARQ-Schemata die das ein Einbeziehen von Nachbarknoten zur Sendewiederholung vorsehen, wie z. B. [109], können bei gleichzeitiger Nutzung der Netzwerkcodierung von quittierten Einzelnachrichten profitieren. Voraussetzung dafür ist, dass die fehlenden Nachrichten von den Nachbarknoten identifiziert werden können.

Im Fall von Kommunikation mittels Nachrichtenströmen statt Einzelnachrichten, kann ein Knoten dazu mit expliziten Negativquittung (*Negative Acknowledgement*, NACK) reagieren.

Sobald das Ausbleiben von Nachrichten festgestellt wird, die regulär auf bereits empfangene Nachrichten folgen sollten, wird dann per NACK die Sendewiederholung von Nachbarknoten angefordert. NACKs erweisen sich in der Broadcast-Kommunikation als besonders effizient, da sie nur noch im irregulären Fehlerfall erforderlich sind. Nachbarknoten können aber keine Maßnahmen zur Unterstützung zur Wiederherstellung verlorener Nachrichten oder des Decodierens anbieten, solange nicht ein eindeutiger Identifikator der im Paket enthaltenen Nachrichten quittiert wird. Zur Sendewiederholung durch die Nachbarn können schließlich auch andere Codierungsvektoren oder Paketkombinationen genutzt werden, als in der Ursprungsnachricht vorhanden waren. Somit kann eine Sendewiederholung in Doppelfunktion genutzt werden, um neben der Wiederherstellung fehlender Nachrichten auch innovative Nachrichtenkombinationen für andere Nachbarn zu verteilen.

Das *Collective Space Problem* [56] tritt dabei in einem Szenario mit gerichteter Weiterleitung von Nachrichtenströmen auf. Blockweise werden mehrere codierte Pakete aus den innovativen Nachrichten an eine Gruppe von Nachfolgerknoten gesendet. Die Pakete enthalten die gleichen Nachrichten, sind aber mit unterschiedlichem Codierungsvektor codiert. Gemeinsam empfangen die Nachfolgerknoten alle versendeten Pakete, jedoch jeder einzelne Nachfolgerknoten eine unterschiedliche Kombination. Somit kann kein Nachfolger den Erhalt aller Pakete an eine Quelle bestätigen. Mittels einer Reihe vorberechneter und bereits im gesamten Netzwerk verteilter *Hash-Matrizen* können die Codierungsvektoren der empfangenen Pakete so kombiniert werden, dass eine *kumulativ codierte Empfangsbestätigung (CCACK)* durch jeden Nachfolger erstellt werden kann. Die CCACKs werden an die Quelle des Nachrichtenstroms zurückgesendet. Dort kann festgestellt werden, ob Sendewiederholung nötig sind oder ob die Gruppe der Nachfolgeknoten gemeinsam alle innovativen Nachrichtenkombinationen zum Decodieren erhalten haben.

5.2.2.2 Kumulatives Zusammenfassen von Empfangsbestätigungen

Eine Möglichkeit zur Senkung des Nachrichtenverkehrs durch Einzelquittung bietet das Zusammenfassen zu kumulierten Bestätigungen (*Cumulative Acknowledgements*, CACKs) [49]. Im Zusammenspiel mit Netzwerkcodierung bietet sich darüber hinaus auch eine Reduktion der Nachrichtenanzahl durch Codieren von Quittungen und gleichzeitiges Bündeln an.

Durch Senden eines *Nullspace-Vektors* als komprimierte Quittung, kann nachgeprüft werden, ob ein Nachbar weitere Pakete mit einer innovativen Nachrichtenkombination erhalten hat [84]. Dazu wird ein Vektor z ermittelt für den das innere Produkt Null ergibt: $z \cdot C = 0$. Der Nullspace-Vektor wird beim Weiterleiten den regulären Nachrichten mittels sogenanntem *Piggybacking* hinzugefügt. Der Sendung kann durch Mithören der weitergeleiteten Nachricht und Bilden des inneren Produktes über z und die Ursprungspakete erkennen, ob dem Empfänger weitere innovativen Pakete fehlen. Dies ist genau dann der Fall, wenn das innere Produkt ungleich Null ist [83].

Mit einem Nullspace-Vektor werden in Form eines Codierungsvektors alle Pakete, die zum Decoding der gleichen Nachricht benötigt werden, durch den Empfängerknoten auf einmal bestätigt. Zum Beispiel könnte der Empfängerknoten die Codierungsvektoren $C_1 = (3, 2, 12)$ und $C_2 = (3, -1, 3)$ erhalten haben. Diese können mit dem gemeinsamen Nullspace-Vektor $z = (2, 3, -1)$ bestätigt werden.

Allerdings bringt die Vorgehensweise mit Nullspace-Vektoren auch einige Nachteile mit sich. Der Vektor $C_3 = (1, 5, 17)$ kann durch z bestätigt werden, ist aber nicht innovativ, da er linear abhängig von den schon zuvor erhaltenen Codierungsvektoren C_1 und C_2 ist. Es kann in Folge zu False-Positive Signalisierung durch den Nullspace-Vektor kommen, bei der ein Sender den Datentransport fälschlicherweise einstellt, weil z nahelegt, dass alle innovativen Nachrichten übermittelt wurden [56]. Umgekehrt sorgt das Collective Space Problem dafür, dass ein Sender in manchen Fällen kontinuierlich weiter sendet, da sich nicht erkennen lässt, dass alle innovativen Nachrichten auf die Nachfolgeknoten aufgeteilt vorhanden sind.

Eine Lösung des Collective Space Problems mittels Nullspace-Vektoren präsentiert schließlich [56] durch die Einführung von *Cumulative Coded Acknowledgements* (CCACKs). Bei diesem Verfahren muss der Sender jedoch sämtliche Codierungsvektoren aus Paketen von eingegangenen, weitergeleiteten und decodierten Nachrichten protokollieren. Dazu werden drei separate Pufferspeicher verwendet: Ein Empfangspuffer $\mathcal{B}_{RX}$, ein Ausgangspuffer $\mathcal{B}_{TX}$ und ein Puffer für die lokalen innovativen Nachrichten $\mathcal{B}_{Inf}$. Sobald eine Nullvektor-Quittung eintrifft werden die Codierungsvektoren in den Puffern $\mathcal{B}_{RX}$ und $\mathcal{B}_{TX}$ als *gehört* markiert, deren Kombination als inneres Produkt Null ergibt. Der Sender sendet so lange weitere innovative Nachrichtenkombinationen an nachfolgende Knoten, bis $rang(\mathcal{B}_{Inf})$ und der Rang der Matrix aus den gehörten Codierungsvektoren aus $\mathcal{B}_{RX} \cup \mathcal{B}_{TX}$ identisch sind. Während dieser Ansatz für Nachrichtenströme das Broadcast-Medium in geeigneter Form und auf effiziente Weise ausnutzen kann, muss jedoch für Einzelnachrichten und in Netzen, in denen die Topologie keine eindeutige Vorgänger-Nachfolger-Relation zwischen den Knoten definiert, eine weniger effiziente Lösung gefunden werden. Trotz der effizienten Kommunikation ist bei diesem Ansatz auch die zusätzliche Rechenleistung und Speichermenge nicht zu vernachlässigen, die zur Berechnung der inneren Produkte auf dem verwendeten Galois-Feld und zum Protokollieren und Zwischenspeichern der Codierungsvektoren benötigt wird. Die Autoren konzipieren daher die Nutzung von CCACK für die leistungsstärkere Geräteklasse der *Wireless Mesh Networks*, an Stelle von drahtlosen Sensornetzen.

5.2.2.3 Implizite Empfangsbestätigungen

In Broadcastnetzen, die das explizite Mithören fremder Nachrichten (sogenanntes *Overhearing*) für die Kommunikation einsetzen, kann die Quittierung der versendeten Pakete und Nachrichten auch implizit erfolgen, ohne dass dafür zusätzliche Sendevorgänge in Gegenrichtung notwendig sind. Eine Reihe von Forschungsarbeiten in diesem Feld zeigt dabei die möglichen Variantenvielfalt bei Weiterleitungs- und Routingszenarien (vgl. [57, 91, 117, 124]).

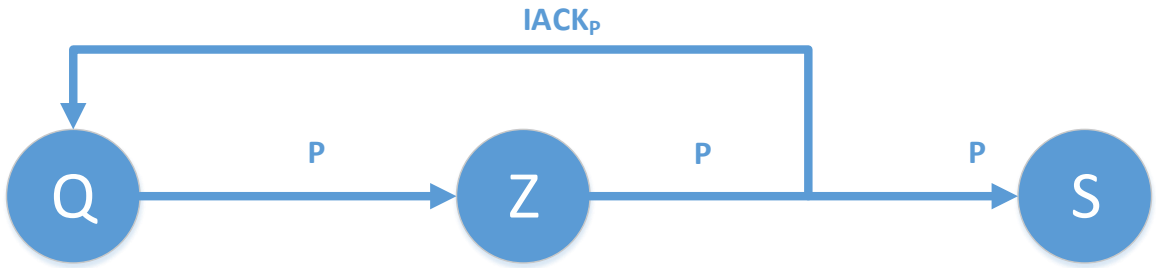

Abbildung 5.3 Grundidee impliziter Bestätigung von empfangenen Paketen

Abbildung 5.3 stellt die Grundidee von impliziten Bestätigungen dar. Ausgehend von dem Senderknoten Q wird das Paket P über den Zwischenknoten Z an den Knoten S weitergeleitet. Nach dem Senden von P hört Knoten Q dabei weiterhin bei der Kommunikation seines Nachbarn Z mit. Z empfängt das Paket und leitet es unmittelbar weiter, ohne eine Sendebestätigung für P zurückzusenden. Da die Kommunikation per Broadcast erfolgt, hört Q die Weiterleitung von P ebenso wie der Zielknoten S, ohne dass sich die Knoten Q und S in gegenseitiger Sendereichweite befinden. Beim erneuten Empfang von P bei Knoten Q kann dieser durch Vergleich mit seinem lokalen Sendepuffer feststellen, dass er dieses Paket zuvor selbst versendet hat. Der erneute Eingang von P kann also als implizite Empfangsbestätigung $IACK_P$ des Knotens Z interpretiert werden und das erneut eingehende Paket kann direkt verworfen werden. Bleibt die implizite Empfangsbestätigung aus, so sendet Knoten Q nach Ablauf eines voreingestellten *Retransmission Timers* t_{ret} unaufgefordert eine Sendewiederholung.

Implizite Bestätigungen haben jedoch wesentliche Einschränkungen in zwei Situationen:

1. Keine Bestätigungen durch Zielknoten: Auf dem letzten Hop zum Zielknoten fällt ohne zusätzliche Empfangsbestätigung mittels explizitem ACK keine Bestätigung an, da keine Weiterleitung durch den Zielknoten erfolgt.

2. Verhalten bei mehreren weiterleitenden Nachbarn: Gibt es mehrere weiterleitende Nachbarn $Z_1, ..., Z_k$, so signalisiert eine implizite Bestätigung nur, dass ein einzelner Nachbar das Paket empfangen hat. Für die Bestätigung aller Nachbarn muss entsprechend auf den Empfang aller impliziten Bestätigungen gewartet werden. Dennoch kann auch das Warten auf die ersten

i impliziten Bestätigungen ein gewünschtes Verhalten darstellen. So kann etwa die Weiterleitung sichergestellt werden, ohne den Netzwerkverkehr mit redundanten Sendewiederholungen zu belasten.

5.2.2.4 Empfangsbestätigungen in OCSSIM

OCSSIM setzt zur Kommunikation das implizite Quittieren der Nachrichten mittels Overhearing ein, ohne dass dafür eigenständige Empfangsbestätigungen durch den Empfänger gesendet werden müssen.

Dazu wird hier erklärt, wie die im vorherigen Abschnitt angesprochenen Situationen in OCSSIM behandelt werden. Quell- und Senkenknoten, die als adressierter Zielknoten einer im Paket codierten Nachricht in Frage kommen, leiten eine Nachricht und ggf. auch das unveränderte Paket dennoch weiter. Dies hat folgende Ursachen:

- Die Verteilung von Nachrichten an adressierte Netzwerkbereiche sollte nicht bereits durch das Einstellen der Weiterleitung beim ersten adressierten Empfängerknoten unterbrochen werden, der die Nachricht decodieren kann.
- Ein Paket kann üblicherweise nicht nur die codierten Nachrichten enthalten, die an den jeweiligen Zielknoten gerichtet sind, sondern auch weitere mit entgegengesetzter Übertragungsrichtung oder wird durch eine codierte Rückantwortnachricht ergänzt.
- Bei sehr dynamischer Netzwerktopologie kann die Anzahl der Nachbarn stark schwanken, so dass weitere Knoten seit der letzten ausgehenden Nachricht in der Nachbarschaft hinzugekommen sind. Dies kann mittels Weiterleitung durch den Zielknoten ermittelt werden, da eine implizite Bestätigung von diesen Knoten zu erwarten ist.

Für die implizite Paketquittung $IACK_P$ oder Nachrichtenquittung $IACK_M$ benötigt OCSSIM jeweils nur das erste beobachtete Paket bzw. die Nachricht die von einem Nachbarn weitergeleitet wurde.

Begründet ist dies zum Einen in der allgemein hohen Erreichbarkeit der Nachbarschaft über redundante Pfade durch die zugrunde liegende Multi-Path-Topologie bei ausreichender Netzwerkdichte. Hierbei ist abzuwägen zwischen dem Nutzen durch Sendewiederholung und ineffizienten Sendewiederholungen, die zumindest für einen Teil der Nachbarknoten redundante Informationen enthalten. Ein Nachbar mit fehlender implizierter Sendequittung kann dabei im weiteren Verlauf von anderen Nachbarn – insbesondere auch durch den Nachbarn mit zuerst empfangener impliziter Bestätigung – mit den fehlenden Informationen versorgt werden.

Zum Anderen ist dem Senderknoten beim Warten auf eine implizite Bestätigung nicht unmittelbar bekannt, wie groß die eigene Nachbarschaft ist. Demnach kann auch die zu erwartende Anzahl impliziter Bestätigungen durch den Sender nicht eindeutig ermittelt werden. Von hohen Änderungsraten der Nachbarschaft durch

die dynamische Netztopologie ist auszugehen. Zudem findet in OCSSIM kein periodischer Austausch von Netzwerkinformationen oder *Neighbor Discovery* statt.

Des Weiteren muss zur impliziten Bestätigung von Paketen und Nachrichten berücksichtigt werden, dass die Zwischenknoten in OCSSIM die Möglichkeit haben, die codierten Nachrichten eines Paketes per Reassembling neu kombinieren zu können. Dabei können nach der Neukombination zwar noch die einzelnen enthaltenen Nachrichten anhand ihres individuellen Codierungsvektors eindeutig bestimmt werden, nicht jedoch das Ursprungspaket des vorhergehenden Senders. Zur schnellen Identifikation unverändert weitergeleiteter Pakete kann aber die Kombination der einzelnen enthaltenen Codierungsvektoren herangezogen werden. Diese werden zu diesem Zweck in aufsteigend sortierter Reihenfolge im Paket übertragen.

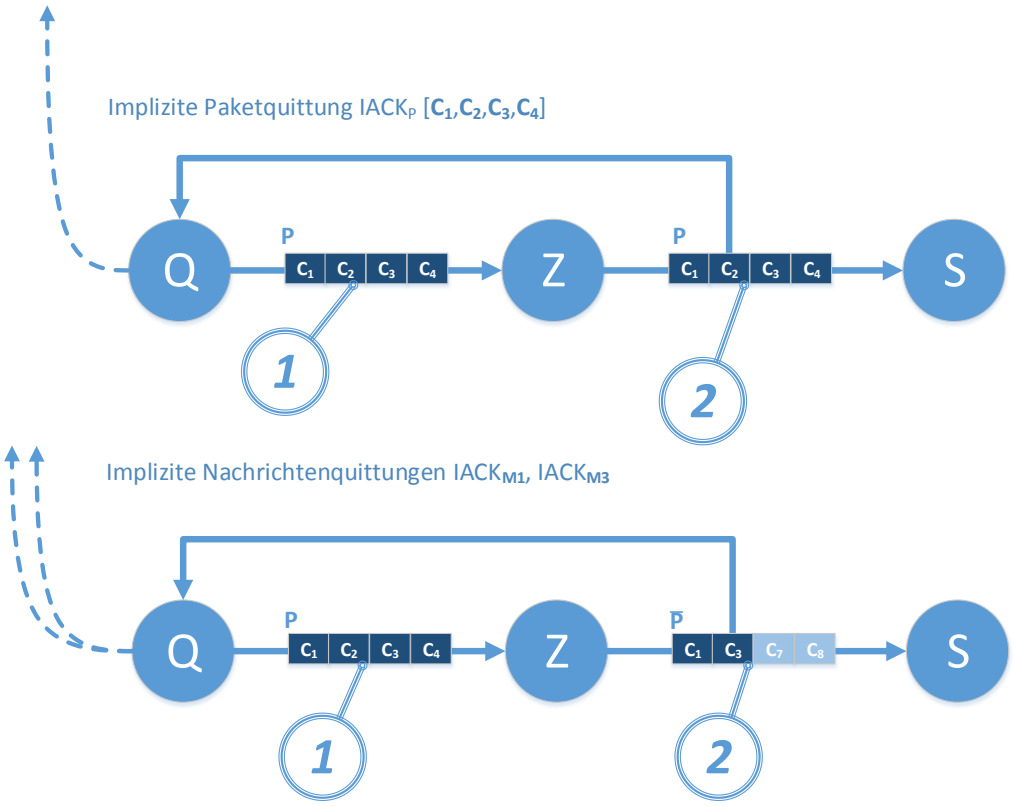

Abbildung 5.4 Implizite Bestätigung von Paketen und Nachrichten in OCSSIM

OCSSIM sieht sowohl die implizite Quittierung von Nachrichten als auch die implizite Quittierung von Paketen vor, sofern sie unverändert von einem Nachbarknoten weitergeleitet wurden. Beide Quittungen werden von einem Knoten jedoch unterschiedlich interpretiert. Abbildung 5.4 zeigt den beispielhaften Ablauf der impliziten Bestätigung für beide Interpretationsweisen. In Schritt 1 wird zunächst das Paket P vom Sender Q verschickt, welches die Nachrichten mit dem Codierungsvektor $C_1, ..., C_4$ enthält. Im oberen Fall wird anschließend die Nach-

richt vom Zwischenknoten Z unverändert an S weitergeleitet. Da Q ebenfalls die Kommunikation mithören kann, wird das bekannte Paket als *implizite Paketquittung* $IACK_P$ für das Paket P interpretiert. Dagegen wird im unteren Fall P durch Neukombination mit lokal bei Z gespeicherten Nachrichten zu $\bar{P}$ transformiert und anschließend weitergeleitet. Aus Sicht von Q kann nun kein identisches Paket mehr auf dem Medium festgestellt werden. Es kann aber weiterhin anhand der Codierungsvektoren C_1 und C_2 beobachtet werden, dass in $\bar{P}$ die codierten Nachrichten M_1 und M_2 enthalten sind. Der Knoten Q interpretiert $\bar{P}$ also als *implizite Nachrichtenquittungen* $IACK_{M_1}$ und $IACK_{M_2}$.

Falschinformation zur Sendewiederholung

Nun wird einfaches Szenario mit drei Knoten A, B und C betrachtet. Knoten A sendet ein Paket an Knoten B, der wiederum das Paket an Knoten C weiterleitet. In Abhängigkeit der verwendeten impliziten Quittierungsmethode können bei der Sendewiederholung zwei Fehlersituationen auftreten.

Im ersten Fall wird die implizite Quittierung des Paketes verwendet. Bleibt die implizite Quittung eines Paketes durch Knoten B aus, so erfolgt nach Ablauf eines *Retransmission Timers* t_{ret} die Sendewiederholung durch Knoten A. Wenn das Paket durch B unverändert weitergeleitet wurde, ist eine Sendewiederholung nach Ablauf von t_{ret} dennoch sinnvoll, da Knoten A davon ausgehen muss, dass das Paket vom weiterleitenden Knoten B nicht empfangen wurde.

Werden die Nachrichten dagegen von Knoten B vor dem Weiterleiten neu kombiniert, so erfolgt statt der impliziten Quittierung des Paketes die implizite Nachrichtenquittierung. Obwohl das Paket erfolgreich von B empfangen wurde, erfolgt in diesem Fall fälschlich eine Sendewiederholung durch Knoten A, da dieser auf die implizite Quittierung des Paketes durch B wartet.

In Fall Zwei wird stattdessen die implizite Quittierung von einzelnen Nachrichten verwendet um den Empfang zu bestätigen. Wenn die Sendewiederholung auf dieser Grundlage erfolgt, kann eine Sendewiederholung fälschlicherweise unterlassen werden.

Dies geschieht genau dann, wenn eine Einzelnachricht schon mittels Paketen von Dritten (Knoten D) auf dem weiterleitenden Knoten B decodiert werden konnte, obwohl das von A an C weiterzuleitende Paket nicht empfangen wurde. Das tatsächlich durch B weitergeleitete Paket kann daher von Knoten D stammen und sich vom Paket von Knoten A unterscheiden, auch wenn beide Pakete die gleiche quittierte Nachricht enthalten. Die implizite Quittung von Knoten B steht dann in keinem ursächlichen Zusammenhang mit dem Paket von Knoten A. Somit kann das Paket von Knoten A tatsächlich niemals Knoten B erreicht haben. Dennoch erfolgt hier die implizite Nachrichtenquittierung an Knoten A.

Zur Lösung des Dilemmas der Sendewiederholung setzt OCSSIM deshalb ein heuristisches Verfahren ein:

Im Paketkopf wird vom jeweils weiterleitenden Knoten vermerkt, ob das Paket unverändert weitergeleitet wird oder ob die enthaltenen Nachrichten neu kombiniert werden.

Sendet Knoten A nun ein Paket P, dass von Knoten B als $\bar{P}$ weitergeleitet wird, dann dient $\bar{P}$ also auch als implizite Quittung für P bei Empfang durch A. Wurde $\bar{P} = P$ unverändert weitergeleitet, so dient es als Paketquittung $IACK_P$ und es findet keine Sendewiederholung statt, da B eindeutig das gleiche Paket vorliegen hat.

Wird dagegen ein weitergeleitetes Paket mit $P \neq \bar{P}$ erkannt, bei dem die Nachrichten als „neu kombiniert“ gekennzeichnet wurden und gilt außerdem $\exists M_i \in P$ und $\exists M_j \in \bar{P}$ mit $M_i = M_j$, so wird die maximal durchgeführte Anzahl an Sendewiederholungen pro Nachrichtenquittung $IACK_M$ anteilig an $|P|$ reduziert.

Diese Vorgehensweise vermeidet zwar das Senden von unnötigen Sendewiederholungen nicht gänzlich, erhöht aber die Effizienz des Kommunikationsschema.

Die Quantität der fehlenden und unnötigen Sendewiederholungen wird jedoch deutlich reduziert, da die Ähnlichkeit des gesendeten und weitergeleiteten Paketes anhand der enthaltenen Nachrichten geprüft wird.

5.2.3 Latenz

Die Verwendung von opportunistischen Verfahren zur Weiterleitung bei Netzwerkcodierung verursacht auf den Zwischenknoten an mehreren Stellen Verzögerungen, die sich unmittelbar auf die Latenz für die Übertragung der Nachricht M zwischen Quell- und Zielknoten auswirken.

- t_{pfad}: Zusätzliche Verzögerung durch Weiterleitung von M auf nicht-optimalen Pfaden zwischen Quell- und Zielknoten.
- t_{dec}: Zeitdauer für das Durchführen des Decodierens von Paketen in denen M enthalten ist. Die Zeitdauer umfasst die Resolution von Nachrichten, sowie den Zeitaufwand für das Umwandeln der Pakete in Nachrichten und das Anwenden mathematischer Verfahren zum Decodieren gemäß des verwendeten Codierungsschemas.
- t_{enc}: Zeitdauer für die Codierung einer Nachrichtenkombination in der M enthalten ist. Die Dauer umfasst neben der Berechnung eines codierten Paketes entsprechend des Codierungsschemas auch die Auswahl geeigneter Nachrichten zur Kombination und die Berechnung der Codierungsvektoren.
- t_{inov}: Zeitdauer bis auf dem Zwischenknoten ausreichend viele innovative Nachrichten zum Decodieren von M eingetroffen sind.

Als Gesamtverzögerung ergibt sich auf einem Pfad mit den Zwischenknoten Z_1 bis Z_k:

$$t(Q \to S) = t_{pfad} + \sum_{i=1}^{k} (t_{dec}(Z_i) + t_{enc}(Z_i) + t_{inov}(Z_i)) \tag{5.1}$$

Die durch t_{pfad} verursachte Verzögerung tritt nicht bei jeder Topologie auf und wäre hier durch die Einführung eines Routing-Protokolls ggf. vermeidbar. Da-

bei ist jedoch der zeitliche Aufwand zur Etablierung einer Routingtopologie zu berücksichtigen, die der Weiterleitung von Einzelnachrichten voraus geht.

Die Verzögerung durch t_{dec} und t_{enc} kann durch Auswahl eines kleinen Wertes m zur Beschränkung der Nachrichtenmenge und Auswahl eines einfachen Codierungsschemas, etwa mit einem kleinen Galois-Feld, reduziert werden. Für ressourcenbeschränkte, drahtlose Sensornetze bietet sich daher insbesondere XOR als Codierungsschema an. Auf leistungsfähigen Systemen sind diese Verzögerungen in Relation zu den anderen jedoch zu vernachlässigen.

Den deutlichsten Einfluss auf die Gesamtverzögerung der Zwischenknoten hat jedoch t_{inov}. Diese Zeitdauer wird sowohl durch das Weiterleitungsverfahren, als auch das Codierungsschema beeinflusst. Die Beschränkung von m kann der Verzögerung entgegen wirken, indem die Codierungskomplexität gesenkt wird und weniger andere Pakete zum Decodieren vorausgesetzt werden. Dennoch ist für zeitkritische Informationen eine Vermeidung dieser Latenzen wünschenswert. OCSSIM sieht daher auch die Priorisierung von Einzelnachrichten vor, indem auf Netzwerkcodierung der Nachrichten und Kombination mit anderen Nachrichten verzichtet wird.

Effektiv kommt dies einer Situation mit $m = 1$ gleich. Es wird in OCSSIM durch Nutzung des *Fast-Forwarding-Mechanismus* (siehe Kapitel 5.4.2), aber auch die Verzögerung von t_{dec} und t_{enc} reduziert.

Werden vom Weiterleitungsverfahren Mechanismen verwendet, die ein zeitweises Rückstellen oder dauerhaftes Verwerfen von Nachrichten vorsehen, wird auch t_{inov} beeinflusst. Priorisiertes Weiterleiten in OCSSIM nutzt daher eine reduzierte Rückstelldauer bei der Nachrichtenverteilung und das Verwerfen von Nachrichten geschieht nur bei bekannten Nachrichten, die bereits weitergeleitet wurden oder häufig genug in der Nachbarschaft beobachtet werden konnten, um die Abdeckung und Weiterleitung im Netzwerk zu gewährleisten (siehe Kapitel 5.5.1).

Aufgrund des großen Einflusses von t_{inov} auf die Latenz, eignet sich ein opportunistisches Weiterleitungsverfahren nicht für jede Art von Anwendung.

Werden durch eine Quelle häufig Nachrichten gesendet, so ist eine geringe Zeitdauer t_{inov} die Folge, da schnell die nötige Nachrichtenmenge zum Decodieren auf dem Zwischenknoten bereitsteht. Hier kommt es nicht darauf an, dass Nachrichten streng periodisch oder mit einer bestimmten Regelmäßigkeit gesendet werden, sondern dass im entsprechenden Zeitraum eine ausreichend hohes Datenaufkommen gesendet wird, so dass das verwendete Codierungsschema ein Decodieren im Zeitrahmen der maximal tolerierbaren Latenz ermöglichen kann.

Senden Quellen stattdessen nur sporadisch einzelne Nachrichten, so kann eine sehr hohe Latenz entstehen. In diesem Fall sollten die einzelnen Nachrichten zur Weiterleitung priorisiert werden, um die Latenz zu reduzieren. Eine weitere Möglichkeit ist das Kombinieren von Nachrichten aus sporadischen Quellen, mit Nachrichten aus Quellen, die häufig Nachrichten senden. Die Nachrichten aus beiden Quellen können von einem weiterleitenden Knoten in ein gemeinsames Paket codiert wer-

den. Bei der Weiterleitung kann die Nachricht aus der sporadischen Quelle somit von der niedrigeren t_{inov} der häufigen Nachrichtenquelle profitieren.

5.3 Zustandshaltung und Zwischenspeicherung zu Codierungszwecken

Die Nutzung der Netzwerkcodierung wird erst durch die Bereitstellung ausreichender Zwischenspeicher für Nachrichten auf den Zwischenknoten möglich. Sowohl zum Sammeln der erforderlichen Pakete für das Decodieren, als auch zum Sammeln von Nachrichten für gemeinsame Codierung, ist je nach Codierungskomplexität des Codierungsschemas eine Mindestspeichermenge in getrennten Speichern erforderlich.

Die Realisierung und das Zusammenspiel der Puffer wird ausführlich in Kapitel 6.4 behandelt.

OCSSIM sieht zur Zustandshaltung drei unterschiedliche Typen von Paket- und Nachrichtenpuffer zur lokalen Zwischenspeicherung vor.

5.3.1 Encoding Buffer

Der *Encoding Buffer* $\mathcal{B}_{enc}$ dient als Pufferspeicher für decodierte und neu erstellte Nachrichten. Er wird nur benutzt, um Nachrichten bis zum nächsten Codierungszeitpunkt zu speichern. Ein Codierungsdurchlauf wird entweder durch Erreichen des maximalen Füllstandes des Encoding Buffers angestoßen oder durch einen konfigurierbaren periodischen Zeitgebers t_{enc}. Der Zeitgeber dient dabei auch zur Anpassung der Latenz während der Weiterleitung von Nachrichten durch die Reassembling-Stufe (siehe Abschnitt 5.4.3).

Die Größe des Encoding Buffers ist abhängig von der maximalen Anzahl codierter Nachrichten m, die in einem Paket gespeichert werden können. Außerdem ist die Größe entscheidend für die Anzahl der Kombinationsmöglichkeiten, die ein Knoten aus den empfangenen Nachrichten beim Neucodieren erzeugen können soll. Zweckmäßig ist eine Mindestgrößen, von $|\mathcal{B}_{enc}| \geq 2m$ um die Nachrichten von zwei Paketen mit maximaler Nachrichtenanzahl zu kombinieren. Eine Größe von $|\mathcal{B}_{enc}| = m^2$ erlaubt eine höhere Vielfalt an Nachrichtenkombinationen, aber erhöht auch gegebenenfalls die Latenz, da zunächst der Puffer ausreichend gefüllt werden muss, bis eine neue Nachrichtenkombination erzeugt wird.

5.3.2 Decoding Buffer

Der Decoding Buffer $\mathcal{B}_{dec}$ wird benötigt, um die Nutzdaten und den Codierungsvektor $C_1, ..., C_m$ eines Paketes zu speichern. Die Daten eingehender Pakete, die nicht unmittelbar mit den vorhandenen Informationen decodiert werden können, werden in diesen Puffer überführt. Dabei werden die Daten so lange gepuffert, bis eine Decodierung mittels Kombination vorhandener und später eingegangener Nachrichten möglich ist oder bis der Decoding Buffer seinen maximalen Füllstand erreicht hat. Wird der maximale Füllstand durch eingehende Pakete überschritten, so werden die ältesten vorhandenen Daten aus dem Puffer entfernt und verworfen,

bevor die neuen Daten eingefügt werden. Die Größe des Decoding Buffers hat starken Einfluss auf die Anzahl der Decoding-Möglichkeiten, ist jedoch zugleich eine entscheidende Einflussgröße für den Speicherbedarf des Kommunikationsschemas. Ein Anstieg im Nachrichtenverkehr, eine Erhöhung der Anzahl der Netzwerkteilnehmer, aktiven Anwendungen oder Nachbarknoten bedingen im Allgemeinen auch einen größeren Decoding-Puffer, wenn die Anzahl der Decoding-Möglichkeiten auf einem konstanten Niveau gehalten werden soll.

5.3.3 Atomic Buffer

Der Atomic Buffer $\mathcal{B}_{atom}$ dient wie auch der Encoding Buffer zur Zwischenspeicherung von decodierten Nachrichten. Während atomare Pakete direkt im Speicher abgelegt werden können, werden andere Pakete beim Decodieren in mehrere, kleinere Pakete aufgelöst. Werden die Pakete dabei in atomare Pakete zerlegt, so werden sie anschließend im Atomic Buffer gespeichert. Im Gegensatz zum Decoding Buffer bestehen die gespeicherten Daten neben den Nutzdaten nicht aus dem gesamten Codierungsvektor $C_1, ..., C_m$ eines Paketes, sondern nur aus dem relevanten Teil C_i für die Nachricht M_i des atomaren Paketes. Die gespeicherten Daten des Atomic Buffers erleichtern insbesondere den Decodierungsvorgang später eintreffender Pakete. Dazu werden die Einträge des Atomic Buffers noch vor den Einträgen des Decoding Buffers zum Decoding genutzt.

Können die Puffereinträge mit einem eintreffenden Paketes P kombiniert werden, so gilt aufgrund der Eigenschaft atomarer Pakete und der Codierungskomplexität $c_{\mathcal{B}_{atom}} = 1$, dass die Codierungskomplexität nach Anwendung der Resolution sinkt $c_{\mathcal{R}(P)} < c_P$. Kann ein Paket mit $k = |P|$ codierten Nachrichten fortgesetzt mit Einträgen aus $\mathcal{B}_{atom}$ aufgelöst werden, so verbleibt nach $k - 1$ Resolutionsschritten wiederum ein atomares Paket, das direkt in eine decodierte Nachricht umgewandelt werden kann. Ist die fortgesetzte Resolution mit Einträgen des Atomic Buffers nicht mehr möglich, so wird entsprechend dem verwendeten Codierungsschemas der Decodierungsprozess mit Einträgen aus $\mathcal{B}_{dec}$ fortgesetzt.

Um bei wachsenden Netzwerkgrößen das Verwerfen von alten Einträgen des Atomic Buffers zu vermeiden, muss die Größe des Pufferspeichers an die wachsende Anzahl von Übertragungen und versendeten Pakete angepasst werden.

5.3.4 Basisannahmen zur Zwischenspeicherung

Zum Entwurf der Nachrichten- und Paket-Zwischenspeicher für Encoding und Decoding wurden die folgenden Basisannahmen getroffen:

- Die Anzahl der Puffereinträge für atomare Pakete ist im direkten Vergleich zu den benötigten Puffereinträgen für die regulären Pakete hoch, da viele Pakete bereits direkt beim Empfang decodiert werden können oder ein Paket aus einer bereits bekannten und nicht-innovativen Nachrichtenkombination besteht, die verworfen werden kann. Der Speicherbedarf von OCSSIM kann daher signifikant reduziert werden, wenn für atomare Pakete ein eigener Pufferspeicher vorgesehen wird. Atomare Pakete benötigen pro Eintrag im

Pufferspeicher, neben der Nachricht im Klartext, lediglich einen einzelnen Codierungsvektor als Kennung der enthaltenen Einzelnachricht. Würden die Nachrichten stattdessen gemeinsam mit regulären Paketen gespeichert, so würde pro Eintrag ein Speichermehraufwand von $m - 1$ Codierungsvektoren anfallen. Zusätzlich muss die vom Codierungsschema vorgegebene Länge jedes Vektoreintrages berücksichtigt werden. Bei Random Linear Coding (siehe 3.3.4) muss beispielsweise der Speichermehraufwand zusätzlich mit der Größe des verwendeten Galois-Feldes multipliziert werden.

- Die Größe des Feldes für den Codierungsvektor C sollte möglichst klein gewählt werden. Unterschiedliche Pakete, die innerhalb des gleichen Zeitintervalls eintreffen, müssen jedoch anhand C unterscheidbar bleiben. Die Anzahl Nachrichten, die in ein gemeinsames Paket codiert werden $k < m$ sollte kleiner als die durchschnittliche Anzahl der Nachbarn pro Knoten sein. Kleinere Werte für k vereinfachen den Decoding-Prozess und erhöhen die Redundanz auf Kosten des erreichbaren Durchsatzes.

- Die Beschränkungen drahtloser Sensorknoten hinsichtlich der verfügbaren Speichermenge und Nachrichtengröße erfordern eine strikte Beschränkung der Anzahl der gemeinsam in ein Paket codierten Nachrichten. Beispielsweise ist die standardmäßig vorgegebene Nachrichtengröße des Betriebssystems TinyOS auf 29 Bytes begrenzt [65].

  Sämtliche Puffer für die Decodierung und Codierung, sowie Warteschlangen die von OCSSIM benötigt werden, müssen sich dabei den verfügbaren Arbeitsspeicher eines Sensorknotens mit den Anwendungen teilen.

5.4 Pipeline-Verarbeitungsmodell eingehender Pakete

Die lokale Verarbeitung eingehender Pakete erfolgt in OCSSIM gemäß eines mehrstufigen Pipeline-Modells. Das grundlegende Modell unterscheidet sich dabei zwischen den Sensorknoten nicht, da jeder Knoten unterschiedliche Rollen für die einzelnen Übertragungen einnimmt und Pakete in der Regel unterschiedliche Nachrichten aus mehreren verschiedenen Ende-zu-Ende-Übertragungen bündeln.

Abbildung 5.5 zeigt das grundlegende schematische Pipeline-Verarbeitungsmodell eingehender Pakete auf einem einzelnen Sensorknoten. Alle eingehenden Pakete eines Knotens stammen von direkten Nachbarn, die bereits eine positive Weiterleitungsentscheidung gefällt haben. Eingehende Pakete werden ausschließlich per Overhearing empfangen und müssen vor der weiteren Verarbeitung zunächst anhand ihrer Eigenschaften untersucht werden.

Dazu sind die folgenden Eigenschaften eines codierten Paketes zu betrachten:

Decoding-Möglichkeiten: Zunächst muss ein empfangender Knoten das eingehende Paket daraufhin prüfen, ob Nachrichten des Paketes mit Hilfe von lokal vorhandenen Nachrichten decodiert werden können. Dazu wird der

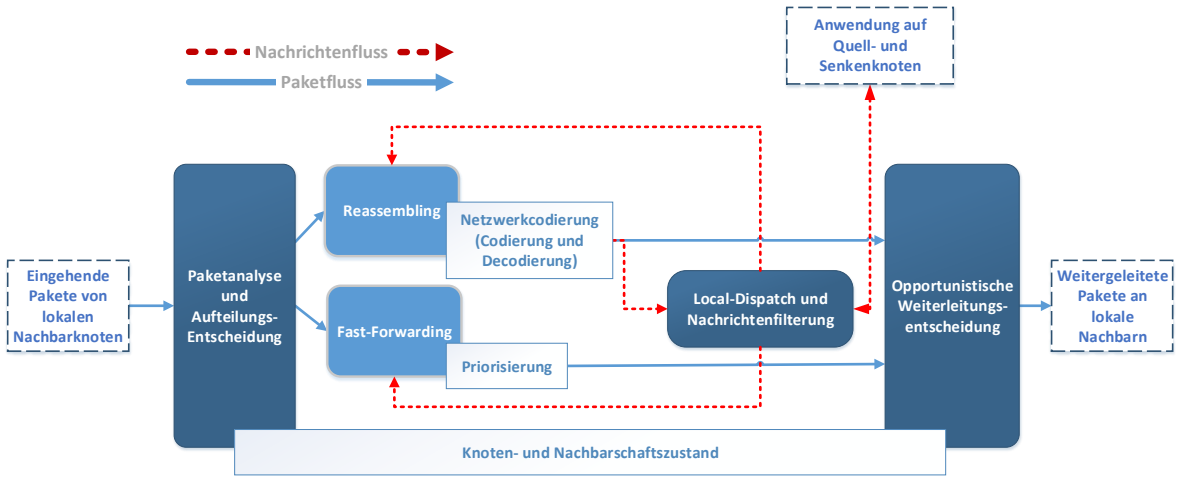

Abbildung 5.5 Grundlegendes Verarbeitungsmodell für eingehende Pakete auf OCSSIM-Knoten

Codierungsvektor der im Paket gespeicherten Nachrichten ausgelesen. Dieser wurde beim Codieren des Paketes erstellt und nutzt je nach verwendetem Codierungsschema Skalare, Hashsummen oder Hilfswerte, um die enthaltenen Nachrichten identifizieren und decodieren zu können. OCSSIM sieht hier für jede enthaltene Nachricht einen eigenen Vektor vor und beschränkt zudem die maximale Anzahl von Nachrichten pro Paket. Die Anzahl der tatsächlich enthaltenen Nachrichten kann direkt anhand der verwendeten Vektoren ermittelt werden. Ist das Paket ein atomares Paket mit nur einer einzeln Nachricht, so kann das Paket ohne Zuhilfenahme weiterer gepufferter Pakete mit dem verwendeten Codierungsschema decodiert werden.

Sender des Paketes: Aufgrund der in einem Paket codierten Übertragungen, können zwar die Quellknoten der Nachrichten ohne vorhergehendes Decoding nicht ermittelt werden. Jedes Paket enthält aber eine Absenderadresse des Nachbarknotens, der das betrachtete Paket gesendet hat. Diese Information spielt im Weiterleitungsprozess eine vielseitige Rolle. Sie wird sowohl verwendet, um die Größe der lokalen Nachbarschaft für spätere Weiterleitungsentscheidungen zu ermitteln, als auch für eine Überprüfung welche Nachbarknoten die eigenen Nachrichten weiterleiten. Darüber hinaus kann die Information auch dazu verwendet werden, Nachrichten aus vorgegebenen Richtungen und Netzbereichen zu filtern und ein zyklisches Weiterleiten zwischen zwei Nachbarn zu unterbinden.

Priorisierung: Nachrichten, die als priorisiert gekennzeichnet sind, müssen zur bevorzugten Verarbeitung im Strom der eingehenden Pakete erkannt, ausgefiltert und ggf. zugestellt werden. Pakete, die priorisierte Nachrichten enthalten, werden zu diesem Zweck mit einer geringen Anzahl von Nachrichten gemeinsam oder in atomare Pakete codiert.

Paketkennung: Die Kombination der *MsgIDs* aus den enthaltenen Nachrichten kann bei geeigneten kollisionsarmen Verfahren auch dazu genutzt werden, um ein Paket hinreichend genau zu kennzeichnen. Somit wird ein Cache-Vergleich mit vorausgehenden Paketen möglich. Bei einem Cache-Treffer kann das eingehende Paket ohne weitere Verarbeitung verworfen werden, da identische Informationen mit hoher Wahrscheinlichkeit bereits empfangen wurden. Die Paketkennung durch die *MsgIDs* ersetzt hier einen üblichen Vergleich mit der Sequenznummer eines empfangenen Paketes, um Schleifen oder redundante Weiterleitungen zu vermeiden.

Nach der Prüfung der Paketeigenschaften wird in der *Aufteilungsstufe* des Pipeline-Modells der weitere Verarbeitungsschritt des Paketes festgelegt. Hier wird bereits eine Vorentscheidung für die Weiterleitung der Nachricht gefällt.

Die Aufteilungsstufe entscheidet dabei, ob ein eintreffendes Paket mittels der *Fast-Forwarding-Stufe* priorisiert weitergeleitet werden soll. Außerdem ist es möglich die Nachrichten des Paketes zunächst zu decodieren, bevor die lokal vorhandenen Nachrichten in der *Reassembling-Stufe* in einer neuen Kombination in ein Paket zu codieren, um anschließend weitergeleitet zu werden.

Jedes Paket kann dabei zugleich in der Fast-Forwarding-Stufe, als auch der Reassembling-Stufe weiterverarbeitet werden. So ist zum Beispiel eine lokale Zustellung von decodierten Nachrichten aus der Reassembling-Stufe möglich, die zugleich auch an weitere Knoten der Nachbarschaft mittels der Fast-Forwarding-Stufe weitergeleitet werden.

Sowohl die lokale Zustellung, als auch das Reassembling, setzen einen Decoding-Schritt voraus, bei dem das eingehende Paket zumindest teilweise decodiert werden kann. Die dabei decodierten Nachrichten können direkt in den nachfolgenden Stufen verwendet werden. Nicht-decodierte Teile der Pakete können zumindest in der Reassembling-Stufe weiter genutzt werden, um ggf. von benachbarten Knoten decodiert werden zu können (siehe Kapitel 5.4.3). Auch das Decoding später eintreffender Pakete kann mittels Pufferung teilweise nicht-decodierter Pakete verbessert werden.

Anschließend muss in der *Local-Dispatch*-Stufe entschieden werden, ob Nachrichten des Paketes an diesen Knoten adressiert sind und somit einer lokalen Anwendung zugestellt werden sollen. Dies ist der Fall sofern der betrachtete Knoten Quelle bzw. Senke einer Nachrichtenübertragung ist.

Die opportunistische Weiterleitungsentscheidung findet als letzter Schritt der Pipeline statt. Dabei wird sichergestellt dass Pakete in dieser Phase bereits wieder per Netzwerkcodierung codiert wurden oder codiert geblieben sind.

Mit dem OCSSIM Pipeline-Modell können verteilte und kooperative Anwendungen entwickelt werden, die passiv Zustands- und Monitoringinformationen des Netzwerkes sammeln, indem der verarbeitete Nachrichtenfluss aus eingehenden Paketen der Nachbarschaft beobachtet und gefiltert wird.

Weil die Hardware von drahtlosen Sensorknoten häufig keine echte Nebenläufigkeit ermöglicht, muss ggf. bei Verwendung das eingesetzten Pipeline-Modell auf eine parallele Ausführung mehrerer Stufen verzichtet werden. Dennoch erlaubt das Pipeline-Modell, dass die Nachrichten unterschiedlicher Quellen, Zeitpunkte und Anwendungen in den jeweils gleichen Stufen verarbeitet werden können. Der wichtigste Vorteil ist aber die Möglichkeit, zu verschiedenen Zeitpunkten empfangene Pakete, gemeinsam mit neu eintreffenden Paketen zu decodieren. Somit abstrahiert das Pipeline-Modell von einer Einzelbehandlung unterschiedlicher Nachrichtentypen und -formate. Es stellt den jeweils realisierten Anwendungen als Schnittstelle einen einheitlichen Nachrichtenstrom mit Möglichkeiten zur Priorisierung zur Verfügung. Für die Anwendungsentwicklung kann zudem auf weitere Maßnahmen für das Routing oder zur Adaption an dynamisch veränderliche Netzwerktopologien verzichtet werden.

Da beim Pipline-Modell keine Unterscheidung nach Quellen oder Anwendungen durchgeführt wird, ist es den Knoten im Netzwerk möglich kooperativ zu arbeiten. Dabei werden auf den Knoten auch Nachrichten decodiert und ausgewertet, die nicht explizit an den beobachtenden Knoten adressiert sind. Zudem werden durch die Bündelung von Nachrichten in codierten Paketen auch Informationen abseits eines direkten Übertragungspfades weitergeleitet, so dass die filterbaren Nachrichten auch netzwerkweit oder an Netzbereiche außerhalb der eigenen Nachbarschaft adressiert sein können. Dieses Verhalten kann für sicherheitskritische Anwendungen bedenklich sein, so dass weitere Maßnahmen zur Sicherstellung der Vertraulichkeit und Authentizität getroffen werden müssen.

Sollte eine Anwendung weitere Maßnahmen zur Sicherstellung der Vertraulichkeit und Authentizität erfordern, so können diese auf Anwendungsebene durchgeführt werden. Durch eine Anwendung bereitgestellte Daten werden durch OCSSIM nicht interpretiert. Zusätzliche Sicherheitsmaßnahmen können auf der Nachricht vor der Codierung bzw. nach der Decodierung durchgeführt werden. Deshalb bleiben Sicherheitsmaßnahmen, wie z. B. Ende-zu-Ende-Verschlüsselung, mit dem OCSSIM Pipeline-Modell möglich.

In kooperative Netzwerken und Anwendungen kann das Verhalten aber auch einen deutlichen Mehrwert darstellen, da Knoten die vorwiegend an der Kommunikation einer anderen Anwendung beteiligt wären, zusätzliche Pfade zwischen Quelle und Senke aufspannen können. Damit lässt sich die Zuverlässigkeit der Kommunikation durch redundantes Bereitstellen der gesendeten Daten erhöhen. Das gemeinsame Codieren von Nachrichten unterschiedlicher Anwendungen ermöglicht das Decodieren, auch wenn eine einzelne Anwendung nur sehr wenige Daten versendet. Zudem könne die bereitstehenden Knoten gemeinsam durch die Kommunikation unterschiedlicher Anwendung oft effektiver und gleichmäßiger ausgelastet werden.

5.4.1 Paketanalyse und Aufteilung eintreffender Pakete

Eingehende Pakete werden in der Aufteilungsstufe der OCSSIM-Pipeline für eine Weiterleitung den nachfolgenden Stufen *Fast-Forwarding* und *Reassembling* zuge-

teilt. Zweck dieser Aufteilung ist die Koordination der Weiterleitung anhand von unterschiedlichen Kriterien.

Während die Fast-Forwarding-Stufe primär der Priorisierung und initialen Verteilung von Paketen mit niedriger Codierungskomplexität dient, ist der Zweck der Reassembling-Stufe das effiziente Nutzen der Netzwerktopologie, indem über verschiedene Pfade unterschiedliche Pakete verteilt werden und so neue Codierungs- und Decoding-Möglichkeiten geschaffen werden.

Paketaufteilung auf dem lokalen Knoten

Die Aufteilung der Pakete zur Fast-Forwarding-Stufe und Reassembling-Stufe geschieht in OCSSIM im wesentlichen anhand der bereits in Kapitel 5.2.1 vorgestellten Kriterien zu *Ursprungsknoten*, *Codierungskomplexität* und *Kenntnis des Paketes*.

Anhand des Merkmals *Codierungskomplexität* lässt sich direkt ermitteln, ob ein atomares Paket vorliegt ($c = 1$). Wird das atomare Paket zur Initialisierung der Übertragung innerhalb der vorgegebenen Anzahl Hops vom Ursprungsknoten aus weitergeleitet, so wird das Paket der Fast-Forwarding-Stufe zugeteilt. Wird die vorgegebene Anzahl Hops h dagegen überschritten, so muss weiter überprüft werden, ob die Nachricht des atomaren Paketes dauerhaft priorisiert werden soll. Dann erfolgt ebenfalls eine Zuteilung an die Fast-Forwarding-Stufe. Ohne Priorisierung erfolgt hier eine Zuteilung an die Reassembling-Stufe, so dass nun eine Kombination aus Nachrichten an die Nachbarknoten weiterverteilt wird.

Wird die Nachricht eines atomaren Paketes dauerhaft priorisiert, so wird das Paket stets der Fast-Forwarding-Stufe zugeteilt.

Pakete mit $c > 1$ werden heuristisch behandelt. Pakete mit höherer Codierungskomplexität erhalten dabei tendenziell eine niedrigere Priorität bei der Weiterleitung. Dies wird gewährleistet, indem die einzelnen $MsgIDs$ der enthaltenen Nachrichten mit den bekannten $MsgIDs$ der lokal vorliegenden, decodierten Nachrichten aus $\mathcal{B}_{enc}$ und $\mathcal{B}_{atom}$ verglichen werden.

Sind sämtliche $MsgIDs$ im Paket unbekannt, erfolgt stets eine Weiterleitung mittels der Fast-Forwarding-Stufe. Ist die Anzahl der bekannten $MsgID$ größer als $thr_{Reassembling} = \frac{c}{2}$, so wird das Paket der Reassembling-Stufe zugeführt, da dies für vorhandene Decoding-Möglichkeiten spricht. Ansonsten wird das Paket über die Reassembling- und die Fast-Forwarding-Stufe weitergeleitet, um es sowohl schnell in der Nachbarschaft zur Verfügung zu stellen, als auch die lokale Decodierung und Nachrichtenverteilung zu verbessern.

Im Fall, dass alle $MsgIDs$ lokal bekannt sind, ist ein frühes Verwerfen der Nachricht bereits in der Aufteilungsstufe effizient. Ein anderes Verhalten würde die unnötige Verteilung von nicht-innovativen Nachrichten im Netzwerk nach sich ziehen.

Auswirkungen auf die Paketaufteilung nachfolgender Knoten

Pakete können vom Ursprungsknoten neuer Nachrichten mit einem Parameter h versehen werden, der angibt, wie viele Hops die Nachricht in einem atomaren Paket

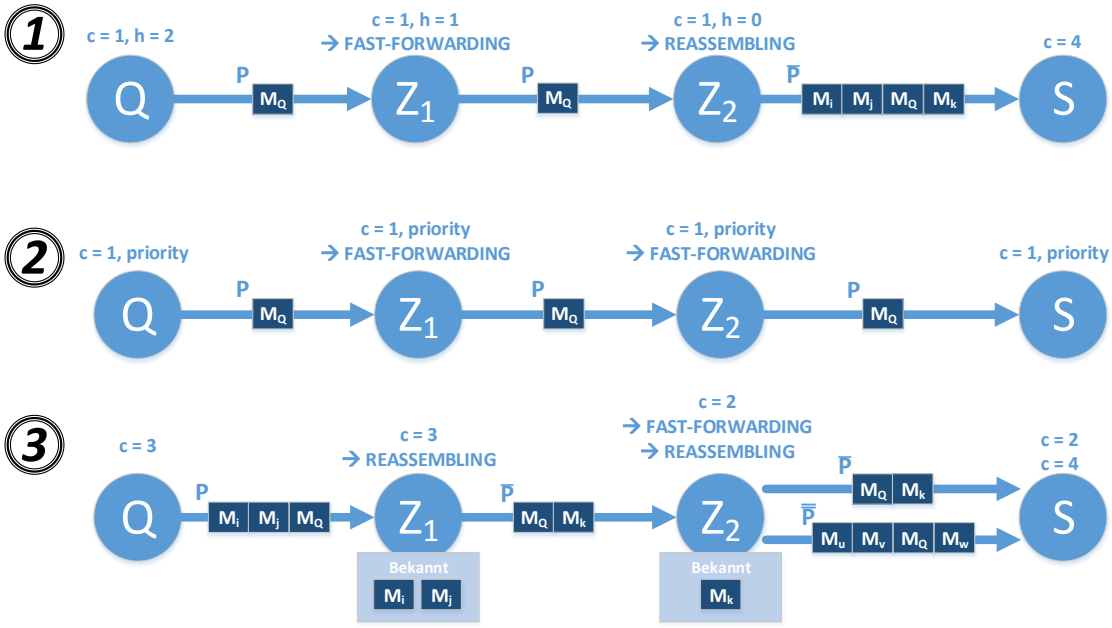

Abbildung 5.6 Aufteilung eingehender Pakete zur Weiterleitung

mittels Fast-Forwarding weitergeleitet werden muss. Jeder Knoten dekrementiert h bei der Weiterleitung mittels Fast-Forwarding. Erst bei erreichen von $h = 0$ darf die Nachricht zur Weiterleitung mit anderen Nachrichten kombiniert werden. Damit wird erreicht, dass auf den ersten Hops eine möglichst hohe Verbreitung der neuer Nachrichten vorhanden ist, die das Decodieren erleichtert und die Anzahl möglicher Nachrichtenkombinationen bei der Codierung erhöht.

Beispielhaft wird das Verfahren zur Aufteilung eingehender Pakete in Abbildung 5.6 dargestellt. In Beispiel ① erfolgt die Weiterleitung des atomaren Paketes P bei der Initialisierung der Übertragung und einer Priorisierung über $h = 2$ Hops. Der Zwischenknoten Z_1 leitet das Paket unverändert mittels Fast-Forwarding weiter und reduziert zuvor lediglich h. Z_2 reduziert h abermals, so dass $h = 0$ nun die weitere Verteilung mittels der Reassembling-Stufe vorsieht und Z_2 die in P enthaltene Nachricht M_Q mit den lokal vorliegenden Nachrichten M_i, M_j und M_k zu $\bar{P}$ kombiniert.

In Beispiel ② findet die Weiterleitung von P durchgehend priorisiert statt, so dass die Zwischenknoten stets zwischen Q und S die Fast Forwarding-Stufe nutzen.

Das Beispiel ③ zeigt die Aufteilung von nicht-atomaren Paketen. Dazu besteht P im Beispiel aus der Nachrichtenkombination von M_i, M_j und der lokal erstellten Nachricht M_Q. Die Nachrichten M_i und M_j sind dabei schon durch vorangegangene Verteilungsvorgänge in Teilen des Netzwerkes bekannt. Soll Z_1 nun P eine Stufe zur Weiterleitung zuteilen, überprüft er anhand der $MsgIDs$ die enthaltenen Nachrichten mit den $MsgIDs$ der bereits bekannten Nachrichten M_i und M_j. Da zwei bekannte Nachrichten vorliegen, wird der Schwellenwert von $thr_{Reassembling} = \frac{3}{2}$ überschritten und die Reassembling-Stufe ausgewählt. Auf Z_2 findet der gleiche

Vorgang für das neu kombinierte Paket $\bar{P}$ statt. Es liegt jedoch keine Mehrheit an bekannten Nachrichten in $\bar{P}$ vor. Somit wird $\bar{P}$ unverändert mittels der Fast-Forwarding-Stufe an S weitergeleitet. Zudem wird M_Q später als Teil der neu kombinierten Nachricht $\bar{\bar{P}}$ mit weiteren Nachrichten M_u, M_v und M_w übertragen.

Prinzipiell ist für die Bewertung der Pakete eine Vielzahl weiterer Metriken anwendbar. Beispielsweise lässt sich die Anzahl der Nachbarn und ihr jeweiliger Knotengrad verwenden, um Wahrscheinlichkeiten zur Weiterleitung zu berechnen, die an die zugrunde liegende Netzwerktopologie angepasst sind (vgl. [58]). Somit lässt sich auch das Verhalten von Knoten an kritischen Positionen, wie Engstellen oder bei abseits liegenden Knoten im Zusammenspiel mit der Nachbarschaft, verbessern. Weitere Bewertungsverfahren werden in Kapitel 6.3.5 diskutiert.

Die Zustellung der im Paket codierten Nachrichten an einen Zielknoten oder einzelne Anwendungen kann für reguläre Pakete in der Aufteilungsstufe noch nicht erfolgen. Anders ist dies beim Empfang atomarer Pakete, wo eine beschleunigte Nachrichtenzustellung an eine Anwendung bereits in dieser Stufe möglich ist. Die beschleunigte Zustellung eignet sich insbesondere, wenn eine lokale Anwendung zugunsten der Latenz auf die gemeinsame Codierung von Nachrichten unterschiedlicher Quellen und Übertragungen verzichtet. In diesem Fall können zum Transport der Nachrichten jeweils Einzelpakete verwendet werden, die neben der Nachricht eine gültige Paketidentifikation und $MsgID$ besitzen muss. Die Entscheidung, ob und mittels welcher Stufe ein atomares Paket weitergeleitet wird, bleibt davon unberührt.

5.4.2 Priorisierung von Nachrichten

Bevor ein Knoten eigene Nachrichten versendet, wartet er eine vorgegebene Zeit, um die eigene Nachricht mit Nachrichten anderer Knoten in einem Paket kombinieren zu können und somit die Anzahl der insgesamt benötigten Pakete bzw. Sendevorgänge zu reduzieren.

Wenn trotz des Wartezeitraumes keine anderen Übertragungen im Netzwerk zur Nachrichtenkombination genutzt werden können oder keine weiteren Übertragungen in der eigenen Nachbarschaft präsent sind, können zwei bedeutende Situationen auftreten. Beide Situationen treten üblicherweise initial zu Beginn der Verteilung neuer Nachrichten auf und betreffen Anfragenachrichten von Quellknoten in stärkerem Maße als Rückantwortnachrichten.

In der ersten Situation erhöht sich die Latenz zwischen Quelle und Senke deutlich, ohne dass sich die Übertragungseffizienz durch Kombination der Nachrichten verbessern lässt. Dies liegt darin begründet, dass ein Knoten mit Sendewunsch seiner Nachricht jeweils vor der Weiterleitung maximal die Zeitspanne t_{enc} abwarten muss, wenn so lange keine weiteren Nachrichten zur Kombination auf dem Knoten vorliegen.

Die Verteilung von Nachrichten, die mit erhöhter Priorität zum Zielknoten transportiert werden sollen, wird durch diesen Mechanismus unnötig verzögert. Gerade für die Signalisierung von Fehlern ist daher eine getrennte Behandlung priorisierter Nachrichten notwendig.

Die zweite Situation zeichnet sich durch eine langsame Verbreitung der Nachrichten in einem codierten Paket aus, das einen hohen Informationsgehalt aufweist und somit viele für Nachbarn unbekannte Nachrichten enthält. In dieser Situation wird die Verbreitung der Nachrichten durch zusätzliche Verzögerung vor dem möglichen Decodieren der Nachricht aufgehalten. Bei Paketen mit innovativer Nachrichtenkombination und kleiner Nachrichtenmenge n ist die zusätzliche Verzögerung daher nicht vertretbar.

Hier bieten sich mehrere Maßnahmen zur Verbesserung der initialen Verteilung neuer Nachrichten an.

- Versenden von Paketen mit geringer Nachrichtenanzahl: Die Reduktion der Nachrichtenanzahl vereinfacht das Decodieren, da weniger andere Nachrichten zur Resolution bekannt sein müssen. Da diese Maßnahme zugleich im direkten Widerspruch mit der Zielsetzung steht, möglichst viele Nachrichten codiert zusammenzufassen, ist eine Beschränkung auf wenige Hops oder die lokale Nachbarschaft sinnvoll. Ist eine Nachricht nach der initialen Phase einer größeren Zahl von Zwischenknoten bekannt, so sollte die Maßnahme abgesetzt werden.

- Kombination von neuen mit alten Nachrichten: Neue Nachrichten können mit einem hohen Anteil von alten Nachrichten kombiniert werden, die bereits von diesem Knoten übertragen wurden, um die Decoding-Wahrscheinlichkeit auf benachbarten Knoten zu erhöhen. Zur Umsetzung dieser Maßnahme ist jedoch der Nachrichtenpuffer je Knoten zu vergrößern, da alte Nachrichten länger gespeichert und somit erst später verworfen werden können.

- Weiterleiten von nicht-decodierten Paketen: Auch wenn ein Zwischenknoten nach dem Empfang eines neuen Paketes die darin enthaltenen Nachrichten nicht decodieren kann, so ist dennoch ein Weiterleiten des unveränderten Paketes zweckmäßig, um die Decoding-Möglichkeiten auf benachbarten Knoten zu unterstützen. Benachbarte Knoten können die fehlenden Informationen für erfolgreiches Decodieren beispielsweise bereits über einen anderen Pfad erhalten haben.

Zur Vermeidung der langsamen initialen Verteilung von neuen Nachrichten wird im OCSSIM-Kommunikationsschema die *Fast-Forwarding*-Stufe verwendet. Ihre Aufgabe ist vorrangig ein schnelles Weiterleiten von priorisierten Nachrichten und Nachrichten niedriger Codierungskomplexität. Außerdem wird die Fast-Forwarding-Stufe genutzt, um eine Grundsättigung von innovativen Nachrichten in der direkten Nachbarschaft zu erreichen.

Dass Verwenden von Fast-Forwarding in gesättigten Netzwerkbereichen ist jedoch kontraproduktiv. Durch redundantes Versenden von nicht-innovativen Nachrichten erhöhen sich dann die Übertragungskosten. Außerdem führt dies zu erhöhter Kollisions- und Staugefahr. Die Einführung des Parameter h (siehe Abschnitt 5.4.1) garantiert hier, dass eine ausreichende Anzahl von Knoten unterschiedliche Nach-

richtenkombinationen erzeugen kann, so dass ein Decodieren in großen Teilen des Netzwerkes ermöglicht wird.

5.4.3 Neukombination von Nachrichten

Der wesentliche Zweck der *Reassembling-Stufe* ist es, die Anzahl potentieller Decoding-Möglichkeiten auf Nachbarknoten zu erhöhen. Zugleich werden die Übertragungen unterschiedlicher Nachbarknoten gebündelt, um die Nachrichtenmenge während der Verteilung im Netzwerk effizient zu begrenzen. Dazu werden neue Nachrichten aus zuvor empfangenen und erfolgreich decodierten Nachrichten erstellt und gemeinsam mit Paketfragmenten codiert, die bei der Resolution vorangegangener Pakete nicht vollständig aufgelöst werden konnten. Etwaige Nachbarknoten können diese Paketfragmente aber ggf. decodieren, wenn sie über zusätzliche innovative Nachrichten verfügen, die lokal noch nicht empfangen wurden. In der Reassembling-Stufe erstellte Pakete werden anschließend an die eigene Nachbarschaft weitergeleitet.

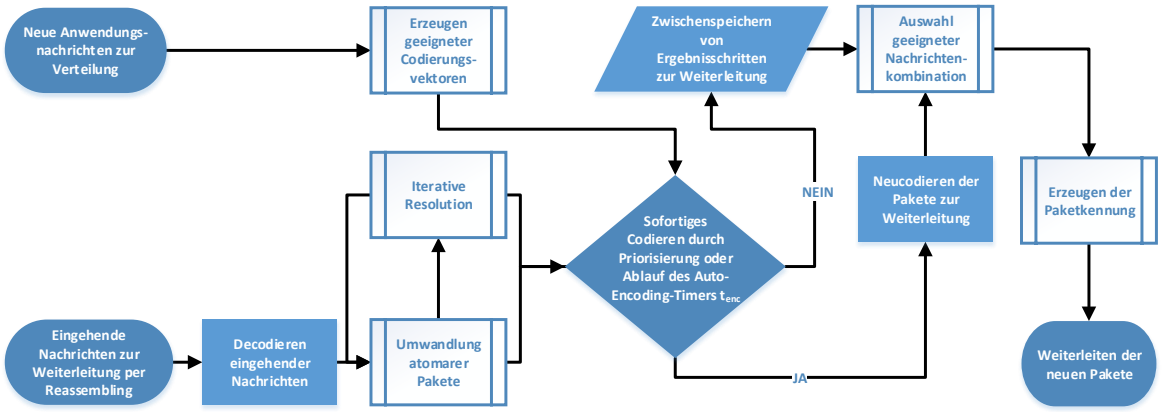

Abbildung 5.7 Ablaufdiagramm der Reassembling-Stufe

Abbildung 5.7 zeigt den Ablauf und die Einzelschritte der Reassembling-Stufe. Im Folgenden werden die einzelnen Teilschritte mit ihren Aufgaben und Funktionsweisen besprochen.

Die Reassembling-Stufe erhält an zwei Punkten neue Eingabeinformationen. Die erste Möglichkeit neue Informationen in die Reassembling-Stufe einzubringen, führt über neue Anwendungsnachrichten aus der vorgelagerten Aufteilungsstufe.

Erzeugen geeigneter Codierungsvektoren

Um die neue Nachricht gemeinsam mit anderen Nachrichten in ein Paket codieren zu können, wird zunächst ein neuer Codierungsvektor für die Nachricht benötigt. Die konkrete Vorgehensweise ist abhängig vom verwendeten Codierungsschema (vgl. Kapitel 3.3). Während bei Linear Random Coding die Codierungsvektoren im Wesentlichen zufällig aus einem Galois-Feld mit 2^n Elementen und einer eindeutigen Identifikation der Ausgangsnachricht ermittelt werden, ist für eine

Codierung per XOR lediglich die eindeutige Identifikation der Ausgangsnachricht für die spätere Decodierung notwendig.

Diese Form eines Codierungsvektors wird als *Nachrichtenidentifikator* der Nachricht M_i oder einfach $MsgID_i$ bezeichnet. Diese $MsgID$ kann beispielsweise durch eine einfache multiplikative Hashfunktion $\mathcal{H}$ erzeugt werden, die den Identifikator des Ursprungsknotens und eine laufende Sequenznummer als Eingabewerte nutzt.

Mögliche Kollisionen der Hashfunktion führen im schlechtesten Fall beim Decodieren zu fehlerhaften Nachrichten, die bei Verwendung einer Prüfsumme im Nachrichtenfeld erkannt werden könnten. Kollisionen sollten weitgehend vermieden werden. Dennoch muss für die Hashfunktion eine sinnvolle Abwägung zwischen Komplexität, Kollisionssicherheit und Länge der erzeugten Bitfolge für den Codierungsvektor erfolgen.

Nach dem Erzeugen der $MsgID_i$ wird diese gemeinsam mit der Nachricht M_i in den Zwischenspeichern $\mathcal{B}_{enc}$ und $\mathcal{B}_{atom}$ abgelegt. Somit ist nun sowohl das Erstellen von codierten Paketen, als auch einfacheres Decodieren von Paketen, die eine eigene Nachricht enthalten, möglich.

Der Zweite Weg, um Informationen der Reassembling-Stufe zuzuführen, sind Pakete, die von Nachbarknoten empfangen und anschließend von der Dispatch-Stufe dem Reassembling-Prozess zugeteilt wurden.

Diese Pakete liegen in codierter Form vor, so dass zunächst das Decodieren der enthaltenen Nachrichten stattfinden muss, bevor die Nachrichten zu neu kombinierten Paketen zusammengestellt werden können. Die eintreffenden Pakete werden für das Decodieren zunächst in $\mathcal{B}_{dec}$ bereitgestellt.

Decodierung

Daraufhin erfolgt die Decodierung in der Reassembling-Stufe in zwei alternierenden Teilschritten: *Iterative Resolution* und *Umwandlung atomarer Pakete* in decodierte Nachrichten.

Bei der iterativen Resolution wird zunächst die Codierungskomplexität $c(P)$ schrittweise durch fortgesetztes Kombinieren der empfangenen Pakete mit bereits im lokalen Speicher vorhandenen decodierten Nachrichten und weiteren Paketen reduziert (vgl. Abschnitt 3.2). Das Reduzieren der Codierungskomplexität in der Reassembling-Stufe bei der Resolution ist hier gleichzusetzen mit der schrittweisen Verringerung von $|P|$, also der Anzahl der in einem Paket enthaltenen Nachrichten.

Lokal gespeicherte Pakete, die zur Resolution herangezogen werden können, sind Pakete deren Codierungskomplexität in einer vorangegangenen iterativen Resolution mittels der vorhandenen Informationen vereinfacht wurde, die jedoch nicht vollständig decodiert werden konnten. Alternativ ist auch die Resolution mit Paketen möglich, die zuvor selber nicht vereinfacht oder decodiert werden konnten. Vorzugsweise sollte die Resolution zunächst aber mit decodierten Nachrichten erfolgen, da sie bei der Resolution am geeignetsten die Codierungskomplexität der empfangenen Pakete vereinfachen können, ohne zusätzlichen Speicher zu benötigen.

Das folgende, von OCSSIM verwendete heuristische Verfahren zur schrittweisen Reduzierung der Codierungskomplexität mit Resolutionsschritten, eignet sich insbesondere für Codierungsschemata die auf XOR und einer festen Nachrichtenlänge basieren. Die feste Nachrichtenlänge erfordert die Beschränkung der maximalen Nachrichtenzahl durch den Parameter m, die gemeinsam in einem Paket codiert werden. Im Gegensatz dazu eignet sich für ein Codierungsschema auf Basis von Random Linear Coding (siehe Kapitel 3.3.4) eine Abwandlung des Gaußsches Eliminationsverfahren zum schrittweisen Reduzieren der Freiheitsgrade von Codierungsvektoren. Ein parallelisierbares Verfahren für Random Linear Coding in Sensornetzen und die damit verbundenen Anforderungen zur Rechenleistung werden in [17] diskutiert.

Resolutionsschritt

Sei $P_i := M_{i,1} \oplus ... \oplus M_{i,j}$ mit $j \leq m$ und der $MsgID_i := \{\mathcal{H}(C_{i,j})\}$. Weiterhin sei P_1 ein gerade empfangenes Paket und P_2 ein lokal gespeichertes Paket. Der Resolutionsschritt kann bei der Kombination zweier Pakete $P_{join} = P_1 \oplus P_2$ vier mögliche Ergebnisse hervorrufen.

(a) $|P_{join}| > m$: Das kombinierte Paket P_{join} enthält mehr Nachrichten als die maximal zulässige Paketgröße. Das kombinierte Paket wird verworfen, da sonst die Codierungskomplexität anwächst und die Decoding-Möglichkeiten auf anderen Knoten abnehmen würden. P_1 wird für spätere Resolutionsschritte im Decoding Buffer gespeichert.

(b) $MsgID_1 \cap MsgID_2 = \emptyset$: Die beiden Pakete enthalten keine identischen Nachrichten. So dass die Resolution die Codierungskomplexität nicht reduzieren kann. In diesem Fall kann P_1 nur unverändert weitergeleitet werden und sollte für spätere Resolutionsschritte im Decoding Buffer verbleiben.

(c) $c(P_{join}) = 1$: Eine Codierungskomplexität von 1 ist gleichbedeutend mit dem Vorliegen eines atomaren Paketes. Das kombinierte Paket P_{join} kann decodiert werden und beim Neucodieren mit anderen atomaren Paketen kombiniert werden.

(d) $1 < c(P_{join}) \leq \max\{c(P_1), c(P_2)\}$: Das kombinierte Paket P_{join} enthält eine Nachrichtenkombination, die als solche eine niedrigere Codierungskomplexität hat, als das Paket von P_1 und P_2, mit der größeren Anzahl Nachrichten vor dem Resolutionsschritt. Das kombinierte Paket P_{join} wird als *Paketfragment* im Decoding Buffer für folgende Resolutionsschritte und das Decodieren nachfolgender Pakete gespeichert.

In Abbildung 5.8 ist ein beispielhafter Ablauf einer Resolution mit den verwendeten Paketen aus $\mathcal{B}_{dec}$ bzw. $\mathcal{B}_{atom}$ und der dazugehörigen Resolutionsschritte in OCSSIM dargestellt. Auf die Darstellung der verwendeten Puffer, wurde zur besseren Übersichtlichkeit, in der Abbildung verzichtet.

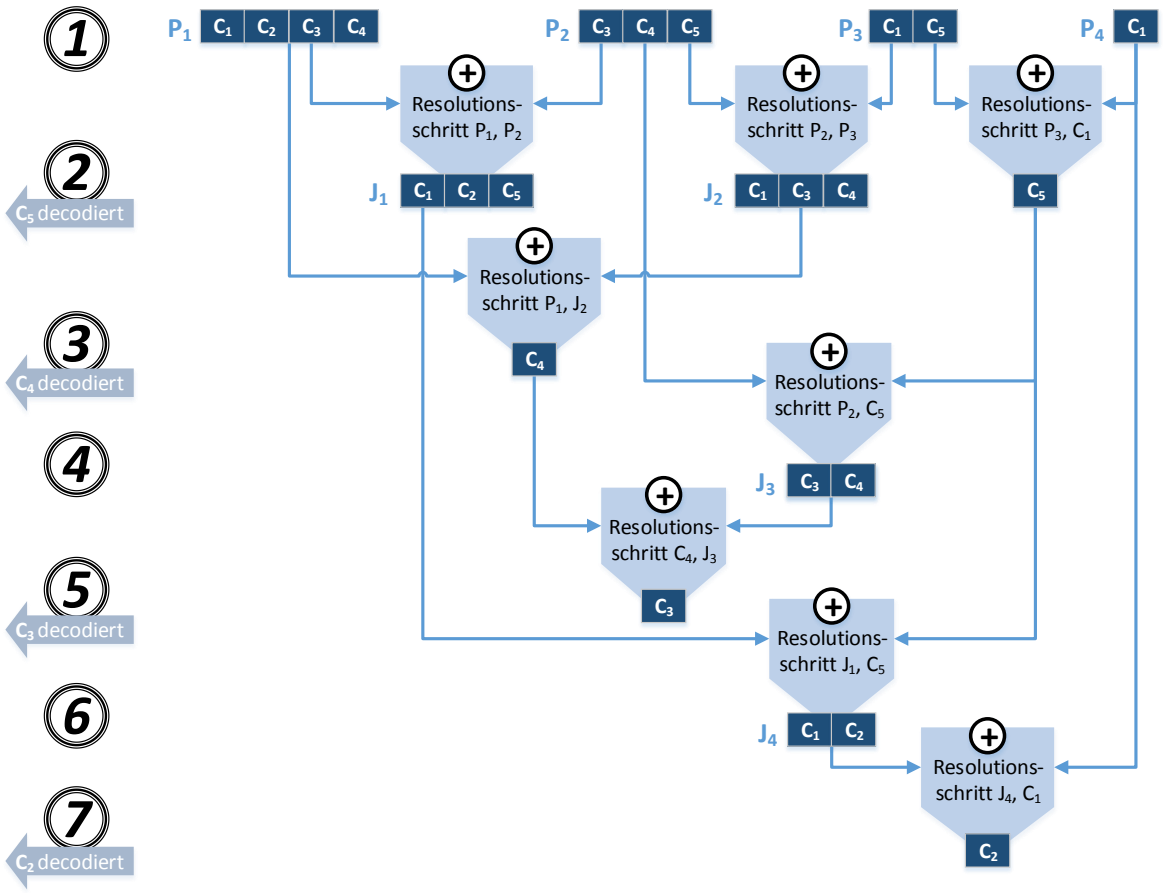

Abbildung 5.8 Exemplarischer Ablauf der iterativen Resolution in OCSSIM

Im Beispiel befinden sich zum Zeitpunkt ① zunächst die Pakete P_1 mit dem Codierungsvektor $[C_1, C_2, C_3, C_4]$, P_2 mit zugehörigem Codierungsvektor $[C_3, C_4, C_5]$ und P_3 mit Codierungsvektor $[C_1, C_5]$ im lokalen Decoding Buffer $\mathcal{B}_{dec}$. Zusätzlich wurde das atomare Paket P_4 bereits in einem vorhergehenden Durchlauf decodiert, so dass sich C_1 im Atomic Buffer $\mathcal{B}_{atom}$ befindet. Die Beschränkung der Nachrichten pro Paket sei $m = 4$. Durch die ersten drei Resolutionsschritte entstehen die mittels XOR kombinierten Pakete J_1, J_2 und ein atomares Paket mit dem Codierungsvektor C_5.

Da die Pakete bei ② noch nicht in $\mathcal{B}_{dec}$ oder $\mathcal{B}_{atom}$ existieren, werden J_1 und J_2 in $\mathcal{B}_{dec}$ und C_5 in $\mathcal{B}_{atom}$ gespeichert. Die zu C_5 gehörende Nachricht liegt zu diesem Zeitpunkt bereits decodiert vor.

Beim Übergang zu ③ erfolgt die Resolution zwischen P_1 und J_2. Es gilt $|P_1 \oplus J_2| = c(P_1 \oplus J_2) = 1$. Das Ergebnis des Resolutionsschrittes ist somit ein atomares Paket wie in Fall (c) angegeben. C_4 kann nun in $\mathcal{B}_{atom}$ gespeichert werden und die dazugehörige Nachricht ist in decodierter Form vorliegend.

Bei ④ werden P_2 und C_5 kombiniert. Das Resolutionsergebnis entspricht Fall (d), so dass die neue Nachrichtenkombination J_3 mit den Codierungsvektoren $[C_3, C_4]$ entsteht. Würde P_2 hier stattdessen mit J_1 kombiniert werden, so würde wieder

das Paket P_1 entstehen, dass sich bereits in $\mathcal{B}_{dec}$ befindet und folglich verworfen würde.

In ⑤ erfolgt die Resolution wieder nach Fall (c), so dass die Nachricht zu C_3 decodiert vorliegt.

⑥ und ⑦ sind Iterationen nach Fall (d) und (c). Am Ende der Resolution liegen in diesem Beispiel sämtliche Nachrichten in decodierter Form vor. Alle verbleibenden Pakete in $\mathcal{B}_{dec}$, die ausschließlich aus einer Kombination von Nachrichten aus $\mathcal{B}_{atom}$ bestehen, können nun aus dem Zwischenspeicher entfernt werden.

Fall (b) kann im gegebenen Beispiel nicht auftreten, da alle Ausgangspakete und Nachrichten jeweils einen gemeinsamen Codierungsvektor besitzen. Dagegen wäre für das Auftreten von Fall (a) im Beispiel erforderlich, dass P_1 und P_2 statt C_3 und C_4 nur einen einzigen gemeinsamen Codierungsvektor besitzen.

Umwandlung atomarer Pakete

Die Umwandlung atomarer Pakete erfolgt jeweils nach einem iterativen Resolutionsschritt mit Ergebnis (c). Bei anderem Resolutionsergebnis wird dieser Schritt übersprungen und mit dem nächsten Resolutionsschritt fortgefahren.

Bekannte Nachrichten müssen nicht erneut decodiert werden und können an dieser Stelle verworfen werden. Das atomare Paket wird dazu mit den bekannten atomaren Paketen in $\mathcal{B}_{enc}$ und $\mathcal{B}_{atom}$ verglichen und ggf. $\mathcal{B}_{atom}$ hinzugefügt.

Bei XOR-Codierung enthalten die Nutzdaten eines atomaren Paketes durch die Resolution bereits nur noch die decodierten Nachrichtendaten. Bei Random Linear Coding ist nun die Lösung des linearen Gleichungssystems eindeutig bestimmbar und erfolgt durch Division der Paketnutzdaten durch den Codierungsvektor im verwendeten Galois-Feld.

Zur Zustellung einer Nachricht an die Anwendungsschicht kann der verwendete Codierungsvektor jeweils anschließend verworfen werden.

Neucodierung und Auswahl der Nachrichtenkombination

Die Auswahl der Nachrichten und Pakete zum Neucodieren einer Nachrichtenkombination erfolgt jeweils automatisch nach Ablauf des Zeitgebers zur Codierung t_{enc} oder bei Erreichen des maximalen Pufferfüllstandes von $\mathcal{B}_{enc}$. Zudem ist auch ein sofortiges Codieren vorgesehen, wenn dazu eine Signalisierung durch eine Anwendung erfolgt, um neue Anwendungsnachrichten mit hoher Priorität im Netzwerk verteilen zu können.

Zur Nachrichtenkombination eines neu codierten Paket werden zufällig Nachrichten des Encoding Buffers ausgewählt. Dabei können beliebige Nachrichten des Encoding Buffers kombiniert werden, unabhängig davon, ob die Nachrichten zuvor decodiert wurden oder neu zu verteilende Nachrichten sind.

Durch einen *Paketmix* lassen sich in OCSSIM auch neue Nachrichtenkombinationen aus mehreren Paketfragmenten aus dem Decoding Buffer oder Paketfragmenten und Einzelnachrichten erstellen. Dabei muss beachtet werden, dass die maximale Anzahl der Nachrichten pro Paket m durch den Paketmix nicht überschritten wird.

Zur Sicherstellung von Decoding-Möglichkeiten in der Nachbarschaft muss bei der Codierung von Paketen eine ausreichend große Anzahl von Kombinationsmöglichkeiten für die Pakete bestehen. Dies kann durch die Größen der Pufferspeicher $|\mathcal{B}_{dec}|$ und insbesondere $|\mathcal{B}_{enc}|$ ermöglicht werden, um einen größeren Kombinationsraum für die Zufallsauswahl zu erreichen. Auch die Erhöhung von t_{enc} sorgt für höhere mittlere Pufferfüllstände und in Folge für einen größeren Kombinationsraum. Dennoch muss bei beiden Maßnahmen eine Abwägung zwischen Decoding-Möglichkeiten und induzierter Latenz bei der Weiterleitung getroffen werden.

Eine weitere Möglichkeit die Decoding-Möglichkeiten zu erhöhen, die in OCSSIM Verwendung findet, ist die dynamische Anpassung der Paketgrößen nach Pufferfüllstand.

Im Allgemeinen steigen die Decoding-Möglichkeiten eines Knotens durch den Empfang von Paketen mit unterschiedlicher Anzahl codierter Nachrichten und unterschiedlicher Codierungskomplexität, da während der Resolution eine größere Anzahl verwertbarer Paketfragmente erzeugt werden kann.

Dabei wird zunächst ein Paket maximal möglicher Größe k erzeugt, das durch $k \leq m$ und die Einträge aus $\mathcal{B}_{enc}$ beschränkt wird. Anschließend werden schrittweise durch Absenken von $k < m$ kleinere Pakete aus den restlichen Einträgen des Zwischenspeichers erzeugt, bis bei $k = 1$ nur noch atomare Pakete erzeugt werden können oder bis $\mathcal{B}_{enc}$ keine Einträge mehr enthält. Sind dagegen bei $k = 1$ noch Einträge vorhanden, wird das nächste Paket wieder mit maximaler Größe m erzeugt.

Erzeugen der Paketkennung und Weiterleitung

Nach der Neucodierung wird eine geeignete Paketkennung erzeugt, die folgende Eigenschaften erfüllt:

Eindeutige $PacketID$**:** Die Identifizierbarkeit des Paketes in der knotenlokalen Nachbarschaft ist u.a erforderlich für die Quittierung des Paketes und zur Vermeidung der Weiterleitung identischer oder selbst initiierter Pakete.

Eindeutige $MsgID_i$ **der enthaltenen** M_i**:** Die Identifikation der einzelnen, im Paket codierten Nachrichten, muss gewährleistet sein, um kombinierbare Pakete zu ermitteln.

Extrahierbare C_i**:** Die Codierungsvektoren der enthaltenen Nachrichten müssen eindeutig ermittelbar sein. Die Zuordnung der C_i zu passenden $MsgID_i$ muss gewährleistet sein. Die Zuordnung der C_i und $MsgID_i$ zur passenden M_i erfolgt dagegen erst bei erneuter Decodierung.

Im Falle der Codierung mit einem XOR-Codierungsschema werden diese Voraussetzungen für die $PacketID$ bereits bei einfacher Konkatenation von $MsgIDs$ mit fester Länge und aufsteigender Reihenfolge erfüllt. Da $MsgID_i$ und C_i in diesem Fall identisch sind, ist auch die letzte Voraussetzung erfüllt.

5.4.4 Filtern eingehender und Verteilung ausgehender Nachrichten

Die Stufe *Local-Dispatch und Nachrichtenfilterung* ist zuständig für das Initiieren der Verteilung von Anfragen und Rückantworten, die von Anwendungen oberhalb von OCCSIM an das Kommunikationsschema weitergereicht werden.

Ausgehende Nachrichten werden an die Reassembling-Komponente weitergereicht, damit diese mit anderen gepufferten Nachrichten kombiniert und codiert werden können. Priorisierte Nachrichten werden zur Codierung eines atomaren Paketes ebenfalls an die Reassembling-Stufe übergeben. Dabei wird aber sichergestellt, dass keine Kombination mit anderen Nachrichten erfolgt und der erforderliche Hop-Wert h zur initialen Verteilung gesetzt wird.

In Gegenrichtung ist die Stufe dafür zuständig, eingehende Nachrichten für Senkenknoten entgegen zu nehmen und an die entsprechende Anwendung weiterzuleiten. Dazu werden erfolgreich decodierte Nachrichten der Reassembling-Stufe entgegen genommen. Anhand der Zieladressierung der Nachrichten muss darauf hin überprüft werden, ob sich der jeweilige Knoten in der Menge möglicher Zielknoten befindet.

Um zu entscheiden, ob ein Knoten zur Menge der Zielknoten gehört, können lokal gespeicherte Informationen zum Status der Nachbarschaft abgefragt werden. Beispielsweise kann ein Knoten die folgenden Informationen über Nachbarknoten regelmäßig ermitteln und lokal speichern:

- Knotengrad
- verbleibende Energiemenge
- Anzahl gesendeter und empfangener Nachrichten von diesem Knoten
- Zeitpunkt des letzten Empfangs
- Signalstärke beim Empfang von Paketen
- Ausstattungsmerkmale des Knotens (z. B. Sensorik, Speichermenge)
- Grad der Aussetzung für eine sensorisch erfasstes Untersuchungsphänomen

Auf diese Weise ist es möglich, eine Menge von Knoten nicht nur anhand eines eindeutigen Identifikators zu adressieren, sondern ggf. auch mittels der gespeicherten Informationen über Knotenzustands- und Topologieinformationen oder Nachbarschaftsbeziehungen. Beispielsweise können zentrale Knoten adressiert werden, die eine festgelegte Mindestanzahl an Nachbarn besitzen oder Knoten, die sich in Reichweite eines interessanten Untersuchungsphänomens befinden und entsprechend exponierte Sensormessungen aufweisen.

5.5 Lokale Weiterleitungsstrategien

Drahtlose Sensornetze umfassen im Allgemeinen eine Vielzahl von Knoten, die ein Beobachtungsgebiet nur gemeinsam Abdecken können. Um Nachrichten im

gesamten Abdeckungsbereich des Netzwerkes transportieren zu können, ist daher das Aufspannen eines Multi-Hop-Netzwerkes die übliche Vorgehensweise zum Datentransport.

In drahtlosen Sensornetzen sind, genau wie in anderen paketvermittelnden Netzen, Forschungsansätze mit *Store-and-Forward*-Strategien vorherrschend. Dabei werden empfangene Dateneinheiten von Zwischenknoten vor dem Weiterleiten lokal gepuffert. Anders als bei *Cut-Through*-Strategien werden die Dateneinheiten dabei jeweils erst mit leichter Verzögerung weiter gesendet. Während dieses Verhalten die Latenz zwischen den kommunizierenden Endsystemen mit jedem Zwischenknoten erhöht, ist aber die Nutzung von ergänzenden Fehlererkennungs- und Korrekturmaßnahmen auf den Zwischenknoten möglich.

Die für Sensornetze typischen Aggregationsverfahren, die Informationen aus mehreren Dateneinheiten in einer einzelnen Dateneinheit zusammenfassen, sind auf Store-and-Forward-Strategien angewiesen, um sequentiell empfangene Dateneinheiten kombinieren zu können. Insbesondere wenn keine dauerhafte Konnektivität zwischen den weiterleitenden Knoten gewährleistet werden kann, wie etwa im Falle von verzögerungstolleranten Netzwerken (DTN, Delay-Tolerant-Networks), bei hoher Mobilität der Knoten oder sehr dynamischer Netzwerktopologie muss ebenso eine zeitweise Pufferung der Daten erfolgen. In all diesen Fällen ist die Nutzung einer Store-and-Forward-Strategie üblich.

Hohe Paketverlustraten auf einzelnen Verbindungen zwischen weiterleitenden Knoten können in drahtlosen Sensornetzen häufig beobachtet werden. Viele Protokolle setzen daher auch Maßnahmen zur Hop-zu-Hop Fehlerkorrektur mit lokal gepufferten Daten ein, um eine erneute Ende-zu-Ende-Übertragung zu vermeiden. Ein typischer Vertreter dieser Protokollgruppe ist PSFQ [109], dessen Mechanismen wesentlich auf einer Store-and-Forward-Strategie beruhen und zur Fehlerkorrektur die Neuübertragung verlorener Pakete von Nachbarknoten zulässt. Im Rahmen des Weiterleitungsvorgangs werden die Pakete durch die Nachbarknoten gepuffert.

Decode-and-Forward-Strategien (vergleiche [107]) wandeln die Verhaltensweise von Store-and-Forward-Strategien ab, indem sie auf codierten Datenströmen arbeiten und vor dem Weiterleiten zunächst das Decodieren, Speichern und Neucodieren der Dateneinheiten voraussetzen. Im Zusammenspiel mit einer Netzwerkcodierung sind Decode-and-Forward-Strategien besonders interessant, sofern die Zwischenknoten Nachrichten decodieren und codieren können. Dies erlaubt sowohl eine Fehlerbehandlung pro Hop, als auch die Möglichkeit die Codierung an die Bedingungen des jeweiligen Netzbereiches zu adaptieren und Verteilungsgrad, sowie die Existenz von vorhandenen Informationen bei benachbarten Knoten einzubeziehen.

Die von OCSSIM verwendete Weiterleitungsstrategie kann als Kombination von Store-and-Forward und Decode-and-Forward-Verfahren betrachtet werden. Dabei wird bei Verwendung der Reassembling-Stufe eine angepasste Decode-and-Forward-Strategie eingesetzt, während die Fast-Forwarding-Stufe eine Store-and-Forward-Strategie verwendet.

Dementsprechend müssen Zwischenknoten auf dem Weiterleitungspfad zwischen Quell- und Senkenknoten in der Lage sein, Pakete zu decodieren und anschließend erneut als Kombination mit weiteren Nachrichten zu codieren. Jeder Zwischenknoten muss außerdem entscheiden, ob ein Paket weitergeleitet wird oder ob codiert enthaltene Nachrichten verworfen werden sollen. Diese Weiterleitungsentscheidung wird wesentlich durch die zwei Stufen *Fast-Forwarding* und *Reassembling* vorbereitet.

5.5.1 Opportunistische Weiterleitungsentscheidung

Die Entscheidung, ob eine Nachricht letztlich weitergeleitet werden soll, findet in OCSSIM *opportunistisch* statt. Das bedeutet, der jeweilige Knoten entscheidet auf Basis seines lokalen Zustands und der lokal verfügbaren Informationen in der vorherrschenden Situation.

Grundlage der Entscheidung in der Stufe *opportunistische Weiterleitungsentscheidung* sind daher die gesammelte Information aus den vorhergehenden Stufen zu Priorisierung, Knotenzustand und der Neukombination und Decodierung von Nachrichten mittels des gewählten Codierungsschemas.

Eine besondere Rolle spielt hier schon die Stufe zur Paketanalyse und Aufteilung eingehender Pakete (siehe Kapitel 5.4.1), die zur Vorbereitung der Weiterleitungsentscheidung beiträgt. Dort ermittelte Informationen werden zur Verwendung für die abschließende Entscheidung bereitgehalten.

Auch der Zustand der Nachbarschaft kann als Entscheidungshilfe dienen, um zu bewerten, ob Pakete bevorzugt weitergeleitet werden müssen. Die Zustandshaltung der Nachbarschaft wird dabei fortlaufend aktualisiert. Dies geschieht vor allem beim Empfang von mitgehörten Paketen, z. B. beim Erkennen von eigenen Paketen die von Nachbarn weitergeleitet wurden.

Das aktive Versenden von zusätzlichen Nachrichten zwischen Nachbarn, um Zustandsinformationen regelmäßig zu aktualisieren, kann die Informationsbasis ergänzen. Dies ist jedoch für die grundlegende Funktionsweise des OCSSIM-Kommunikationsschemas nicht zwingend erforderlich, da bereits passiv durch Overhearing ermittelte Informationen aus dem allgemeinen Nachrichtenverkehr, eine geeignete Entscheidungsgrundlage bieten.

Zur Kontrolle der Nachrichten- und Paketverteilung im Netzwerk wird bei der Weiterleitungsentscheidung auf folgende Maßnahmen zurückgegriffen.

Zurückstellen des Paketes (Back-Off): Das Weiterleiten eines Paketes erfolgt in OCSSIM erst nach einer zufällig gewählten Rückstellzeit. Die Rückstellzeit für Pakete der Fast-Forwarding-Stufe $t_b^F \in \left[t_{min}^F, t_{max}^F\right]$ und die Rückstellzeit der Reassembling-Stufe $t_b^R \in \left[t_{min}^R, t_{max}^R\right]$ sollten im Allgemeinen aus disjunkten Intervallen gewählt werden, so dass $t_{max}^F < t_{min}^R$ gilt, um eine strikte Priorisierung per Fast-Forwarding zu garantieren.

Wird ein bereits zuvor empfangenes Paket vor Ablauf von t_b^F bzw. t_b^R erneut empfangen, so wird die Rückstellzeit um einen festen Wert t_{inc}^F bzw. t_{inc}^R

erhöht. Der Mechanismus dient zur Vermeidung des Sendens redundanter Pakete innerhalb der gleichen Nachbarschaft und einem kurzen Zeitraum.

Unterdrücken bekannter Pakete: Wird ein Paket immer wieder vor der Weiterleitung zurückgestellt, so ist bei Erreichen der Schwelle thr_{drop} davon auszugehen, dass es in der ganzen Nachbarschaft eines Knotens bereits empfangen wurde. Eine erneute Weiterleitung durch den Knoten nach dreimaliger Rückstellung würde nicht zu einer signifikanten Steigerung der Netzwerkabdeckung führen [82] und wird in Folge aus Effizienzgründen in OCSSIM unterlassen.

Kontrolle der Paketmenge: Eine Kontrolle der weitergeleiteten Paketmenge ist mittels Beschränkung der Sendewiederholungen (siehe Kapitel 5.2.2.4) oder mittels Anpassung der Paketgrößen während des Codierens (siehe Kapitel 5.4.3) möglich.

Verwerfen von Paketen mangels Informationsgewinn: Kann bereits vor der Weiterleitung anhand der vorhandenen Informationen ermittelt werden, dass in der Nachbarschaft durch Weiterleitung kein oder nur ein sehr geringer Informationsgewinn zu erwarten ist, kann das Paket auch ohne Weiterleitung verworfen werden. (siehe Kapitel 4.2.1).

Mangels Unicast-Kommunikation ist das gezielte Weiterleiten eines Paketes an einen ausgewählten Empfängerknoten in OCSSIM nicht möglich. Im Gegenzug ermöglicht die Kommunikation mittels Broadcast und Overhearing aber die Verteilung der Pakete über mehrere Pfade durch inhärente Ausnutzung der vorhandenen Alternativpfade in einer Netzwerktopologie.

Wie das Verteilen von Paketen über mehrere Pfade in OCSSIM im Zusammenspiel mit den obigen Maßnahmen zur Weiterleitung erfolgt, zeigt das Beispiel in Abbildung 5.9.

Im Beispiel wird dazu ein festes Paket verwendet, entsprechend der Weiterleitung in der Fast-Forwarding-Stufe. Schritt ① zeigt die Ausgangstopologie und die benachbarten Knoten. Während der farblich hervorgehobene Knoten einen aktiven Sender signalisiert, haben dunkel markierte Knoten das Paket bereits empfangen. Die Verteilung des Paketes P erfolgt ausgehend von Knoten Q zu Knoten S. In Schritt ② beginnt Knoten Q und sendet P an seine Nachbarknoten Z_1, Z_2 und Z_3. Jeder dieser Empfängerknoten wählt einen zufälligen Back-Off und vermerkt den ersten Empfang von P. In Schritt ③ endet der Back-Off des Knotens Z_3 der das Paket per Broadcast an Q, Z_1 und Z_4. Der Knoten Q verwirft das empfangene Paket sofort, da es bereits durch Q versendet wurde. Knoten Z_1 stellt fest, dass das Paket ein weiteres Mal empfangen wurde. Daraufhin wird der Zeitgeber für das Rückstellen der Weiterleitung erhöht. Z_4 erhält P das erste Mal und wählt auf Zufallsbasis einen sehr kleinen Back-Off, so dass der Knoten in Schritt ④ die Weiterleitung fortsetzt. Hier erhält Z_1 das Paket zum dritten Mal und erhöht

wiederum die Rückstellzeit, während Z_3 das Paket direkt verwirft. Das Weiterleiten in Schritt ⑤ führt schließlich dazu, dass Z_1 das Paket verwirft, da $thr_{drop} = 3$ überschritten wurde. Der Knoten S erhält dennoch das gewünschte Paket von Knoten Z_2, so dass der Knoten Z_1 in Schritt ⑥ keine weitere Aktion durchführen muss, um die Verteilung an seinen Nachbarn S sicherzustellen. Grundlage hierfür ist die engmaschige Topologie und die Existenz mehrerer Pfade zwischen Q und S, die durch das Verteilungsverfahren adaptiv ausgenutzt wird.

5.6 Zusammenfassung

In diesem Kapitel wurden die Kernproblematiken erläutert, die bei der opportunistischer Weiterleitung von Paketen auftreten, wenn Netzwerkcodierung dazu genutzt wird, codierte Pakete aus Kombination von Einzelnachrichten zu erzeugen.

Neben dem Verhalten bei der Initialisierung der Verteilung neuer Nachrichten im Netzwerk, ist hier das Sättigungsverhalten mit redundanten Nachrichten zu beachten. Zusätzlich ist die Quittierung von codierten Nachrichten bei Broadcast-Kommunikationsszenarien und die zusätzliche Latenz durch das Decodieren und Codieren der verteilten Nachrichten auf Zwischenknoten eine der Kernproblematiken.

Den Hauptteil des Kapitels bildet das OCSSIM-Kommunikationsschema, welches auf dem vorgestellten Konzept zur opportunistischen Weiterleitungsentscheidung gründet und die Verteilung der Nachrichten mittels Netzwerkcodierung koordiniert. In dieser Arbeit wurden dazu jeweils geeignete Verfahren und Maßnahmen entwickelt, um Lösungen für die identifizierten Kernproblematiken im Kommunikationsschema zu realisieren. Lokale Weiterleitungsentscheidungen in OCSSIM werden lediglich auf Grundlage von Beobachtung der Nachbarschaft und Analyse der Paketeigenschaften getroffen, so dass kein weiteres Routingverfahren verwendet werden muss.

Dazu wurde ein heuristisches Verfahren entworfen, das es OCSSIM ermöglicht, trotz codierter Nachrichten und vorherrschender Broadcast-Kommunikation implizite Quittungen auszunutzen, um Sendewiederholungen zu koordinieren. Zusammen mit der Eigenschaft, Nachrichten über unterschiedliche Zwischenknoten an einen Empfänger zu verteilen, ermöglicht es OCSSIM dabei eine sehr hohe Netzwerkabdeckung bei der Nachrichtenverteilung aufrechtzuerhalten.

Das zentrale OCSSIM-Pipeline-Modell mit getrennter Fast-Forwarding- und Reassembling-Stufe gewährleistet dabei, dass Nachrichten bei der Verteilung sowohl priorisiert werden können, als auch in effizienter Weise als Nachrichtenkombination codiert und decodiert werden können. Die mehrstufige Behandlung eingehender Pakete von Nachbarn ergänzt und erweitert hier übliche Netzwerkcodierungs-Verfahren um opportunistische Eigenschaften zur Regulierung der Nachrichtenverteilung auf unbekannten und stark dynamischen Netzwerktopologien.

Durch Adaption von Paketgrößen und Weiterleitungsentscheidungen, wie Unterdrückung von Paketen in der Nachbarschaft oder Verwerfen redundanter In-

formation, erzeugt OCSSIM die nötige dynamische Kooperation zwischen den Knoten, um effektiv Decoding-Möglichkeiten der codiert verteilten Nachrichten im gesamten Netzwerk zu schaffen.

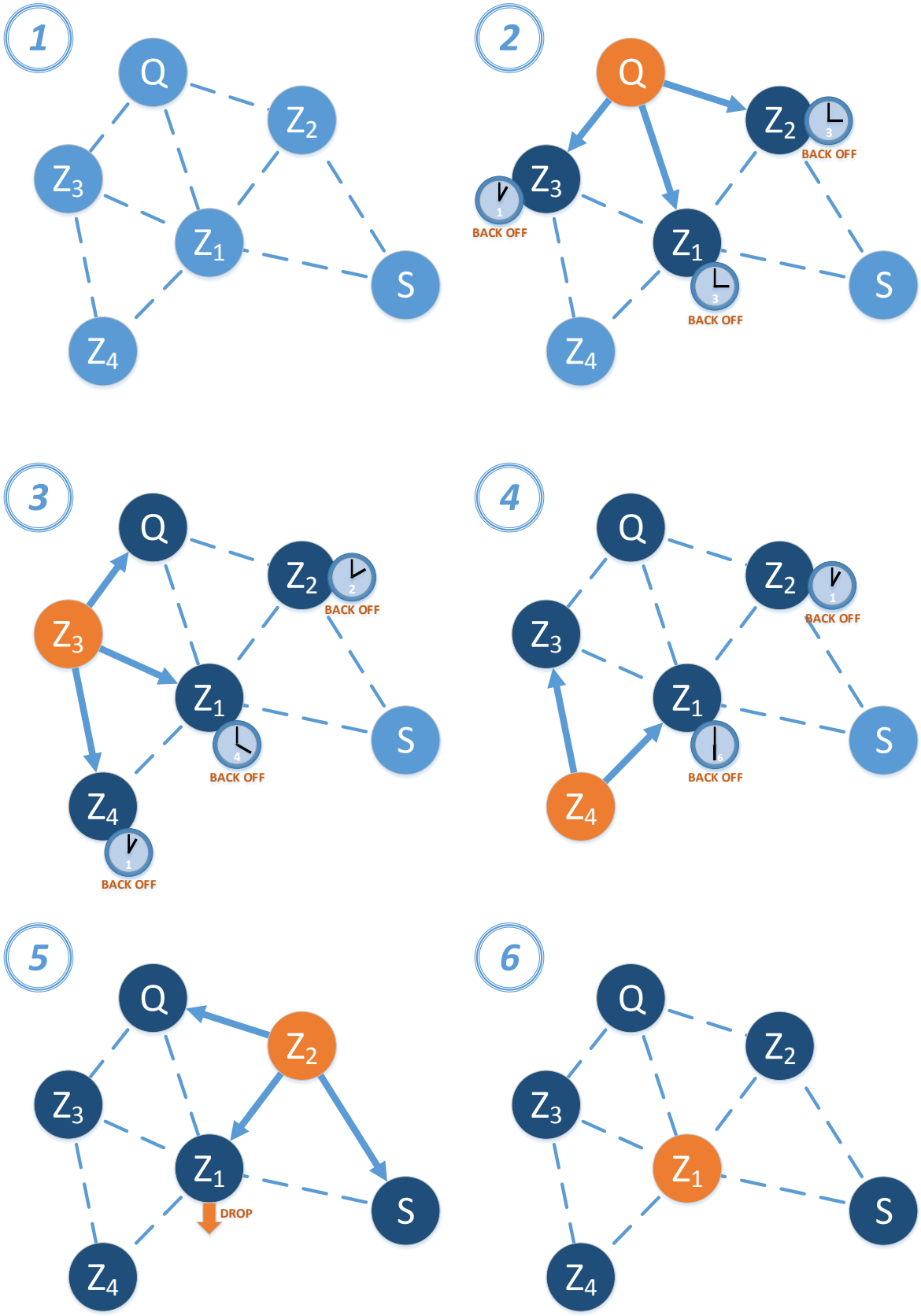

Abbildung 5.9 Beispiel: Verbreitung von Paketen per Weiterleitungsentscheidung in OCSSIM

6. ROSMARIN-Rahmenwerk

Ein Großteil dieser Arbeit beschäftigt sich mit den Anforderungen und Realisierungsmöglichkeiten von OCSSIM auf Hardwareplattformen für drahtlose Sensornetze. In diesem Kapitel wird ein Rahmenwerk eingeführt, das eine prototypische Realisierung des entworfenen Kommunikationsschemas (siehe Kapitel 5) umsetzt.

Das entwickelte **R**ahmenwerk für **o**pportunistische **S**ensornetz-**M**ulticast-**A**nwendungen und **r**edundante **I**nformationsverteilung mittels **N**etzwerkcodierung, kurz *ROSMARIN*, erlaubt dabei eine gezielte Betrachtung des Zusammenspiels der in OCSSIM eingesetzten Konzepte und Kommunikationsmuster.

Im Vordergrund steht hier die Simulation und Evaluation der Rekonstruktionseigenschaften bei Netzwerkcodierung. Daneben werden Speicherbedarf und Robustheit des Kommunikationsschemas bei Topologieänderungen, z. B. durch Änderung der Knotenanzahl oder Knotendichte betrachtet. Weitere Merkmale für die Leistungsfähigkeit des Kommunikationsschemas sind der erzielbare Datendurchsatz zwischen Quellen und Senken, sowie die dabei erzeugte Latenz.

Nach der Einführung und Einordnung der Architektur und Realisierungsumgebung des Rahmenwerkes erfolgt in diesem Kapitel eine genauere Betrachtung, wie die Prinzipien des OCSSIM-Kommunikationsschemas realisiert werden können. Im Anschluss werden die zur Realisierung benötigen Cache- und Speicherstrukturen betrachtet.

Im zweiten Teil des Kapitels liegt der Schwerpunkt auf der Evaluation des Kommunikationsschemas und der verwendeten Evalutionswerkzeuge. Anhand ausgewählter Evaluationsszenarien wird dabei das Kommunikationsverhalten in Multi-Hop-Sensornetzen analysiert.

Zweck des Rahmenwerkes

Das Rahmenwerk bildet die Grundlage zur Realisierung von Anwendungen, die Netzwerkcodierung und opportunistische Weiterleitung auf Netzwerk- und Transportschicht aktiv ausnutzen.

Neben der funktionalen Realisierung der Methoden und Prinzipien des Kommunikationsschemas, sind in das Rahmenwerk weitere, ergänzende Funktionen und Werkzeuge integriert.

Im Einzelnen sind dies:

- Monitoring- und Analysewerkzeuge zur Protokollierung des Knotenzustands und der Nutzung der eingesetzten Zwischenspeicher zur Netzwerkcodierung während der Simulationslaufzeit
- Bereitstellung und Austausch von Informationen des Knotenstatus in der lokalen Nachbarschaft
- Adaptionsmöglichkeiten zur Abstimmung der Zeitpunkte zur Weiterleitung und Netzwerkcodierung
- Mechanismen zur Filterung decodierter Nachrichten zur Zustellung von Multicast-Nachrichten auf den Zielknoten
- Schnittstelle zum Austausch der Schicht zum Medienzugriff, basierend auf der *Unified Radio Power Management Architecture (UPMA)* [54]
- Anwendungsschnittstelle zur netzwerkweiten Verteilung von Nachrichten

Durch Deaktivieren der integrierten Evaluationswerkzeuge kann die gleiche Implementierung auch zur Nutzung in realen Einsatzbedingungen verwendet werden. Somit können ROSMARIN-Anwendungen nicht nur zur simulativen Evaluation genutzt werden, sondern auch auf realen Sensorknoten ausgebracht werden.

6.1 ROSMARIN im Protokollstapel

Abbildung 6.1 zeigt die Einordnung des ROSMARIN-Rahmenwerks in den typischen Protokollstapel eines drahtlosen Sensorknotens.

ROSMARIN ist als kombinierte Netzwerk- und Transportschicht realisiert, die sowohl alternativ, als auch parallel zu anwendungsspezifischen Transport- und Netzwerkschichten verwendet werden kann.

Dazu nutzt ROSMARIN die bereitgestellten Funktionen zum Medienzugriff und bietet eine Anwendungsschnittstelle zur Verteilung von Daten und Zustellung von Nachrichten. Diese kann unmittelbar von einer Sensornetzanwendung genutzt werden oder durch eine zusätzliche Abstraktionsschicht virtualisiert werden, die den Zugriff auf die vorhandenen Knotenressourcen wie z. B. Speicherplatz, aber auch die Datenrate oder die unterschiedlichen anwendungsspezifischen Datensenken und externen Schnittstellen der Sensornetzanwendungen koordiniert. Ebenso

kann neben den Sensornetzanwendungen eine Anwendung zum Monitoring des Netzwerkzustands realisiert werden, die Zustandsinformationen aus opportunistisch weitergeleiteten Nachrichten extrahiert und auswertet. Eine prototypische Vorgehensweise zur Realisierung eines Dienstes zum Netzwerkmonitoring auf Grundlage von ROSMARIN wurde im Rahmen einer Diplomarbeit [103] realisiert.

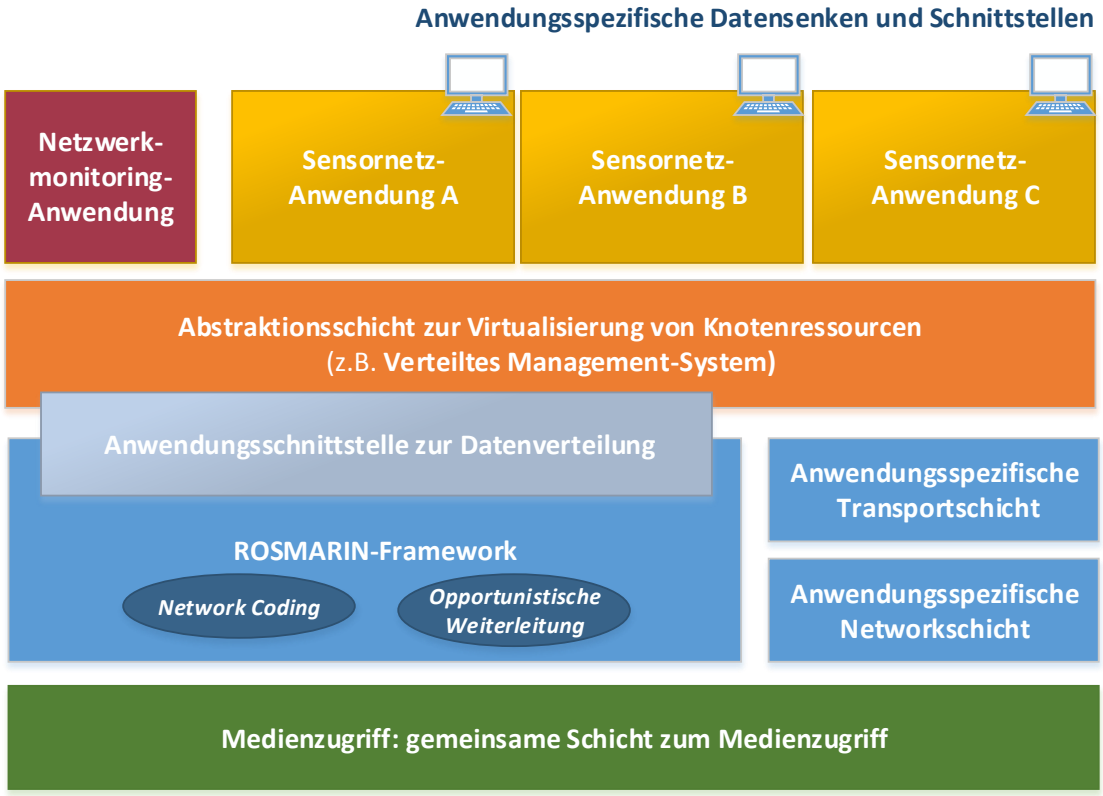

Abbildung 6.1 Einordnung in existierende Protokollstruktur

6.2 Realisierungsumgebung und Evaluationswerkzeuge

Zur Evaluation des OCCSIM-Kommunikationsschemas wurde das ROSMARIN-Rahmenwerk für *TinyOS 2.x* [64] und Sensorknoten des Typs *Iris* [78] implementiert. Die Auswertung erfolgte mittels Simulation eines Sensornetzes durch den Emulator *Avrora*.

TinyOS

TinyOS ist ein Betriebssystem für drahtlose Sensorknoten, das darauf optimiert wurde, auf Sensorknoten mit geringen Hardwareanforderungen ausgeführt zu werden.

Der strikt modulare Aufbau von TinyOS erlaubt es, die von einer Anwendung benötigte Funktionalität bereitzustellen, ohne bedeutenden zusätzlichen Speicherbedarf für nicht verwendeten Programmcode zu erzeugen. Außerdem kann somit

die Wiederverwendung und Anpassung der gewünschten Funktionalität in unterschiedlichen Anwendungen und auf unterschiedlicher Sensorknoten-Hardware ermöglicht werden.

TinyOS wurde sowohl hinsichtlich eines sehr geringen Speicherverbrauchs, als auch hinsichtlich einer hohen Energieeffizienz optimiert. Einzelne Hardwarekomponenten, wie der Funk-Transceiver eines Sensorknotens, lassen sich daher gezielt durch TinyOS aktivieren und deaktivieren, um den Energiebedarf im laufenden Betrieb zu verringern und die Lebenszeit eines Sensorknotens somit zu verlängern.

Der von TinyOS verwendete *nesC*-Compiler basiert auf einem Dialekt der Programmiersprache C und ergänzt diese insbesondere um Programmierkonstrukte zur ereignisbasierten Steuerung der Sensorknoten. Sensorknoten können dadurch auf Ereignisse, wie das Eintreffen neuer Pakete oder festgelegte Messergebnisse der Sensorik gezielt reagieren. Für die zeitgesteuerte Signalisierung von Ereignissen werden durch TinyOS Zeitgeber mit unterschiedlicher Präzision und Auflösung als Module bereitgestellt. Die jeweils verfügbaren Zeitgeber richten sich nach der verwendeten Sensorknoten-Hardware.

Nebenläufige Ereignisse lassen sich mit TinyOS nicht realisieren. Alle nötigen Tasks werden in sequentieller, nicht deterministischer Reihenfolge durchgeführt. Zur Koordination können stattdessen sogenannte asynchrone *Split-Phase*-Operationen durchgeführt werden, die nach Durchführung eines Tasks die ein Ereignis signalisieren, um die Steuerung der Programmlogik an das gewünschte Modul zurückzugeben.

Für die Umsetzung der OCSSIM-Pipeline bedeutet das, dass Fast-Forwarding und Reassembling unabhängig von einander realisiert werden müssen, ohne dass die Reihenfolge der Behandlung eintreffender Pakete in diesen Modulen einen Einfluss auf die Weiterleitungsentscheidung hat.

Iris-Sensorknoten

Also Sensorknoten-Hardware wurden in dieser Arbeit Knoten des Typs *Iris* [78] stellvertretend für typische Sensorknoten mit stark limitiertem Speicherkontingent ausgewählt. Diese werden nativ von TinyOS unterstützt und verfügen über 8 Kilobyte RAM und eine 8-Bit Architektur. Als Verarbeitungseinheit wird ein *Atmel ATmega1281*-Mikrocontroller mit einer Taktfrequenz von 16 MHz verwendet und durch den IEEE 802.15.4 Funk-Transceiver *Atmel AT86RF230* mit einer Datenrate von 250 Kilobit pro Sekunde ergänzt. Die Iris-Sensorknoten ermöglichen mit TinyOS den Zugriff auf 2 Zeitgeber mit einer 8-Bit-Auflösung und 4 Zeitgeber mit 16-Bit-Auflösung und einer Präzession von 32 kHz.

6.2.1 Emulator Avrora

Avrora [102] ist ein auf Java basierender Emulator, der zyklengenau die Instruktionen des 8-bit AVR-Mikrocontrollers des *Mica2*-Sensorknotens durchführen kann. Darüber hinaus kann Avrora ein drahtloses Netzwerk von bis zu 10000 emulierten Sensorknoten einschließlich des Kanalmodells simulieren. Die Anzahl der simulierten Knoten wurde in dieser Arbeit auf 100 Knoten beschränkt. Das erlaubt es,

Aussagen über das Verhalten des Kommunikationsschemas zu treffen, ohne dass das erzeugte Datenvolumen und die Laufzeit der Simulation eine erheblich komplexere Analyse erfordern. Zur Unterstützung des moderneren *MICAz*-Sensorknotens und des CC2420-Funkmoduls wurden der Emulator in der Arbeit [19] zu *AvroraZ* erweitert.Eine weitere Anpassung für die hier verwendeten Sensorknoten vom Typ *Iris* wurde im Rahmen einer Bachelorarbeit [76] durchgeführt.

6.2.2 Testbett

Zur Auswahl geeigneter Evaluationswerkzeuge wurden im Rahmen dieser Arbeit bestehende Testbett-Projekte und Simulationsumgebungen in [42] verglichen.

Insbesondere wurden anhand der unterschiedlichen Forschungsschwerpunkte Energieeffizienz, Mobilität, realistische Umgebungssimulation, Skalierbarkeit und Leistungsmessung, Hardwaredesign der Sensorknoten, Interoperabilität, sowie Anwendungs- und Protokolldesign, verschiedene Entwurfselemente und Vorgehensweisen zur Entwicklung spezifischer Testwerkzeuge identifiziert.

Aufgrund dieser Entwurfselemente wurde den integrierten Evaluationswerkzeugen des ROSMARIN-Rahmenwerks eine Datenstruktur zur Protokollierung und Signalisierung von auftretenden Programmereignissen hinzugefügt. Diese erlaubt es, die Simulationsumgebung zur Ausgabe des Knotenzustands während der Laufzeit zu nutzen und dabei den zusätzlichen Speicheraufwand für die Zustandshaltung auf den Sensorknoten zu begrenzen (siehe Abschnitt 6.5.1).

Im Rahmen der Arbeiten [36, 37] wurde mittels dieser Vorgehensweisen außerdem ein eigenes Testbett zur verteilten Energiemessung in drahtlosen Sensornetzen entwickelt. Aufgrund einer Beschränkung der Knotenanzahl, statischer Topologie und des erforderlichen zeitlichen und materiellen Aufwands zur Ausbringung und Protokollierung der durchgeführten Versuche wurde die nachfolgende Evaluation jedoch vollständig mittels des Netzwerksimulators Avrora durchgeführt. So ist es insbesondere möglich, Aussagen zur Skalierbarkeit des Kommunikationsschemas zu treffen.

6.3 OCSSIM in ROSMARIN

In diesem Abschnitt werden die Entwurfsentscheidungen bei der prototypischen Umsetzung des OCCSIM-Kommunikationsschemas im ROSMARIN-Rahmenwerk diskutiert.

6.3.1 Umsetzung der OCSSIM-Message-Pipeline im ROSMARIN-Rahmenwerk

Abbildung 6.2 zeigt einen Überblick des Datenflusses und der am Weiterleitungsprozess beteiligten Hauptkomponenten. Diese Komponenten sind auf jedem am Prozess beteiligten Knoten realisiert.

OCSSIM Pipeline-Stufe	ROSMARIN-Komponente	TinyOS-Modul
Paketanalyse und Aufteilungs-Entscheidung	Opportunistischer Transceiver	`NWCodeMsgTransceiver`, `DispatchReassembling`, `DispatchFastForward`
Network Coding	Netzwerkcodierung	`XORCoding`
Reassembling	(Re-)Assembling und Nachrichtenverteilung	`DispatchReassembling`
Fast-Forwarding	Fast-Forwarding	`DispatchFastForward`
Priorisierung	Opportunistische Weiterleitungsentscheidung	`DispatchFastForward`
Knoten-und Nachbarschaftszustand	Knoten- und Nachbarschaftszustand	`NodeStatus`, `NeighborDegree`
Local-Dispatch und Nachrichtenfilterung	Nachrichtenzustellung und Filterung, (Re-)Assembling und Nachrichtenverteilung	`MMDecide`, `DispatchReassembling`
Opportunistische Weiterleitungsentscheidung	Opportunistischer Transceiver, Opportunistische Weiterleitungsentscheidung	`NWCodeMsgTransceiver`, `MsgValue`

Tabelle 6.1 Übersicht der Pipline-Stufen, ROSMARIN-Komponenten und TinyOS-Module

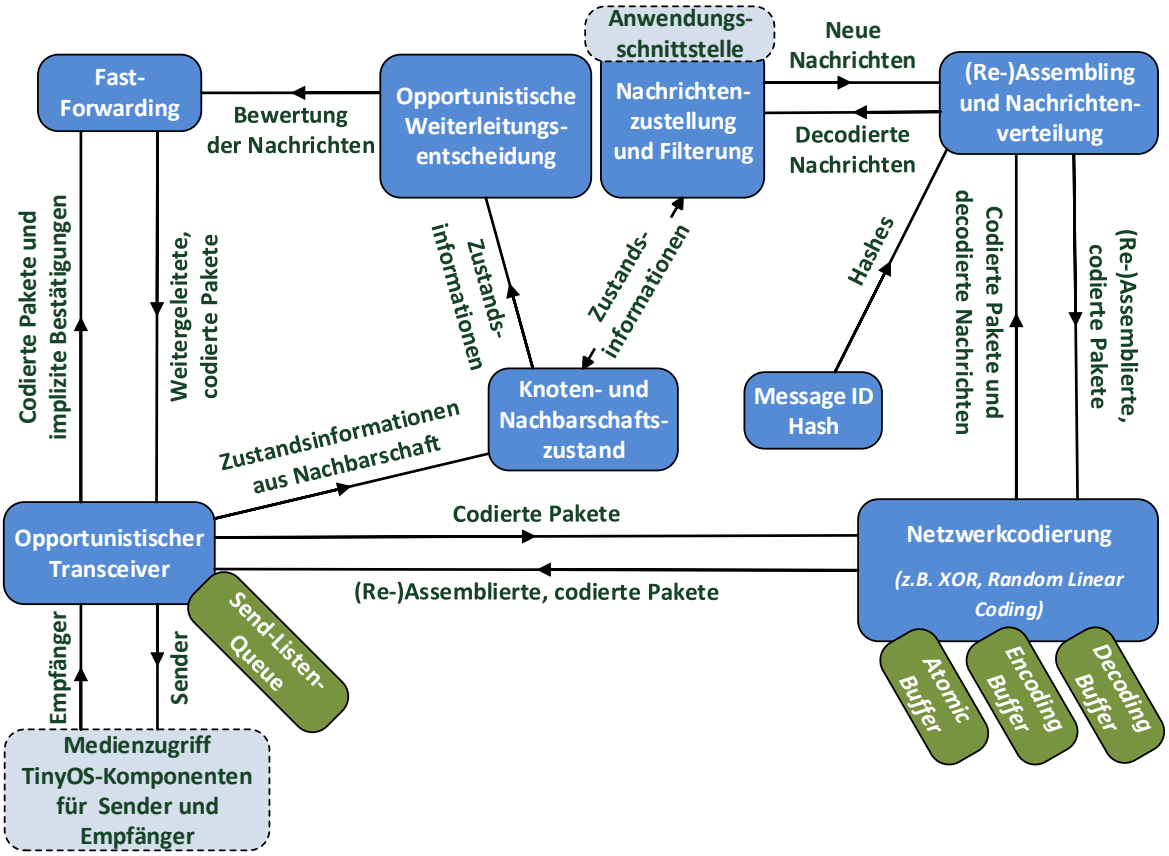

Abbildung 6.2 Übersicht der verwendeten Komponenten zur Weiterleitung von Nachrichten

Der modulare Entwurf von ROSMARIN erlaubt es, einzelne Komponenten auszutauschen oder die bestehenden Komponenten auf einfache Weise anzupassen, so dass beispielsweise verschiedene Codierungsschemata verglichen werden können.

Im Folgenden wird das Verhalten und Zusammenspiel der einzelnen Komponenten und die Entsprechung in der OCSSIM-Message-Pipeline erläutert.

Das in Kapitel 5.4 eingeführte Pipeline-Verarbeitungsmodell (vgl. Abbildung 5.5), bildet das zentrale Komponentengerüst des Rahmenwerks aus Abbildung 6.2 ab, indem eine Trennung der Fast-Forwarding-Stufe und Reassembling-Stufe in den TinyOS-Modulen `DispatchFastForward` bzw. `DispatchReassembling` vorgesehen wurde. Bei Eintreffen eines neuen Paketes signalisiert der Opportunistische Transceiver (`NWCodeMsgTransceiver`) beiden Modulen den Empfang. Die Paketanalyse und Aufteilung der eintreffenden Pakete erfolgt in ROSMARIN nicht in einem eigenen Modul als Aufteilungsstufe, sondern innerhalb von `DispatchFastForward` und `DispatchReassembling`, indem Codierungskomplexität c und Hop-Zähler h parallel ausgewertet werden und jeweils nicht benötigte Stufen selbstständig die Weiterleitung vermeiden.

Die Funktionalität zur Netzwerkcodierung wurde in ROSMARIN für das genutzte Codierungsschema in ein eigenständiges Modul (z. B. `XORCoding`) ausgegliedert,

um einfache Austauschbarkeit unabhängig von `DispatchReassembling` zu ermöglichen. Im Gegensatz dazu wurde die Nachrichtenpriorisierung in `DispatchFastForward` integriert.

Die Weiterleitungsentscheidung und Auswahl der Zielknoten erfolgt im Modul für den Opportunistischen Transceiver `NWCodeMsgTransceiver` unter Nutzung der jeweils implementierten Metrik und Strategie über die Schnittstelle `MsgValue`. Dies wird zum Beispiel im Modul `NeighborDegree` genutzt, das den Knotengrad der Nachbarn als Entscheidungsgrundlage heranzieht. Der dazu verwendete Knoten- und Nachbarschaftsstatus wird im gesonderten Modul `NodeStatus` gesammelt. Der Opportunistische Transceiver koordiniert somit eingehende und ausgehende Pakete und erleichtert die Einführung und Nutzung eines gemeinsamen Zwischenspeichers, der sowohl als Cache, als auch als Filter für redundante Nachrichten verwendet werden kann. Zudem bildet das Modul eine einheitliche Schnittstelle als Übergabepunkt zur Medienzugriffsschicht.

Die Zustellung von Nachrichten an die Anwendungsschicht übernimmt das Modul `MMDecide`. Es kooperiert dazu eng mit `DispatchReassembling`, welches das erfolgreiche Decodieren von Nachrichten signalisiert und neue Nachrichten der Anwendungsschicht zwischenspeichert, um diese initial zu codieren und mit Nachrichten des Encoding Buffers zu kombinieren.

Tabelle 6.1 bietet eine Übersicht der hier erläuterten TinyOS-Module und ihrer Entsprechung in der OCSSIM Pipline-Stufe bzw. als ROSMARIN-Komponente.

6.3.2 Bereitstellung von Schnittstellen

ROSMARIN bietet eine Reihe von Schnittstellen zur Verteilung und Zustellung von Anwendungsdaten im Sensornetz, sowie zur Kontrolle der genutzten Komponenten.

- `MsgDelivery`: Schnittstelle zur Priorisierung, Verteilung und Zustellung von Anwendungsnachrichten. Zur aktiven Filterung von Nachrichten und Informationen aus einem mitgehörten Datenstrom sind Erweiterungen der Schnittstelle um Funktionen zum Abonnieren und Publizieren von Nachrichten möglich. Die Schnittstelle kann z. B. zur Informationssammlung in Netzwerkmonitoringanwendungen genutzt werden und wurde in [103] prototypisch umgesetzt.
- `ImplicitAck`: Umfasst die Signalisierung von impliziten Bestätigungen für Nachrichten und Pakete, sowie die Signalisierung von Sendewiederholungen und mehrfach von Nachbarn empfangenen Paketen.
- `NodeStatusChange`: Signalisiert Änderungen des Knoten- und Nachbarschaftszustands, sowie die Erkennung neuer Nachbarknoten.
- `ComponentCtrl`: Erlaubt das gezielte Aktivieren und Deaktivieren von einzelnen Komponenten, z. B. zur Vermeidung von Weiterleitung mittels Fast-Forwarding.

6.3.3 Codierung und Decodierung

Codierungsschema

Abbildung 6.3 zeigt das Codierungsschema, wie es in ROSMARIN angewendet wird. Dazu sind die jeweiligen Dateneinheiten und die enthaltenen Datenfelder abgebildet die bei der Codierung verwendet werden. Über den Dateneinheiten ist die Größe der jeweiligen Datenfelder in Bytes gekennzeichnet.

Die Dateneinheiten für die decodierten Nachrichten, die bei der Codierung verwendet werden bestehen aus einem 20 Byte großem Datenfeld, das mit einer Nachricht aus dem Encoding Buffer gefüllt wird. Die Dateneinheit wird durch ein ein 2 Byte langes Feld ergänzt, dass die Quelle der Nachricht kennzeichnet. ROSMARIN verwendet dazu die Knoten ID des Quellknotens. Zusätzlich wird die Dateneinheit durch eine fortlaufende Sequenznummer ergänzt, die dafür sorgt, dass mehrere Nachrichten aus der gleichen Quelle unterschieden werden können. Insgesamt ist eine Dateneinheit für decodierte Nachrichten daher 24 Bytes groß.

Ein codiertes Paket ist mit insgesamt 28 Bytes etwas länger und enthält neben einem 20 Byte großen Datenfeld vier jeweils 2 Byte große Felder für kennzeichnende IDs des Paketinhaltes, die durch Hashing erzeugt werden.

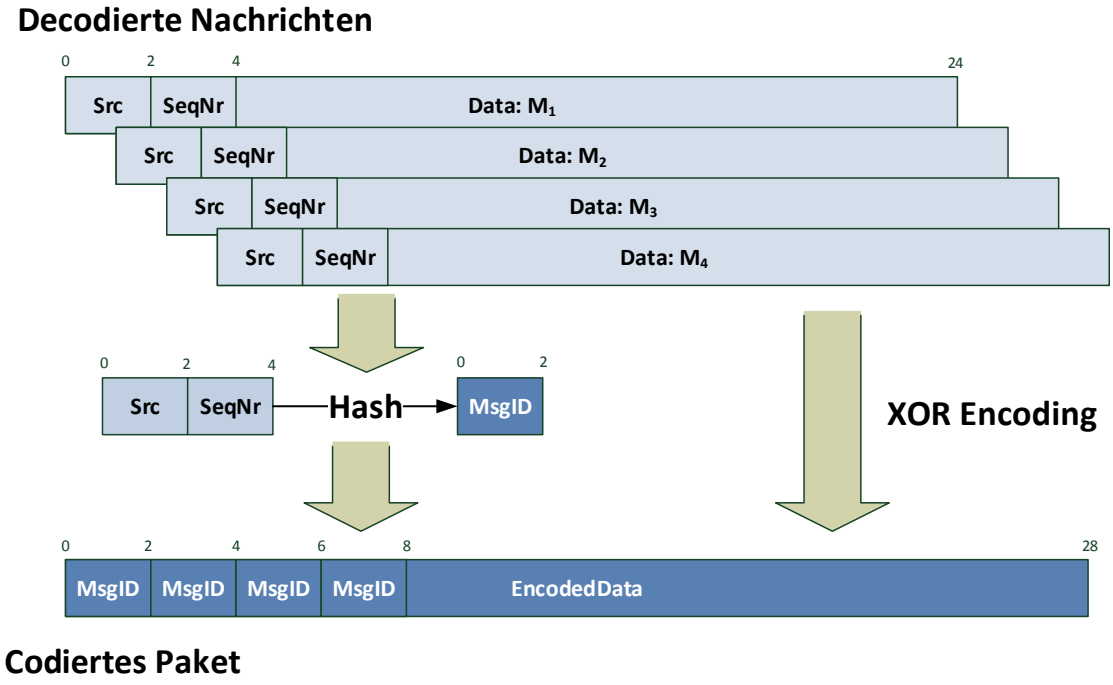

Abbildung 6.3 Codierungsschema mit Hashed Message IDs zur Verringerung der Paketgröße

Kennzeichnung der Nachrichten durch Hashing

Die Identifikation der codierten Nachrichten innerhalb eines Paketes kann bei Verwendung des XOR-Codierungsschemas mittels eines Tupels lediglich aus der

ID der Quelle und der fortlaufenden Sequenznummer erfolgen. ROSMARIN nutzt hier einen einfachen multiplikativen Hash $\mathcal{H}$ im Modul `MessageIDHash`, um die MessageID auf eine Größe von 2 Bytes je Nachricht zu beschränken. Multiplikative Hashfunktionen zeichnen sich durch ihre einfache und schnelle Berechenbarkeit aus, weisen aber keine kryptographischen Eigenschaften, wie hohe Kollisionsresistenz auf. Da eine Hashkollision hier durch erneutes Senden von zusätzlichen Paketen entsprechend der verwendeten Codierungskomplexität ausgeglichen werden kann, ist eine multiplikative Hashfunktion aus Effizienzgründen vorzuziehen. Bei dem verwendeten üblichen Nachrichtenformat für TinyOS mit einer Nutzdatenmenge von 28 Bytes und einer Codierungskomplexität $c = 4$ stehen durch das Hashing der MessageIDs 14% mehr Nutzdaten pro Paket zur Verfügung.

6.3.4 Virtualisierung der Zeitgeber zur Weiterleitungsentscheidung

Grundlage der opportunistischen Weiterleitung in OCSSIM ist die in Kapitel 5.5.1 erläuterte Vorgehensweise, die den Sendezeitpunkt und das Unterdrücken von Nachrichten koordiniert.

Aufgrund der Hardwarearchitektur der verwendeten Iris-Sensorknoten und der Zeitgeber-Abstraktion in TinyOS (siehe Abschnitt 6.2) stehen nur 4 logische Zeitgeber mit einer Auflösung von 16 Bit pro Knoten zur Verfügung.

Das OCSSIM-Kommunikationsschema benötigt jedoch individuelle Zeitgeber für alle opportunistisch weiterzuleitende Pakete P_i, um das Senden redundanter Pakete in der Nachbarschaft zurückzustellen oder zu unterdrücken.

ROSMARIN realisiert Zeitgeber für individuelle Pakete und den Zeitpunkt des Codierens, indem statt einzelnen Zeitgebern ein relativer Zeitversatz $t_{offset}(P_i)$ zu einem gemeinsamen Zeitgeber $T_{ForwardingClock}$ verwaltet wird.

Da TinyOS als ereignisbasiertes System für zeitgesteuerte Ereignisse die Einstellung des Zeitgebers vor dem nächsten Ereignis erfordert, muss hier iterativ vorgegangen werden. Dazu wird beim Auslösen eines Zeitgeber-Ereignis jeweils der verstrichene Zeitraum seit dem letzten Ereignis $t_{eventdiff}$ subtrahiert. Anschließend wird das erneute Auslösen des Zeitgebers auf $\min(t_{offset}(P_i))$ eingestellt und jedes Paket mit $t_{offset}(P_i) \leq 0$ weitergeleitet.

Treffen zwischen zwei Zeitgeber-Ereignissen neue Pakete ein, die für die Reihenfolge der Weiterleitungszeitpunkte relevant sind, so muss der Zeitgeber zu diesem Zeitpunkt angehalten werden, und die bestehende Ereignisfolge muss um $t_{eventdiff}$ aktualisiert werden.

Neben dem Zeitpunkt zur Weiterleitung durch den zufälligen Backoff t_b werden auch Sendewiederholungen t_{ret} und das Zurückstellen und Unterdrücken der Pakete durch diese Virtualisierung des Zeitgebers $T_{ForwardingClock}$ realisiert.

Das Verlängern von $t_{offset}(P_i)$, z. B. beim Eintreffen identischer Nachrichten, ist jederzeit möglich, ohne den Zeitgeber anhalten zu müssen.

6.3.4.1 Fast-Forwarding und Priorisierung von Nachrichten

Sollen neue Nachrichten zwischen einer Quelle und Senke übertragen werden, so wartet zunächst jeder einzelne Knoten auf eingehende Pakete seiner Nach-

barn, um die enthaltenen Nachrichten zu einem neuen Paket zu kombinieren. Dieses Kombinieren und Zusammenfassen von Paketen reduziert die Anzahl der benötigten Pakete im weiteren Verlauf einer Ende-zu-Ende-Übertragung im Multi-Hop-Netzwerk.

Ohne zuvor existierende codierte Übertragungen in der eigenen Nachbarschaft führt dies jedoch zu zwei grundlegenden Problemstellungen zu Beginn der Verteilung codierter Nachrichten.

Zum Einen erhöht sich durch die Wartezeit auf andere Nachbarn vor der Codierung auch die Latenz bereits empfangener Nachrichten bis zur Weiterleitung. Die Latenz zwischen Quelle und Senke der Übertragung wird hier erhöht, da jeder weiterleitende Knoten, der ein Reassembling durchführt, eine Wartezeit von bis zu t_{enc} einhält, so lange keine weiteren Übertragungen zeitgleich im Netzwerk stattfinden. Wird der Encoding Buffer $\mathcal{B}_{enc}$ zuvor mit weiteren Nachrichten aus anderen Übertragungen oder durch neue Daten aus Messungen des weiterleitenden Knotens gefüllt, so ist ein vorzeitiges Weiterleiten möglich. Nachrichten mit hoher Priorität nutzen daher den Fast-Forwarding-Prozess und umgehen das Reassembling und die induzierte Latenz durch Zwischenspeicherung.

Zum Zweiten kann das langsame Verbreiten der codierten Pakete ein zügiges Decodieren auf Knoten verhindern, die bei einer Multi-Hop-Weiterleitung nachfolgen. Dies ist insbesondere dann entscheidend für eine effiziente Verteilung, wenn die Pakete nur wenige, aber innovative Nachrichten enthalten.

Um zu entscheiden, ob Nachrichten per Fast-Forward an Nachbarn weitergeleitet werden sollen, bewertet OCSSIM den Vorteil durch die Weiterleitung entsprechend der Nachricht und Netzwerkeigenschaften. Dazu können verschiedene Heuristiken, Algorithmen und Netzwerkeigenschaften herangezogen werden. Beispielsweise, kann die Anzahl der Nachbarn und ihr Knotengrad verwendet werden, um die Topologie anzupassen und die Weiterleitungswahrscheinlichkeiten neu zu berechnen, wie in [58] vorgeschlagen.

ROSMARIN setzt hier nur eine einfache Heuristik für die Weiterleitungsentscheidung ein. Die Heuristik basiert auf der Anzahl der identischen Pakete, die zuvor aus der lokalen Nachbarschaft $R(P)$ empfangen wurden. Für jeden weiterleitenden Knoten wird einen zufälliger Zeitraum $t_b << t_{enc}$ gewählt, um die Weiterleitung zurückzustellen. In diesem Zeitraum wird der Empfang weiterer Pakete aus der Nachbarschaft abgewartet. Wenn die Anzahl der empfangenen identischen Pakete $R(P)$ über der vordefinierten Schwelle $R(P) > thr_{drop}$ liegt, wird das Paket verworfen, um den Datenverkehr im Netzwerk zu reduzieren und das Risiko von Stausituationen zu verringern.

Dabei ist zu erwarten, dass alternative Weiterleitungsrouten zwischen Senken- und Quellknoten in dichten drahtlosen Netzwerken eine hohe Anzahl identischer Nachrichten hervorrufen. Als Schwelle wurde daher $thr_{drop} = 3$ als Kompromiss zwischen effizienter Weiterleitung und Netzwerkabdeckung gewählt.

6.3.4.2 Reassembling

Der Zweck der *Reassembling*-Komponente ist es, die Anzahl der Decodierungsmöglichkeiten zu erhöhen, indem möglichst unterschiedliche Nachrichten in ein neu erstelltes Paket codiert werden. Dazu können zuvor decodierte Nachrichten mit nicht-decodierten Blöcken aus Nachrichten kombiniert werden. Würde keine zufallsgesteuerte Mischung der Nachrichten stattfinden, so würden potentiell mehrere Nachbarknoten bei der Weiterleitung identische Pakete erzeugen. Werden aber immer identische Pakete empfangen, so kann der Empfänger nicht die erforderliche Anzahl unterschiedlicher Pakete zur Decodierung sammeln (siehe Kapitel 3.2.1).

6.3.5 Bewertungsmodul

Die in OCSSIM verwendete Strategie zur opportunistischen Weiterleitung kann in ROSMARIN mit in einem Modul das die Schnittstelle `MsgValue` realisiert kontrolliert werden.

In dieser Arbeit wurde dazu die einfache heuristische Strategie aus Kapitel 5.5.1 verwendet, die die Weiterleitung anhand eines zufälligen Backoffs t_b vornimmt.

6.3.6 Opportunistische Transceiver-Komponente

Die *Opportunistische Transceiver-Komponente* ist die Komponente, die in ROSMARIN den Austausch der zu sendenden Pakete und eintreffenden Pakete mit der Medienzugriffsschicht vornimmt.

Hier wurde eine einheitliche Komponente zur Koordination aller eintreffenden und weiterzuleitenden Pakete geschaffen, um redundante Pakete unabhängig von der bevorzugten Art der Weiterleitung reduzieren zu können und Sendezeitpunkte, sowie Sendewiederholungen einheitlich handhaben zu können.

Auf einem Knoten eintreffende Pakete werden durch die Opportunistische Transceiver-Komponente hinsichtlich Priorisierung, Codierungskomplexität und Zustandsinformationen aus der Nachbarschaft analysiert. Auf Grundlage dieser Analyse wird den Komponenten zum Fast-Forwarding und zur Netzwerkcodierung signalisiert, dass neue Pakete mit codierten Informationen eingetroffen sind und im Folgenden weitere Verarbeitungsschritte durchgeführt werden können. Die aus dem Paketkopf, den MessageIDs und durch die Medienzugriffsschicht bereitgestellten Zustandsinformationen, wie beispielsweise die ID des sendenden Knotens, werden extrahiert. Sie werden sowohl der Komponente zur Zustandshaltung des Knoten- und Nachbarschaftszustands bereitgestellt, als auch für die Signalisierung von impliziten Empfangsbestätigungen verwendet.

Die Komponente ist darüber hinaus ein Kernelement für die opportunistische Weiterleitungsentscheidung, da hier die Entscheidung getroffen wird, ob Pakete unterdrückt und verworfen werden, wenn diese häufig in der Nachbarschaft kommuniziert werden.

Zur Koordination dient insbesondere die *Send-Listen-Queue* (SLQ). Diese kombiniert eine Warteschlange für ausgehende Pakete mit einem Puffer für bekannte

Pakete und die Zustandshaltung für die opportunistische Weiterleitung. Ihr Zweck und Aufbau wird in Abschnitt 6.4 im Detail erläutert.

6.3.7 Zustandshaltung

Um eine interne Zustandshaltung für komplexere Weiterleitungsentscheidungen zu ermöglichen, bietet ROSMARIN eine Komponente zur Daten- und Zustandshaltung des Knoten- und Nachbarschaftsstatus. Diese *Knoten- und Nachbarschaftszustand*-Komponente kann gespeicherte Informationen ebenfalls direkt für die Nutzung in Anwendungen bereitstellen. Die gespeicherten Informationen können als zusätzliche Entscheidungsgrundlage genutzt werden, um zu bewerten, ob eine Nachricht priorisiert weiterverteilt werden soll. Soll eine Nachricht gezielt in der Nachbarschaft zugestellt werden, ohne dass Adressinformationen vorliegen, kann hier auch die nötige Information zum Identifizieren des gewünschten Nachbarn abgelegt werden. OCSSIM nutzt diese Komponente lediglich für Zustandsinformationen, die aus dem Mithören eines Paketes der direkten Nachbarschaft ermittelt werden können. So lassen sich Aussagen über die Anzahl der Nachbarn treffen. Explizite Statusnachrichten zum Austausch von Statusinformationen über mehrere Hops hinweg sind in OCSSIM nicht vorgesehen, lassen sich aber in ROSMARIN explizit aktivieren und mittels der *Knoten- und Nachbarschaftszustand*-Komponente für den gewünschten Verwendungszweck anpassen. Beispielsweise lässt sich somit eine Auswahl von Zielknoten bei der Weiterleitung oder eine dedizierte Signalisierung von kritischen Zuständen, wie bei Fehlern an der Sensorik, an Nachbarknoten umsetzen.

6.3.8 Filterung redundanter Informationen

Als Alternative zum Zwischenspeichern von bereits weitergeleiteten Paketen und Nachrichten zur Erkennung von später eintreffenden Duplikaten wurde die Nutzung eines *Bloomfilters* untersucht, um die Größe der Send-Listen-Queue und der Zwischenspeicher in ROSMARIN zu reduzieren.

Bloomfilter bezeichnen eine Datenstruktur, die genutzt werden kann, um zu überprüfen, ob ein Element Bestandteil einer Menge aus Elementen ist. Zur Überprüfung und zum Eintragen eines Elements x wird eine Menge aus Hashfunktionen $h_1, ..., h_k$ genutzt und pro Hashfunktion jeweils ein Hashwert $h_i(x)$ über das überprüfte Element gebildet. Die einzelnen Hashwerte geben jeweils eine Position in der Datenstruktur an. Besitzt der Bloomfilter an allen ermittelten Positionen $h_1(x), ..., h_k(x)$ einen gültigen Eintrag, so ist das Element ein Bestandteil der Menge der Elemente des Bloomfilters. Beim Eintragen werden dementsprechend Positionen ohne gültigen Eintrag als gültige Einträge gekennzeichnet. Abbildung 6.4 zeigt beispielhaft das Überprüfen eines Elementes x im Bloomfilter, bei dem die ermittelten Positionen mit Ausnahme des Hashes für $h_i(x)$ gültige Einträge besitzen. Das Element x befindet sich somit nicht im Bloomfilter.

In [16] wird ein datenzentriertes Routing-Protokoll BDP entworfen, das als Kernelement Bloomfilter nutzt, um die Datenverteilung zu kontrollieren. Jeder Knoten

Abbildung 6.4 Überprüfen eines Elements im Bloomfilter

sammelt Informationen zum Zustand der verwendeten Software und stellt diese als Versionsinformationen bereit. Anhand der Versionsinformationen kann durch den Bloomfilter überprüft werden, welche Aktualisierungsinformationen durch den Knoten benötigt werden und welche Informationen beim Empfang direkt verworfen werden können. Dabei kann eine schnellere Verteilung der Nachrichten mit weniger Nachrichten als in DIP [72] erreicht werden. Zugleich ist bei der Verwendung von Bloomfiltern aber das Auftreten von Hashkollisionen möglich, die sogenannte *False Positives* verursachen. Bei False Positives wird ein Eintrag fälschlicher Weise als Duplikat erkannt, so dass benötigte Aktualisierungsinformationen nicht durch den Knoten übernommen werden.

Als weitere Folge von False Positives können Bloomfilter, die bei der Weiterleitungsentscheidung eingesetzt werden, zusätzliche Datenverluste durch das fälschliche Verwerfen von Nachrichten oder Paketen hervorrufen. Zur Reduzierung der Wahrscheinlichkeit von False Positives ist eine Vergrößerung des Bloomfilters notwendig. Entsprechender Zusatzaufwand ist für die benötigte Speichermenge einzuplanen.

Wird für eine vorgegebene Speichermenge von m Bits zur Unterscheidung von n möglichen Elementen eine Anzahl $k = \frac{m}{n} ln2$ Hashfunktionen verwendet (vgl. [34]), so ergibt sich für f als Wahrscheinlichkeit für False Positives:

$$f = (1 - (1 - \frac{1}{m})^{kn})^k$$

Die benötigte Speichermenge eines Bloomfilters steigt jedoch nicht mit der Anzahl der gespeicherten Elemente und kommt daher in Netzwerkszenarien mit hohem Datenaufkommen bevorzugt zum Einsatz, um speicheraufwändiges Caching zu vermeiden.

Im Rahmen einer Bachelorarbeit [86] wurden daher Untersuchungen zur Nutzung eines Bloomfilters zur Weiterleitung in Sensornetzen mit Mehrwegeausbreitung angestellt. In den untersuchten Szenarien konnte bei einer Paketfehlerrate von bis zu 30% auf Einzelverbindungen ermittelt werden, dass die Nutzung von Bloomfiltern nur bei der Weiterleitung von Paketen gewinnbringend eingesetzt werden kann. Dies ist dann möglich, wenn das eingesetzte Verfahren zur Weiterleitung der steigenden Paketfehlerrate durch erhöhtes Senden von redundanten Paketen über unterschiedliche Pfade entgegenwirkt. Der zu erwartende Datenverlust durch False Positives steigt hierbei jedoch äquivalent mit der Paketfehlerrate an, so

dass schließlich von der Verwendung zusätzlicher Bloomfilter für eine optimierte Weiterleitung in OCSSIM abgesehen wurde.

6.3.9 MAC-Anbindung

ROSMARIN nutzt das durch TinyOS vorgegebene CSMA-Verfahren zum Medienzugriff. In Avrora können bei diesem Verfahren Kollisionen zwischen Paketen von unterschiedlichen Knoten auftreten, die trotz vorhergehender Überprüfung des simulierten Mediums (Carrier Sense) nicht erkannt und vermieden werden. Der Medienzugriff in TinyOS erfolgt in der Simulation ohne die Verwendung von Duty-Cycling oder weiterer Techniken zur Optimierung des Energiebedarfs direkt über die Schnittstelle des MAC-Protokolls.

Zur Nutzung alternativer Medienzugriffsverfahren, implementiert ROSMARIN die Schnittstelle der *Unified Radio Power Management Architecture* (UPMA) und ihrer Erweiterung *MAC Layer Architecture* (MLA) [54].

6.4 Struktur und Verwaltung des Cache- und Nachrichtenspeichers

In ROSMARIN werden unterschiedliche Arten von Zwischenspeichern benötigt. Die einzelnen Puffergrößen können an die verfügbare Speichermenge des jeweiligen Sensorknotens angepasst werden. Mit Ausnahme des Encoding Buffers $\mathcal{B}_{enc}$, beeinflusst die Größe des Pufferspeichers die Effizienz des opportunistischen Kommunikations- und des Codierungsschemas bei vorgegebener Netzwerkgröße.

Zusammenspiel der Zwischenspeicher

Abbildung 6.5 zeigt einen Überblick über die verwendeten Zwischenspeicher und die Schritte, die zur Relokation der Einträge führen. Rote Pfeile stellen Schritte dar, die zur Entscheidung über die Weiterleitung auf dem Reassembling- oder Fast-Forward-Pfade gehören. In Blau sind die Schritte der Netzwerkcodierung bzw. der Resolution gekennzeichnet. Jeder Block in der Abbildung symbolisiert ein Paket oder Element des jeweiligen Zwischenspeichers. Die abgebildeten Zahlenwerte sind exemplarisch für die $MsgID_i$ einer dort enthaltenen Nachricht. Pfeile an den oberen Rand der Abbildung symbolisieren eine Signalisierung an Anwendungen, die das Kommunikationsschema nutzen. Pfeile an den unteren Rand der Abbildung symbolisieren dagegen den Empfang neu eintreffender Pakete auf dem Knoten bzw. das Versenden und Weiterleiten von Paketen auf dem Knoten. Im Folgenden werden die Realisierungen der hier abgebildeten Zwischenspeicher anhand ihrer Rolle in diesem Zusammenspiel und ihrer Realisierung in ROSMARIN erläutert.

Zwischenspeicherung bei Fast-Forwarding

Fast-Forwarding nutzt lediglich den Decoding Buffer $\mathcal{B}_{dec}$ zur Zwischenspeicherung. Die empfangenen Pakete für Fast-Forwarding werden zugleich verwendet, um das Decodieren zu erleichtern, indem zusätzliche Decodierungsmöglichkeiten geschaffen werden.

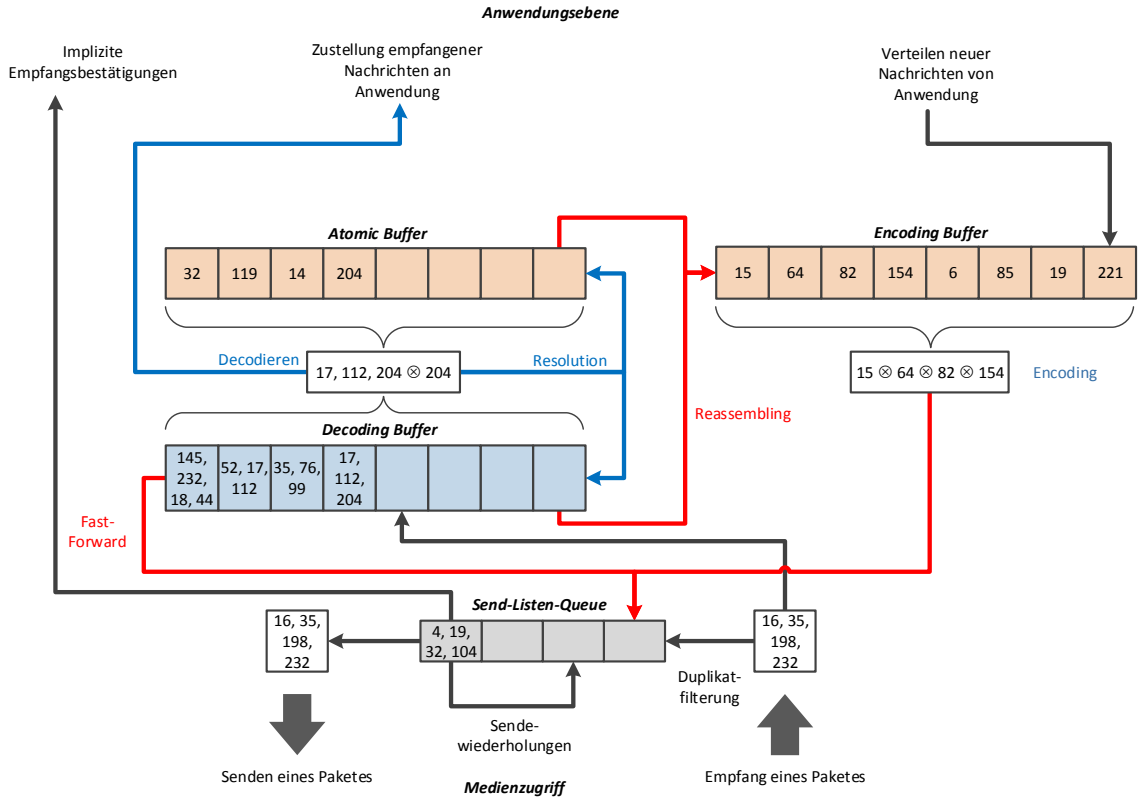

Abbildung 6.5 Überblick über die verwendeten Zwischenspeicher in ROSMARIN

Zwischenspeicherung bei der Netzwerkcodierung und Reassembling

Zum Reassembling werden decodierte Nachrichten aus dem Atomic und Decoding Buffer in den Encoding Buffer verschoben. Erreicht der Encoding Buffer dabei einen ausreichenden Füllstand thr_{enc} oder wird das Zeitlimit t_{enc} überschritten, so werden die enthaltenen Nachrichten des Zwischenspeichers kombiniert und in Paketen unterschiedlicher Codierungskomplexität versendet (siehe Abschnitt 6.3.4.2).

Verteilen neuer Anwendungsnachrichten

Sollen durch Vorgabe einer Anwendung neue Nachrichten mittels ROSMARIN verteilt werden, so werden diese Nachrichten direkt dem Encoding Buffer hinzugefügt. Somit können diese mit weiteren Nachrichten zum Reassembling kombiniert werden. Haben die Anwendungsnachrichten eine hohe Priorität, so wird ein unmittelbares Encoding ohne vorheriges Abwarten von thr_{enc} oder t_{enc} durchgeführt.

Decodieren und Resolution

Der Atomic Buffer $\mathcal{B}_{atom}$ enthält die atomaren Nachrichten, die durch vorhergehende Resolutionsschritte ermittelt werden konnten oder aufgrund hoher Nachrichtenpriorität direkt decodiert werden können. Treffen neue Pakete auf einem Knoten ein, die decodiert werden müssen, so werden bevorzugt die Einträge im Atomic

Buffer für die Resolution genutzt, um die Codierungskomplexität des Paketes zu reduzieren. Decodierte Nachrichten können im Anschluss einer Anwendung unmittelbar zugestellt werden.

Send-Listen-Queue

Des Weiteren wird durch ROSMARIN eine gemeinsame Warteschlange (*Send-Listen-Queue* (SLQ)) für sendebereite Pakete und Pakete die bereits gesendet, aber noch nicht durch Mithören implizit bestätigt wurden, bereitgestellt.
Die Elemente der SLQ enthalten jeweils die folgenden Informationen:

- Einen Eintrag zur Kennzeichnung von Send- und Listen-Elementen. Sobald ein Paket gesendet wurde, wechselt der Eintrag von Send zu Listen.
- Das Paket, welches zu senden ist oder das noch nicht implizit bestätigt wurde.
- Die Anzahl der durchgeführten Sendewiederholungen, um das Senden beim Erreichen des Schwellenwertes für Sendewiederholungen abzubrechen.
- Die Anzahl identischer mitgehörter Pakete von Nachbarn. Beim Erreichen des Schwellenwertes thr_{drop} wird auch hier der Sendevorgang abgebrochen.
- Einen Backoff-Wert, der bei Send-Einträgen zugleich dem Zurückstellen beim Mithören identischer Pakete dient, als auch als Zeitfenster für die Bestätigung des Paketes bei Listen-Einträgen.
- Eine Kennzeichnung bestätigter Elemente. Dies erlaubt es, Elemente in der SLQ verweilen zu lassen, die nicht länger benötigt werden, aber zur Filterung bekannter Pakete herangezogen werden können. Bestätigte Elemente werden daher bei hohem Füllstand der Warteschlange bevorzugt entfernt.

Die SLQ dient nebenbei zur Reduktion des vom Knoten zu behandelnden Datenverkehr, indem empfangene Pakete, die bereits in der Warteschlange vorhanden sind, verworfen werden. Zugleich kann ein Listen-Eintrag in der Schlange implizit als Paket bestätigt werden oder ein Paket mit Send-Eintrag verzögert werden, um auf weitere identische Pakete von Nachbarn zu warten, bzw. den Sendevorgang bei einer ausreichenden Anzahl von Duplikaten abzubrechen.

6.5 Evaluation

Im folgenden Abschnitt werden die Werkzeuge, Anwendungsszenarien und Randbedingungen erläutert, die zur Evaluation des OCCSIM-Kommunikationsschemas mittels des ROSMARIN-Rahmenwerk verwendet wurden. Anschließend werden die ermittelten Evaluationsergebnisse vorgestellt und erläutert.

6.5.1 Integrierte Evaluationsmechanismen des Rahmenwerks und Evaluationsmetriken

Das ROSMARIN-Rahmenwerk enthält zusätzlich zu den Komponenten zur Umsetzung des OCSSIM-Kommunikationsschemas (vgl. Tabelle 6.1und Abbildung 6.2) die weitere Komponente `EvalModule`.

Mittels dieser Komponente können die Zustände von Puffern und Nutzungsinformationen von Funktionen der Komponenten in ROSMARIN während der Laufzeit auf den einzelnen Knoten erfasst werden. Außerdem lässt sich die Komponente je nach Bedarf zur Erfassung weiterer Zustände erweitern und für die Nutzung von ROSMARIN auf realen Sensorknoten außerhalb des Simulators Avrora vollständig deaktivieren.

Durch Ausgabe der Statusinformationen in numerischer Form und anschließender Verarbeitung der Statusinformationen mittels zusätzlicher Skripte im Anschluss an die Simulationslaufzeit konnte der Zusatzaufwand für die Evaluationsmechanismen auf den Sensorknoten stark beschränkt werden. Dazu werden die auftretenden Zustände der Komponenten durch vorgegebene numerische Konstanten repräsentiert. Anders als bei der Repräsentation durch Alphanumerische Zeichen wird hierzu, aufgrund von Optimierungen zur Compile-Zeit, kein zusätzlicher Speicher im RAM des Knotens zur Repräsentation der Zustände benötigt.

Die zusätzlichen Evaluationsausgaben wurden sehr speichereffizient realisiert, so dass lediglich 598 Bytes im ROM und 150 Bytes im RAM des Knotens an zusätzlichem Speicher benötigt werden.

Die integrierten Evaluationswerkzeuge von ROSMARIN umfassen:

- das Protokollieren eines aufgetretenen Knotenzustands oder der Nutzung einer durch ROSMARIN bereitgestellten Funktionalität
- das Erfassen der maximalen und minimalen Füllstände eines Puffers in ROSMARIN

Ergänzend dazu stehen die folgenden Evaluationshilfsmittel zur Verfügung, die nicht in den Code der Knoten integriert wurden, aber vor oder nach Ablauf der Simulation genutzt werden können:

- Hilfsskripte zum Parsen der Zustände der Knoten in ein lesbares Format und zum Summieren der Zustandsinformationen
- Hilfsskripte zum Erzeugen von Gitter- und Zufallstopologien
- Skripte zur Erfassung der erfolgreich verteilten Anfragenachrichten einer Quelle auf den Senkenknoten
- Skripte zur anschließenden Erfassung der erfolgreich rekonstruierten Antwortnachrichten von Senkenknoten auf einem Quellknoten

Das `EvalModule` in ROSMARIN erfasst etwa 70 unterschiedliche Zustandsinformationen insbesondere aus den folgenden Bereichen als Evaluationsmetriken:

- Anzahl der verzögerten Pakete bei Erkennung von Duplikaten in der Nachbarschaft
- Anzahl Sendewiederholungen und der abgebrochenen Sendevorgänge nach Erreichen der Schwelle für Sendewiederholungen
- Anzahl der verworfenen Nachrichten und Pakete aufgrund von vollen Puffern
- Anzahl der erkannten Duplikate für eingehende und ausgehende Pakete und Nachrichten
- Anzahl der per Resolution decodierten Nachrichten und Resolutionsversuche, die nicht zur Decodierung beitragen konnten
- Anzahl der codierten und decodierten Nachrichten
- Anzahl der mehrfach dekodierten Nachrichten
- Anzahl der zugestellten Nachrichten
- Anzahl der decodierten Nachrichten und Reassembling-Vorgänge
- Anzahl der Fast-Forwarding-Vorgänge und dabei verworfene Nachrichten
- Anzahl der erkannten impliziten Bestätigungen für Pakete und Nachrichten.
- Nachrichtentyp einer verteilten Nachricht (Anfrage, Rückantwort oder weitere)
- Minimale und maximale Pufferfüllstände der Send-Listen-Queue und des Atomic, Decoding und Encoding Buffer
- Codierungskomplexität und Paketgröße nach enthaltenen Nachrichten

Daneben dient die Protokollierung von Avrora zur Erfassung von Zeitintervallen und der Anzahl gesendeter und empfangener Pakete und Bytes pro Knoten, sowie der Anzahl der Kollisionen auf dem simulierten Medium.

6.5.2 Gewählte Parametrisierung

Zur Evaluation wurden jeweils die in Tabelle 6.2 angegebenen Parameter genutzt, wenn in den folgenden Abschnitten nicht andere Angaben gemacht werden.

Zur Analyse wurden zwei unterschiedliche Arten von Topologien verwendet. Die statischen Gitter-Topologien (siehe Abbildung 6.6) haben unterschiedliche Größen zwischen 2x2 und 7x7 Knoten. Die Knoten sind jeweils in einem konstanten

Tabelle 6.2 Simulationsparameter

Parameter	Wert
Simulationsdurchläufe pro Parameterset: r	100
Einträge der Send-Listen-Queue: $\lvert\mathcal{B}_{slq}\rvert$	20 codierte Pakete und Kennzeichnungen (880 Bytes RAM, 178 Bytes ROM)
Einträge des Decoding Buffers: $\lvert\mathcal{B}_{dec}\rvert$	20 codierte Pakete (680 Bytes RAM, 46 Bytes ROM)
Einträge des Atomic Buffers: $\lvert\mathcal{B}_{atom}\rvert$	30 decodierte Nachrichten (780 Bytes RAM, 22 Bytes ROM)
Gitter-Topologie	2x2 bis zu7x7 Knoten
Distanz zu Nachbarn	19 m
Zufällig ausgebrachte Topologien	5, 10, 15, 20, 25, 30 Knoten
Ausbringungsfläche	60 x 60 m^2

Abstand von 19 m zu ihren Nachbarn ausgebracht. Jeder Knoten hat maximal 4 Nachbarn. Die Senke wurde an eine statische Position in der ersten Reihe des Gitters ausgebracht, um den Einfluss der Position der Senke auf den Decodierungserfolg möglichst gering zu halten.

In der zweiten Topologie (siehe Abbildung 6.7) sind die Sensorknoten zufällig auf einer vorgegebenen Fläche von 60 x 60 m^2 ausgebracht. Dabei bleibt die Gesamtfläche statisch, die Knotenpositionen werden jedoch in jedem Simulationsdurchlauf geändert.

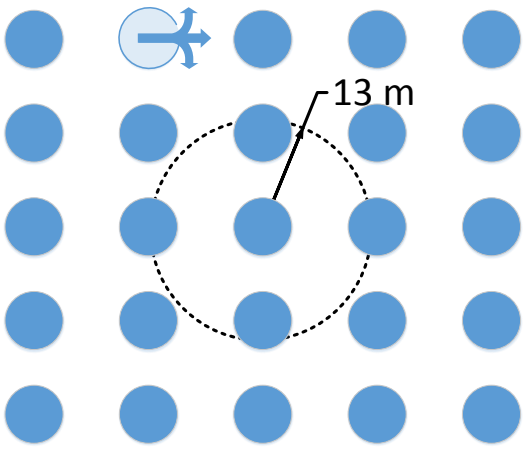

Abbildung 6.6 Topologie zur Evaluation eines Kommunikationsszenarios mit einer einzelnen Senke in einer Gittertopologie

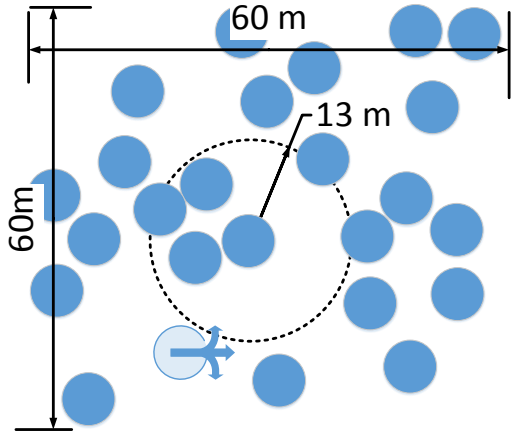

Abbildung 6.7 Topologie zur Evaluation eines Kommunikationsszenarios mit zufälliger Verteilung der Knoten

6.5.3 Evaluation von Einzelsenken

Im Experiment wurde hier eine einzelne Senke verwendet, die eine Anfragenachricht in das Netzwerk einbringt. Die Anfragenachricht wird daraufhin im gesamten Netzwerk an alle erreichbaren Knoten verteilt. Jeder Knoten der eine Anfragenachricht erfolgreich aus den empfangenen Paketen decodieren kann, verhält sich als Nachrichtenquelle und antwortet mit einer Rückantwort, die den Knotenstatus enthält. Die Statusinformation beinhaltet in diesem Experiment lediglich die ID eines aktiven Knotens und signalisiert die Einsatzbereitschaft des Knotens. Jeder weiterleitende Knoten codiert die erhaltenen Nachrichten entsprechend des OCSSIM-Codierungsschemas und kombiniert gegebenenfalls auch Rückantworten mit Anfragenachrichten in ein gemeinsames Paket zur Weiterleitung.

6.5.3.1 Anteil dekodierbarer und rekonstruierbarer Nachrichten

Um das Verhalten des Weiterleitungsprozesses zu ermitteln, spielt die Rekonstruktionsrate der Nachrichten, die codiert in Paketen versendet wurden, die entscheidende Rolle.

Für eine exakte Ermittlung der Verteilungs- und Rekonstruktionsrate der gesendeten Rückantworten speichern und decodieren in diesem Experiment alle Knoten die mitgehörten Nachrichten ihrer Nachbarn.

Da OCCSIM aktiv die Mehrwegeausbreitung zur Weiterleitung redundanter Pakete verwendet, können geringe Paketverlustraten und Ausfälle von einzelnen Knoten kompensiert werden.

Zur Auswertung der erhaltenen Ende-zu-Ende zugestellten Informationen ist daher die genaue Anzahl der zugestellten Pakete oder Nachrichten bedeutungslos. Vielmehr ist hier das Verhältnis zwischen gesendeten Nachrichten und der Anzahl decodierter disjunkter Nachrichten entscheidend. Dieses Verhältnis kann als Metrik für die verteilte Information im Netzwerk herangezogen werden.

6.5.3.2 Einfluss der Topologie

Die zugrunde liegende Netzwerktopologie hat einen entscheidenden Einfluss auf die Verteilungs- und Rekonstruktionsrate im Netzwerk.

Sind in der Netzwerktopologie viele alternative Pfade zwischen Quelle und Senke vorhanden, so erhöht dies die Anzahl der Pakete, die eine Senke beobachten kann.

Eine hohe Anzahl von Nachbarknoten erhöht die Wahrscheinlichkeit eines Knotens, eine Nachricht aus einem Paket decodieren zu können, da eine größere Paketvielfalt in der eigenen Nachbarschaft beobachtet werden kann. Zudem können eigene Nachrichten in Paketen schneller im Netzwerk verteilt werden, als bei Knoten mit einer geringen Anzahl von Nachbarn. Das führt insbesondere zu dem Effekt, dass Nachrichten aus Randbereichen des Netzes oder aus Bereichen mit geringer Konnektivität seltener rekonstruiert werden können.

6.5.3.3 Netzwerkgröße und Gitter-Topologie

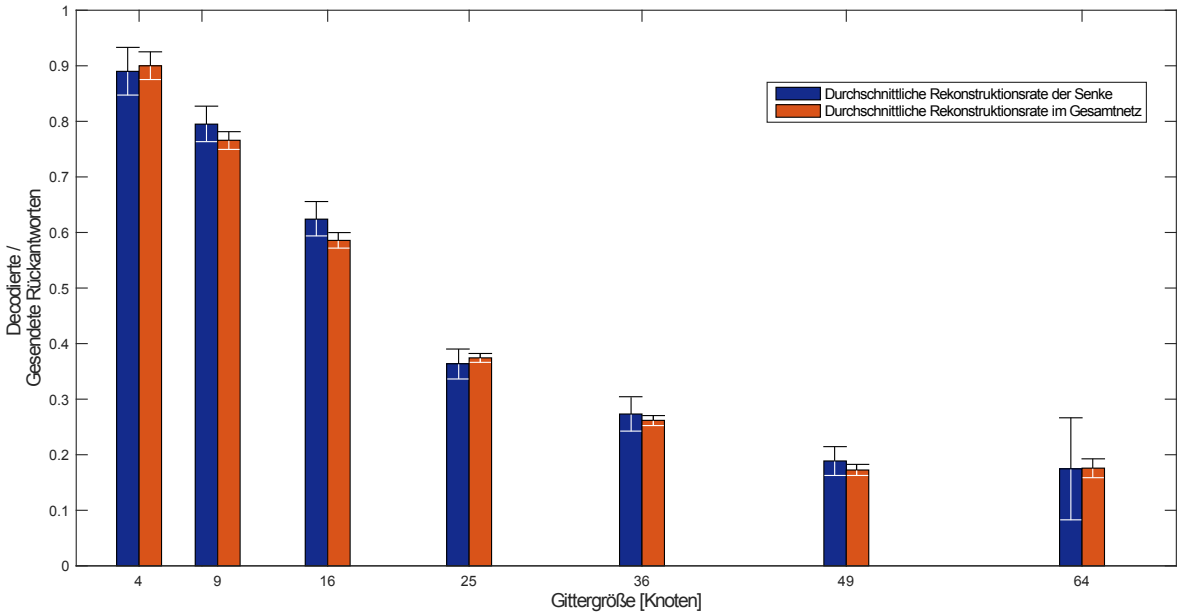

Abbildung 6.8 Anteil decodierter Antwortnachrichten bei Gittertopologien mit unterschiedlichen Netzwerkgrößen

Abbildung 6.8 zeigt die Rekonstruktionsrate der Rückantwortnachrichten, die von der Senke bzw. jeweils einem beliebigen Knoten des Netzwerks erreicht wird. Die Höhe der Balken zeigt den ermittelten Durchschnitt der Rekonstruktionsrate. Die Whisker geben jeweils die Konfidenzbereiche für ein Konfidenzniveau von 99,5% an.

Wächst die Größe des Netzwerks, so nimmt der Anteil von rekonstruierten Rückantworten zu gesendeten Rückantworten ab. Dies ist mit der steigenden Anzahl von Nachrichtenkombinationen zu begründen, die codiert in den Paketen im Netzwerk weitergeleitet werden. Die wachsende Anzahl der Nachrichtenkombinationen verursacht die Reduktion der Decodierungsmöglichkeiten auf den

Zwischenknoten. Die absolute Anzahl decodierter Rückantwortnachrichten pro Knoten schwankt dagegen geringfügig um einen konstanten Wert von ca. 11 Nachrichten, unabhängig von der Netzwerkgröße.

Dies ist ein Hinweis darauf, dass das Verfahren zur opportunistischen Weiterleitung und die Netzwerkcodierung im ganzen Netzwerk wirksam sind. Die Nachrichtenverteilung sorgt dafür, dass prinzipiell alle Knoten über ihre Nachbarn im Netzwerk erreicht werden können, ohne dass der Mechanismus zur Unterdrückung von bekannten Paketen (siehe Kapitel 5.5.1) dazu führen würde, dass einzelne Nachbarschaften von der Verteilung abgeschnitten werden.

Zudem konnte beobachtet werden, dass alle Knoten im Experiment auch eine Rückantwort erstellen und codiert an ihre Nachbarn versenden. Daher muss davon ausgegangen werden, dass der Medienzugriff hier ebenfalls starken Einfluss auf die ermittelte Anzahl der decodierten Rückantworten hat. Bei steigender Netzwerkgröße reicht der in Kapitel 5.5.1 eingeführte Backoff-Mechanismus nicht aus, um die kollisionsfreie Übertragung der codierten Pakete zu gewährleisten. Insbesondere tritt das Problem auf, wenn viele Knoten im gleichen Zeitraum eigene Nachrichten erzeugen und an ihre Nachbarn in Reichweite senden oder Pakete weiterleiten, wie dies im untersuchten Anwendungsszenario bei den Rückantworten der Fall ist. Das verstärkt den Effekt der insgesamt fallenden Rekonstruktionsrate mit steigender Netzwerkgröße.

Dennoch zeigt die konstante Anzahl decodierter Rückantworten, dass die erhaltene Information eines Knotens auch mit steigender Netzwerkgröße nicht einbricht und ein robustes Verhalten des Kommunikationsschema zu erwarten ist.

6.5.3.4 Netzwerkdichte und zufällig verteilte Knoten

In [118] wird die benötigte Anzahl von Nachbarn ermittelt, die ein Knoten benötigt, um in einem zufällig ausgebrachten Sensornetzwerk die Konnektivität für eine Multi-Hop-Topologie aufrecht zu erhalten. Die Anzahl benötigter Nachbarn steigt mit der Netzwerkgröße n mit $\Theta(c \log n)$. Als ein brauchbarer Wert hat sich $c \geq 1,5$ erwiesen, um mit einer stabilen Anzahl von Nachbarn die Konnektivität mit hoher Wahrscheinlichkeit aufrecht zu erhalten.

Die Auswirkungen der durchschnittlichen Anzahl von Nachbarn pro Knoten auf das Rekonstruktionsverhalten wird in Abbildung 6.9 deutlich.

Die Abbildung zeigt den Anteil der Knoten im Netzwerk, die erfolgreich eine Rückantwort decodieren konnten. Die Darstellung zeigt unterschiedliche Netzwerkdichten in zufällig ausgebrachten Topologien. Auch hier zeigt die Höhe der Balken den jeweiligen Durchschnitt an, während die Whisker die Konfidenzbereiche für ein Konfidenzniveau von 99,5% kennzeichnen.

Mit zunehmender Netzwerkdichte nimmt die Rekonstruktionsrate leicht ab. Die Abbildung zeigt zusätzlich die durchschnittliche Antwortrate der Knoten im Experiment. Die Antwortrate fungiert als obere Grenze der erreichbaren Rekonstruktionsrate und ist abhängig von der Konnektivität der Knoten. Sinkt die Knotendichte, so sinkt auch die Konnektivität, da die Wahrscheinlichkeit, dass sich andere Kno-

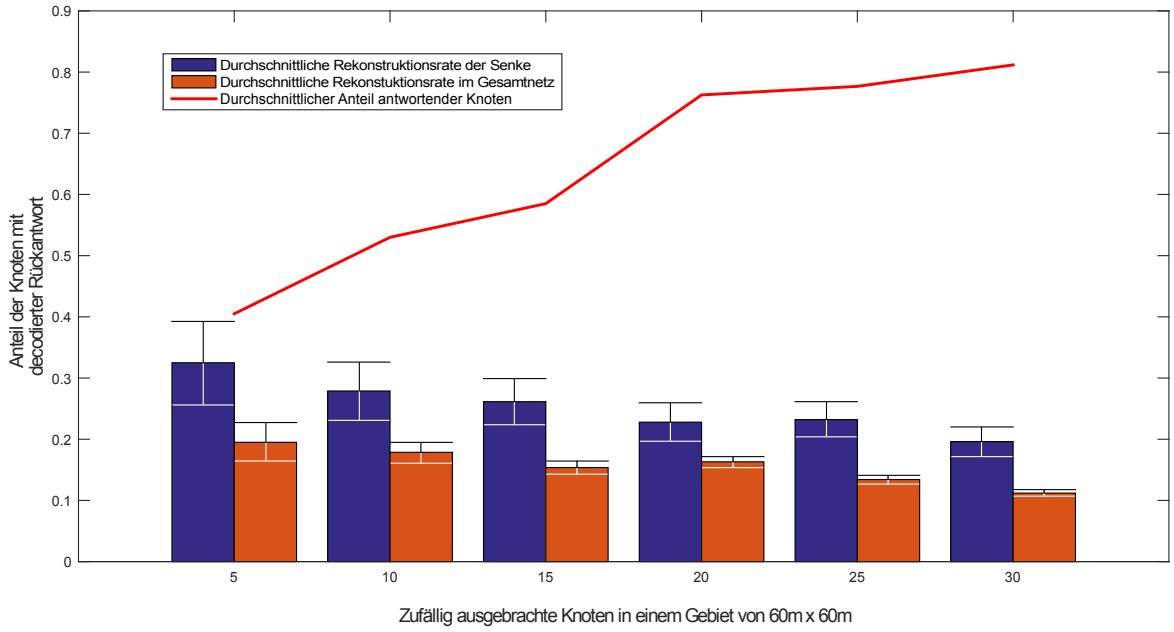

Abbildung 6.9 Anteil decodierter Antwortnachrichten bei zufälliger Topologie mit unterschiedlichen Knotendichten

ten in Sendereichweite befinden abnimmt. In Folge sinkt also die Antwortrate der Knoten, die bereits die Anfragenachricht nicht erhalten.

Dies erlaubt auch eine Schlussfolgerung bezüglich des Kommunikationsverhaltens im Falle von Knotenausfällen. Der Ausfall eines Knotens ist hier äquivalent zum Verhalten eines Netzwerks mit geringerer Dichte.

So lange ein Pfad zwischen Quelle und Senke existiert, adaptiert sich OCSSIM an die vorherrschende Netzzwerktopologie. Unabhängig davon, ob Pakete sequentiell über eine einzelne Verbindung bei einem Knoten eintreffen, oder ob die Verbreitung über mehrere unterschiedliche Verbindungen erfolgt, können die Decodierungsmöglichkeiten durch OCSSIM ausgenutzt werden.

Bei Nutzung des TinyOS CSMA-Medienzugriffsverfahren treten in der Simulation Kollisionen zwischen Paketen benachbarter Knoten auf, die auch die Rekonstruktionsrate der Nachrichten beeinflussen. Dies führt zur Verringerung der Decoding-Möglichkeiten und betrifft insbesondere Netzwerke mit hohen Knotenzahlen oder großer Knotendichte. Die Datenmenge, die von Kollisionen betroffen ist entspricht hier 1% der gesendeten Bytes in der Gittertopologie und bis zu 3% der gesendeten Bytes in dichten, zufällig ausgebrachten Topologien. Durch die Nutzung von anderen Medienzugriffsverfahren kann die Anzahl der Kollisionen weiter reduziert werden (vgl. 3.6.2).

Die Rekonstruktionsrate der Senke ist im Vergleich zu der Rekonstruktionsrate eines durchschnittlichen Knotens im Netzwerk leicht erhöht. Die Reihenfolge der Rückantworten der Knoten ist für diesen Effekt ausschlaggebend. Knoten, die sich in der Nähe der Senke befinden antworten tendenziell früher als Knoten in

größerer Entfernung. Daher sind die Rückantworten von Knoten nahe der Senke oft mit einer niedrigen Komplexität codiert und können leichter decodiert werden. Die Rückantworten von Knoten in größerer Entfernung zur Senke werden dagegen häufiger mit anderen Rückantworten zusammen in ein Paket codiert und sind potentiell schwieriger zu rekonstruieren.

6.5.3.5 Einfluss der Größe des Cache-Speichers

Die Größe des Atomic Buffer $\mathcal{B}_{atom}$ begrenzt die Decodierungsmöglichkeiten. Wird mehr Zwischenspeicher benötigt, als $\mathcal{B}_{atom}$ bereitstellt, dann werden die ältesten Einträge entfernt. Häufig sind diese aber noch für weitere Decodingvorgänge nutzbringend, so dass das Decodieren neu eintreffender Pakete erschwert oder verhindert werden kann.

Hat $\mathcal{B}_{atom}$ eine Größe von 30 Einträgen, so sind die ersten Einträge im Experiment aus dem Zwischenspeicher zu entfernen, wenn die Anzahl der Knoten über der Anzahl der Einträge im Atomic Buffer liegt. Dies trifft sowohl auf die Gittertopologie, als auch auf die zufällige Topologie zu.

Die Größe des Decoding Buffers zu überschreiten, konnte dagegen in diesem Experiment vermieden werden. Daher ist die Verwendung eines eigenständigen Atomic Buffers zweckmäßig, um den Speicherbedarf gegenüber einer Realisierung mit einem einzelnen Decoding Buffers einsparen zu können.

Um die Rekonstruktionsraten und die Effizienz der Weiterleitung zu optimieren, sollte aber die Größe des Atomic Buffers an die Netzwerkgröße angepasst werden.

6.5.3.6 Encoding mit unterschiedlicher Nachrichtenanzahl

Um den Einfluss der Beschränkung der Codierungskomplexität auf die Nachrichtenverteilung zu überprüfen, wurde die Anzahl der maximal in einem Paket enthaltenen Nachrichten variiert. Anders als in den vorhergehenden Evaluationen wurde die Netzwerktopologie nicht verändert. Das Evaluationsszenario entspricht hier den Vorgaben aus Abschnitt 6.5.2 mit einer Gittertopologie mit 5x5 Knoten. Außerdem wurde die maximale Codierungskomplexität n in ROSMARIN für die nun folgenden Evaluationen auf $n = max(c) = 4$ beschränkt, um ein Kompromiss zwischen der Anzahl benötigter Pakete und der benötigten Speicherkapazität, Nachrichtenlänge und Decodingwahrscheinlichkeit zu erreichen.

Dass diese Schranke begründet ist, zeigt das Ergebnis in Abbildung 6.10. In der Abbildung ist die Häufigkeit der gesendeten Pakete und ihre jeweils beobachtete Codierungskomplexität aufgetragen. Dabei wurde das Limit der erlaubten Codierungskomplexität n zwischen 2 und 8 variiert. Während die beobachtete Codierungskomplexität die tatsächliche Anzahl der Nachrichten in einem gesendeten Paket angibt, legt das Limit die maximale Anzahl codierter Nachrichten pro Paket fest. Auf diese Weise kann beobachtet werden, wie effizient ein Codierungsschema der entsprechenden Komplexität auf der vorliegenden Topologie genutzt werden kann.

Es kann hierbei festgestellt werden, dass ein Großteil der Pakete eine niedrige Codierungskomplexität aufweist, unabhängig vom gewählten Limit n. Wird das

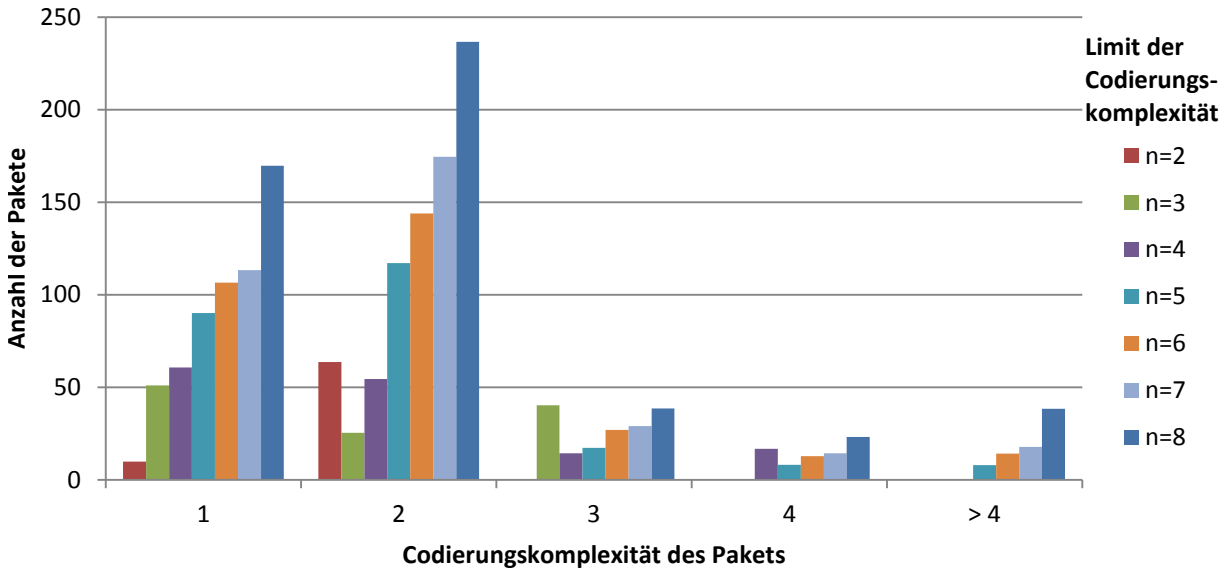

Abbildung 6.10 Sendehäufigkeit von Paketen einer Codierungskomplexität bei limitierter maximaler Codierungskomplexität n

Limit $n > 4$ gewählt, so ist dennoch zu beobachten, dass die größte Gruppe der Pakete nur eine Codierungskomplexität von 2 aufweist. Pakete mit einer Codierungskomplexität größer 4 treten unabhängig vom gewählten Limit in weniger als 5 Prozent der beobachteten Fälle auf.

Gründe für dieses Verhalten sind vor allem auf das gewählte Anwendungsszenario und die unterliegende Gittertopologie mit maximal 4 Nachbarn pro Knoten zurückzuführen. Im untersuchten Szenario antwortet jeder Knoten im Netzwerk auf die verteilte Anfrage mit einer eigenen Rückantwort. Das führt zu der dargestellten Situation, dass in einem Großteil der Pakete jeweils Anfragenachrichten und Rückantworten gemeinsam in ein Paket codiert wurden. Somit weisen viele Pakete, die sich nicht in der Nähe des anfragenden Knotens befinden, eine beobachtete Codierungskomplexität von $m = 2$ auf.

Wird die Codierungskomplexität hier auf ein Maximum $n > 4$ limitiert, so werden vermehrt Nachrichten in ein Paket codiert, die vom gleichen Nachbarn erhalten wurden und nun weitergeleitet werden sollen. Um die Vorteile der Nachrichtenverteilung über unterschiedliche Pfade der Gittertopologie bei der Decodierung zu nutzen, sollte die Codierungskomplexität auf einen Höchstwert von $c = 4$ begrenzt werden.

Zum Decodieren einer Nachricht aus einem Paket werden bis zu c unterschiedliche Pakete benötigt. Daher müssen bei höheren Grenzen gegebenenfalls mehrere unterschiedliche Pakete von einem Nachbarknoten empfangen werden, um ein Decodieren zu ermöglichen.

Abbildung 6.11 zeigt den Anteil decodierter Antwortnachrichten bei unterschiedlicher maximaler Codierungskomplexität.

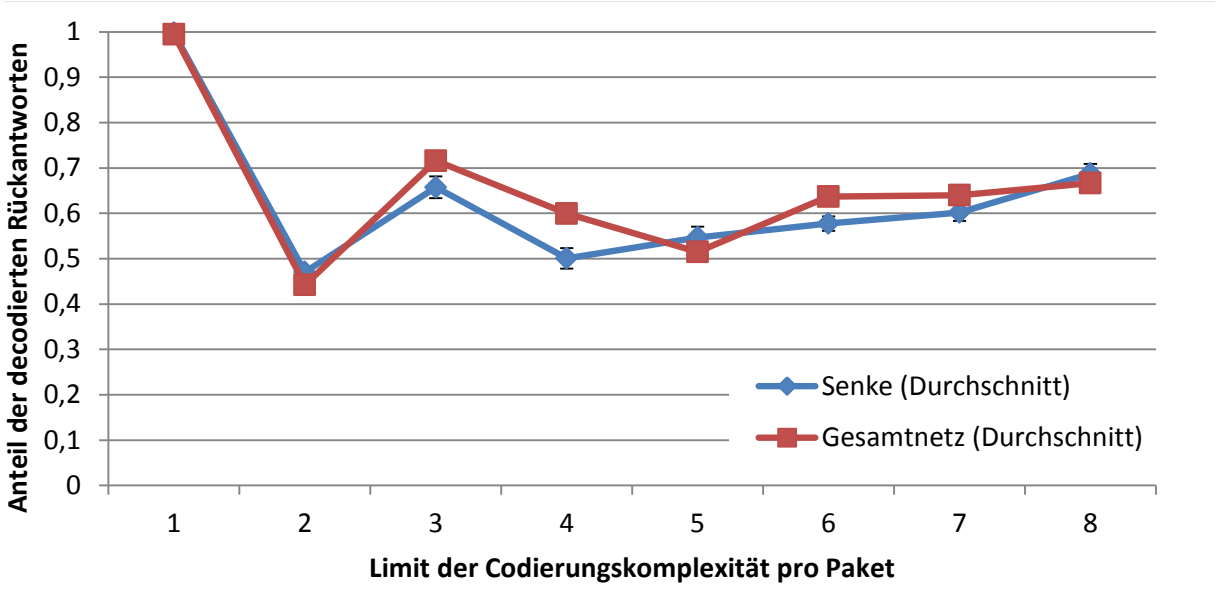

Abbildung 6.11 Anteil decodierter Antwortnachrichten bei unterschiedlicher maximaler Codierungskomplexität

Hier zeigt sich, dass das Limit der erlaubten Codierungskomplexität für Werte mit $n > 4$ keinen entscheidenden Einfluss auf die Rekonstruktion der Daten hat, da nur ein sehr geringer Anteil der Pakete das tatsächliche Limit ausnutzt.

Wird das Limit dagegen in den niedrigen Wertebereich mit $n = 2$ verschoben, so bricht der Anteil der decodierten Rückantworten ein. In diesem Bereich werden ist die Anzahl der Nachrichten, die zum Decodieren von Paketen genutzt werden können, zu gering. Ein codiertes Paket mit den beiden Nachrichten $M_1 \oplus M_2$ kann nur decodiert werden, wenn M_1 oder M_2 bereits erhalten wurden. Daher müssen im Fall von $n = 2$ deutlich mehr Nachrichten im Atomic Buffer an Stelle des Decoding Buffers abgelegt werden als für $n > 2$. Die im gesamten Netzwerk gesendete und empfangene Datenmenge wird durch diesen Umstand nicht beeinflusst (vgl. Abbildung fig:CodingLimit2). Allerdings hat dies zur Folge, dass die Pufferkapazitäten des Atomic Buffers schnell überschritten wird und folglich die zum Decodieren der Pakete erforderlichen Nachrichten frühzeitig aus dem Atomic Buffer verworfen werden. Die Rekonstruktionsrate sinkt hier schließlich, da die zum Decodieren weiterer Pakete erforderlichen Nachrichten bereits aus dem Atomic Buffer verworfen wurden, noch bevor die Pakete beim Knoten eintreffen.

Abbildung 6.12 zeigt die Gesamtanzahl der empfangenen und gesendeten Pakete im Netzwerk bei unterschiedlichem Limit der Codierungskomplexität. Neben den Durchschnittswerten der Knoten sind auch Minimum und Maximum der Pakete aufgetragen, um die Schwankungsbreite zu visualisieren. Die Ergebnisse zeigen, dass bei einem Limit der Codierungskomplexität im Bereich von 2 bis 4 die kleinsten Abweichungen um den Durchschnitt zu erwarten sind, so dass das Codierungsschema bei diesem Limit die stabilste Leistungsfähigkeit aufweist. Bei einem

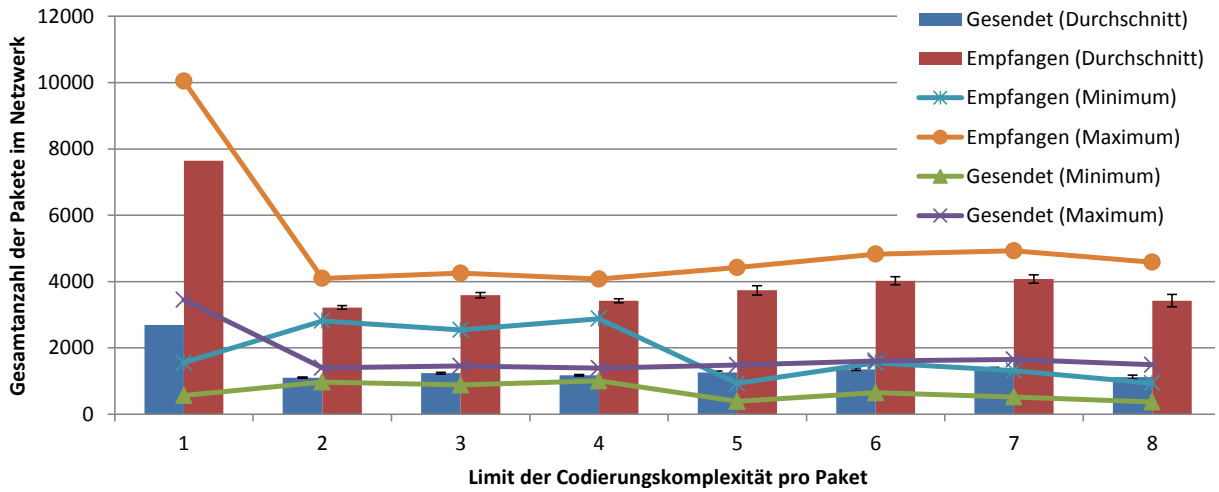

Abbildung 6.12 Gesamtanzahl der Pakete im Netzwerk bei unterschiedlichem Limit der Codierungskomplexität

Limit von 1 findet keine Codierung statt, so dass jede Nachricht einzeln verschickt wird. Es zeigt sich hier auch, dass die Codierung mit dem verwendeten Schema die Nachrichtenmenge auf etwa 30% reduziert. Eine höhere Codierungskomplexität sorgt dabei nicht für eine weitere Reduzierung der Nachrichtenmenge.

6.5.4 Nachrichtenverteilung in OCSSIM und DIP/DRIP

Die in Kapitel 4.3.1 diskutierten Ansätze der Protokolle zur Nachrichtenverteilung DIP und DRIP berücksichtigen anders als OCSSIM nicht die Koordination zur Nachrichtenverteilung von mehreren konkurrierenden Anwendungen im gleichen Netzwerk.

Zum Vergleich wurde daher in der Simulation ein Netzwerk mit einer Gittertopologie von 5 x 5 Knoten genutzt. Mittels ROSMARIN wurde dabei nur die Verteilung einer einzelnen Nachricht betrachtet. Ausgehend von einer Quelle wird die Nachricht an alle Knoten im Netzwerk verteilt, ohne überlappenden Verkehr in der Rückrichtung zu aktivieren.

Zur Verteilung des Datenpaketes muss ROSMARIN im Durchschnitt 1,3 Pakete pro Knoten senden und jeder Knoten empfängt durchschnittlich 3,91 Pakete von den Nachbarn. Dabei konnte in allen Simulationsdurchläufen eine Informationsabdeckung von 100% festgestellt werden. In diesem Versuchsaufbau findet jedoch keine aktive Netzwerkcodierung statt, da nur ein einzelnes atomares Paket im Netzwerk versendet wird. Die beobachteten Ergebnisse sind ausschließlich auf die Nutzung der lokalen opportunistischen Weiterleitungsstrategie zurückzuführen.

In Abbildung 6.13 ist die Informationsabdeckung von OCSSIM und den Protokollen DIP und Drip im zeitlichen Verlauf aufgetragen. Zusätzlich wird noch zwischen einer idealen Protokollvariante ohne Paketverluste und einer Protokollvariante bei Vorliegen von Rauschen (Lossy Model) unterschieden. Zur Vergleichbarkeit wurde bei DIP und Drip ein identischer Trickle-Timer genutzt. Die Ergebnisse sind über 10 Simulationsdurchläufe gemittelt. Das simulierte Rauschverhalten wurde einer

Messung auf physischen Knoten mittels des Testbetts SANDbed (siehe Abschnitt 6.2.2) entnommen. Im Gegensatz zu DIP und DRIP wirkt der Anstieg der Informationsabdeckung durch OCSSIM nicht kontinuierlich, sondern erfolgt innerhalb eines sehr kurzen Zeitraums nach ca. 87 Sekunden. Ursache für dieses Verhalten ist die Pufferung der zu verteilenden Nachrichten in OCSSIM bis der Zeitgeber zum automatischen Codieren t_{enc} aktiv wird. Das Verhalten kann für einzelne Nachrichten vermieden werden, wenn die zu verteilende Nachricht priorisiert wird. Werden mehrere Nachrichten aus der gleichen Quellen oder unterschiedlichen Quellen in der Nachbarschaft zugleich im Netzwerk verteilt, so tritt diese anfängliche Verzögerung nicht auf, da nicht auf den Zeitgeber zum automatischen Codieren gewartet werden muss, wenn mindestens thr_{enc} verschiedene Nachrichten vorliegen, um in ein gemeinsames Paket codiert werden zu können.

Die Informationsabdeckung der Knoten in OCCSIM erreicht im Experiment mit 25 Knoten innerhalb von 1,06 Sekunden einen Anteil von 95%, da auf einen Trickle Timer verzichtet wird und stattdessen die Verteilung unmittelbar über die Mehrwege-Topologie erfolgt. OCSSIM nutzt hier wie Drip die Unterdrückung des Sendens von mehrfach gehörten identischen Nachrichten. Anders als bei Drip kann die doppelte Nachricht aber mittels Mitlauschen auch erkannt werden, wenn sie zwischen benachbarten Knoten ausgetauscht und nicht regulär empfangen wird. Aufgrund der dichten Topologie konnten mittels des Lossy-Modells für OCSSIM keine signifikanten Abweichungen bei der Verteilungsgeschwindigkeit und der erreichten Informationsabdeckung ermittelt werden. Die Verteilung der Nachricht erfolgte bei Störungen auf einer einzelnen Verbindung unmittelbar über redundante Verbindungen in der Topologie.

6.5.5 Erkenntnisse zur Nutzung von OCCSIM

Die Ergebnisse der Evaluation des OCCSIM-Kommunkationsschemas lassen die folgenden Schlussfolgerungen zu:

Netzwerkcodierung in drahtlosen Sensornetzen mit Speicherbeschränkung

Prinzipiell ist die Nutzung von Netzwerkcodierung in drahtlosen Sensornetzen mit Speicherbeschränkung möglich. Dennoch ist die Einschränkung der Netzwerkgröße und der Anzahl der Nachbarn pro Knoten sinnvoll, da sonst auf den einzelnen Sensorknoten große Speichermengen bereitgehalten werden müssen. Diese sind nötig, um von allen potentiellen Nachrichtenquellen ausreichend Pakete zu sammeln, so dass ein erfolgreiches Decodieren möglich ist. Für die Limitierung der kommunizierenden Knoten spricht insbesondere die abfallende Rekonstruktionsrate der Nachrichten bei steigender Netzwerkgröße (vgl. Abschnitt 6.5.3.3). Für den effizienten Einsatz von Netzwerkcodierung sollte daher zusätzlich die Clusterung erwogen werden. Die vollständige Rekonstruktion aller versendeten Nachrichten kann OCCSIM nicht garantieren, daher eignet sich das Kommunikationsschema bevorzugt für Anwendungen, die über die erhaltenen Nachrichten ein Datenmodell extrapolieren können.

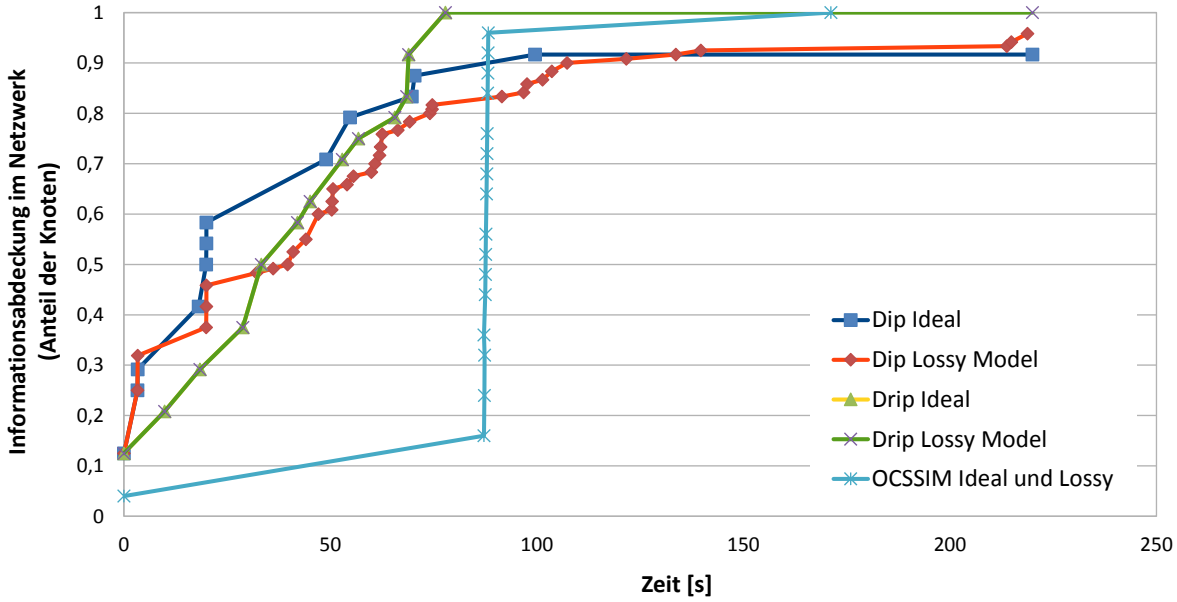

Abbildung 6.13 Informationsabdeckung von DIP, DRIP und OCSSIM bei Verteilung eines Wertes im zeitlichen Verlauf

Kommunikation konkurrierender Anwendungen durch Kombination von Netzwerkcodierung und opportunistischer Weiterleitung

Die Kombination aus beiden Ansätzen erlaubt die Kommunikation von konkurrierenden Anwendungen im gleichen Netzwerk. Sollen die anfallenden Nachrichten auf allen Knoten gespeichert werden, so kann die anfallende Datenmenge jedoch zu Problemen führen. Dies ist insbesondere dann der Fall, wenn:

- viele Quellen existieren, die eine hohe Anzahl Nachrichten erzeugen
- eine geringe Speichermenge auf den Knoten zur Zwischenspeicherung decodierter Nachrichten zur Verfügung steht
- zwischen Quellen und Senken sehr lange Pfade existieren, so dass viele Zwischenknoten zur Weiterleitung benötigt werden
- wenige Quellen existieren, die jeweils nur sporadisch einzelne Nachrichten erzeugen und mit hoher Latenz verteilen
- ein ungeeignetes Verfahren zur Koordination der Zugriffszeitpunkte auf das drahtlose Medium eingesetzt wird

Nachrichtenverteilung über unterschiedliche Pfade

OCCSIM ermöglicht die Zustellung von Nachrichten auch bei dynamischer Topologie und ohne Routing. Die Nachrichtenverteilung ist sehr zuverlässig, alle Knoten können erreicht werden (siehe Abschnitt 6.5.4). Das eingesetzte heuristische

Verfahren sorgt dafür, dass einzelne Knotenausfälle kaum einen Einfluss auf die Zustellung haben, solange das Netz nicht partitioniert wird. Trotzdem sollten einzelne Nachrichten priorisiert werden, um die durch Netzwerkcodierung induzierte Latenz zu vermeiden. Daher sollte OCCSIM nicht für zeitkritische Anwendungen verwendet werden.

6.6 Zusammenfassung

In diesem Kapitel wurde das ROSMARIN-Rahmenwerk zur Realisierung und Evaluation des OCSSIM-Kommunikationsschemas vorgestellt. Das Rahmenwerk wird dazu genutzt, die Konzepte aus Kapitel 5 zur Netzwerkcodierung und opportunistischen Verteilung von Nachrichten in einem Multisource-Multisink-Multicast Szenario umzusetzen. ROSMARIN bildet dabei Grundlage für Simulationen mittels Avrora und integriert Werkzeuge zur Erfassung des Knotenzustands während der Simulationslaufzeit.

Im Evaluationsteil des Kapitels wurde die Rekonstruktionsrate in einem Kommunikationsszenario evaluiert, das die Anfrage durch eine Senke und Rückantwort aller Knoten als Quelle vorsieht. Dabei wurde der Einfluss der verwendeten Pufferspeicher, sowie der Netzwerkgröße und Netzwerktopologie betrachtet. Es konnte ermittelt werden, dass OCSSIM sich bezüglich der Rekonstruktionsrate unabhängig von der Knotendichte der Topologie stabil verhält. Darüber hinaus wurde der Einfluss der Codierungskomplexität auf das Verhalten des Codierungsschemas und das Verteilungsverhalten von OCSSIM in Relation zu DIP und DRIP betrachtet.

7. Zusammenfassung und Ausblick

Drahtlose Netzwerke mit umfangreicher Sensorik finden inzwischen zunehmend als eingebettete Systeme oder in Form von Kleinstgeräten den Weg in den Alltag. Dort können immer mehr Daten erfasst werden und dienen unterschiedlichsten Anwendungen als Umwelt- und Nutzungskontext.

Die beschränkte Leistungsfähigkeit der einzelnen Geräte erfordert ein optimiertes Kommunikationsverhalten zwischen den Geräten, um einerseits leistungsfähige Anwendungen realisieren zu können, die auf der Verwendung räumlich verteilt gewonnener Sensordaten basieren. Andererseits sind einmal ermittelte Sensordaten häufig auch relevant für andere Anwendungszwecke oder Nutzer.

Daher muss die Übermittlung der Sensordaten von den *Datenquellen* zu unterschiedlichen und im Netzwerk verteilten *Datensenken* vorgesehen werden.

Bei dieser Form der Kommunikation überlagert sich der *konkurrierende Multicast-Datenverkehr* unterschiedlicher Anwendungen, so dass eine effiziente Koordination des Datenverkehrs im Netzwerk notwendig wird, um die vorhandenen Hardwareressourcen im Netzwerk möglichst zweckmäßig auszunutzen. Dabei sind sowohl die Aspekte der verfügbaren Netzwerkkapazität, der gewünschten Zuverlässigkeit und der Priorisierung einzelner Nachrichten zu berücksichtigen, als auch die Dynamik der aktuellen Netzwerktopologie.

Diese Arbeit behandelt die Entwicklung des OCSSIM-Kommunikationsschemas, das die Multicast-Kommunikation anwendungsübergreifend in einem einheitlichen Konzept für drahtlose Sensornetze integriert. Dazu werden durch das Kommunikationsschema unterschiedliche Datenquellen und Datensenken durch Nutzung von Netzwerkcodierung zu einer gemeinsamen Datenübertragung zusammengefasst, die gleichzeitig abgehandelt werden kann, unabhängig von der bevorzugten Kommunikationsrichtung der einzelnen Quellen.

Besondere Herausforderungen stellen die Eigenschaften drahtloser Sensornetze dar. Die eingeschränkte Netzwerkkapazität beschränkt das erreichbare Maximum der transportierten Datenmenge aus der Kombination gleichzeitig aktiver Anwendungen. Zudem sind im Sensornetz nur sehr *kleine Speichermengen* verfügbar, so dass dezentrale Techniken, wie das Hashen von IDs der Nachrichten oder spezialisierte Zwischenspeicher eingeführt, analysiert und hinsichtlich eines niedrigen Rechen- und Speicheraufwand zur Zwischenspeicherung und Verteilung der Nachrichten optimiert werden müssen. Daneben muss die Anzahl der benötigten Übertragungen, der zusätzliche Speicheraufwand und eine Anpassung an die Netzwerkdynamik für eine effiziente Kommunikation berücksichtigt werden.
Die Beiträge der Arbeit umfassen daher:

- Ein Verfahren zur heuristischen Netzwerkcodierung auf speicherbeschränkten Systemen in Kombination mit opportunistischer Weiterleitung durch das entworfene *OCSSIM-Kommunikationsschema*.

- Maßnahmen zur Priorisierung, schnellen Weiterleitung und Neukombination von Nachrichten. Diese können sowohl bei Anwendungen zur Datenverteilung, als auch zur Datensammlung genutzt werden.

- Strategien zur lokalen opportunistischen Weiterleitung von Sensordaten. Ohne zusätzlichen Aufwand für einen anwendungsspezifischen Topologieaufbau erlauben diese den Informationsaustausch zur Netzwerkcodierung.

- Implizite Empfangsbestätigungen für Pakete und Nachrichten, welche die Netzwerkcodierung unterstützen und zudem das effiziente Ausnutzen von Topologien mit mehreren Pfaden zwischen Datenquellen und Datensenken durch gezieltes Mithören ermöglichen.

- Die Realisierung eines robusten und modularen ROSMARIN-Rahmenwerkes zur Evaluation und Entwicklung verteilter Sensornetzanwendungen auf Basis des OCSSIM-Kommunikationsschemas.

7.1 Ergebnisse der Arbeit

Im Rahmen dieser Arbeit wurde das Kommunikationsschema *OCSSIM* zur parallelen Nachrichtenverteilung von sich überlagernden Multicast-Übertragungen entworfen und in [41] veröffentlicht. Zentrale Idee des Kommunikationsschemas ist die Kombination von *heuristischer Netzwerkcodierung* der Anwendungsnachrichten und der Nutzung opportunistischer Verfahren zur Mehrwege-Weiterleitung im Netzwerk.
Die Verwendung von heuristischer Netzwerkcodierung trägt vordringlich zur Verbesserung der Zuverlässigkeit, als auch zu einer besseren Auslastung vorhandener Netzwerkkapazitäten und Hardwareressourcen bei. Zu diesem Zweck werden die Nachrichten der verschiedenen Anwendungen durch die Netzwerkknoten in

gemeinsame Pakete codiert. Bei der Codierung entstehen neue *Nachrichtenkombinationen*, die zusätzliche Redundanz in den übermittelten Datenverkehr einbringen können. Dies erlaubt es, dass andere Netzwerkknoten die enthaltenen Nachrichten später decodieren können, um die enthaltenen Sensordaten zu rekonstruieren.

Das erneute Kombinieren von codierten Paketen im Netzwerk eröffnet dabei effiziente Möglichkeiten zur Ausnutzung von Netzwerktopologien mit mehreren Pfaden zwischen Datenquellen und Datensenken. Hierzu wurde ein auf ratenlosen und linearen Codes basierendes, Codierungsschema identifiziert und für die Nutzung bei einem Multisink-Multicast-Kommunikationsszenario in drahtlosen Sensornetzen abgewandelt. Das Codierungsschema ermöglicht es, ohne vorhandene Topologieinformationen die Codierung und Decodierung auch auf leistungsschwachen und speicherbeschränkten Sensorknoten durchzuführen, so dass eine hohe Rekonstruktionsrate der Sensordaten an der Datensenke erreicht wird.

Im Mittelpunkt des ermittelten opportunistischen Verfahrens zur Mehrwege-Weiterleitung steht das effiziente Verteilen und Sammeln von Daten konkurrierender Anwendungen bei einer unbekannten oder hoher Dynamik unterliegenden Netzwerktopologie. Die Kombination mit Netzwerkcodierung erlaubt den Verzicht auf zusätzliche Routing-Mechanismen.

In Folge ist eine Nutzung des OCSSIM-Kommunikationsschemas unabhängig von Position oder Anzahl der Datensenken zur Datenverteilung im Netzwerk möglich. Damit wurde eine Möglichkeit geschaffen, den zeitweisen Anwendungsbetrieb ohne eine statische und dauerhaft verfügbare Datensenke zu ermöglichen. Die Netzwerkcodierung sorgt dafür, dass verteilt gespeicherte Kommunikationsdaten zu einem späteren Zeitpunkt wieder rekonstruiert werden können.

Darüber hinaus kann eine Priorisierung von Informationen zur Fehlerbehandlung oder die Filterung von Monitoring-Informationen aus codierten Datenströmen realisiert werden. Netzwerkknoten wählen dazu den Grad der redundanten Mehrwege-Weiterleitung anhand der zu erwartenden Übertragungslatenz und gewählten Priorisierung.

Die beiden sich ergänzenden Prozesse Reassembling und Fast-Forwarding koordinieren das Kommunikationsschema, indem kontrolliert wird, welche Pakete weitergeleitet und unterdrückt werden.

Der Reassembling-Prozess sieht die Neukombination von Nachrichten vor, um die bestehende Redundanz in der Struktur der Netzwerktopologie ausnutzen zu können und fehlende Redundanz gegebenenfalls durch geeignete Codierung hinzuzufügen. Dies ermöglicht eine Steuerung der Zuverlässigkeit unter Berücksichtigung der Netzwerkkapazität.

Der Fast-Forwarding-Prozess sorgt für eine schnelle Weiterleitung leicht zu decodierender Nachrichten und dient sowohl der schnellen Verteilung priorisierter Nachrichten, als auch einer Regulierung der Nachrichtenmenge bei der Initiierung neuer Nachrichtenübertragungen.

Das entworfene OCSSIM-Kommunikationsschema sieht dabei sowohl die gemeinsame Übertragung von Daten aus unterschiedlichen Quellen der selben Anwendung, als auch die gemeinsame Übertragung der Daten unterschiedlicher Anwendungen vor. Redundante Pfade der Netzwerktopologie werden aktiv durch opportunistische Methoden zur Mehrwege-Weiterleitung ausgenutzt.

Im Rahmen der Arbeit wurde dazu ein lokales Verfahren zur Analyse mitgehörter Pakte von benachbarten Netzwerkknoten entworfen, das die ermittelten Informationen aus dem Netzbereich nutzen kann, um Redundanzen der Kommunikation zu steuern und daraufhin die Auswahl des zweckmäßigeren Weiterleitungsprozesses zu treffen.

Der beabsichtigte Verzicht auf Empfangsbestätigungen im Kommunikationsschema dient sowohl der Verringerung der benötigten Netzwerkkapazität und Anzahl der Sendevorgänge. Zugleich ist aber auch der Verlust von Funktionalität für zuverlässige Übertragungen, wie z. B. eine automatische Sendewiederholung die Folge. OCSSIM führt daher eine implizite Form der Empfangsbestätigung für Nachrichten und Pakete ein, die auf dem Mithören von zuvor gesendeten Daten basiert, die durch einen benachbarten Netzwerkknoten weitergeleitet wurden. So kann gezielt eine Sendewiederholung oder die Sendeunterdrückung von bekannten Daten herbeigeführt werden.

Das modulare Rahmenwerk *ROSMARIN* wurde in Laufe der Arbeit zur Evaluation des OCSSIM-Kommunikationsschemas auf ressourcenbeschränkter Hardware entworfen und realisiert. Dabei konnten vorherige Betrachtungen existierender Analyse- und Bewertungswerkzeuge in drahtlosen Sensornetzen in [42] genutzt werden, um geeignete Analysewerkzeuge zur Nutzung auf speicherarmer, verteilter Sensorhardware zu realisieren.

Auf Grundlage des Rahmenwerks wurde eine prototypische Realisierung des OCSSIM-Kommunikationsschemas erstellt und in Sensornetzwerken unterschiedlicher Größe und Netzwerktopologie mittels des Emulators *Avrora* analysiert.

Das ROSMARIN-Rahmenwerk eröffnet dabei eine Grundlage zur Realisierung von Anwendungen, welche die Konzepte der Netzwerkcodierung und der opportunistischen Weiterleitung aktiv auf Netzwerk- und Transportschicht nutzen wollen. Dazu erlaubt es die Anpassung, Parametrisierung und Erweiterung dieser Kommunikationskonzepte, und integriert geeignete Monitoring- und Analysewerkzeuge zur Bewertung der Datenverteilung und Übertragungszuverlässigkeit.

Mittels simulativer Experimente konnte der Zusammenhang zwischen den Rekonstruktionseigenschaften der Kommunikationsdaten und der Anzahl und Dichte der Sensorknoten in unterschiedlichen Topologien bestimmt werden. Mess- bzw. Sendefrequenz der Anwendungen, sowie die verfügbare Speichermenge der Knoten erweisen sich als effektive Grenzen für die Anzahl kooperierender Netzwerkteilnehmer im OCSSIM-Kommunikationsschema. Es konnte dabei gezeigt werden, dass die Netzwerkcodierung bereits in kleinen Sensornetzen mit bis zu 50 Sensorknoten nutzbringend umgesetzt werden kann, bei denen ein Zwischenspeicher von weniger als 2,5 Kilobyte für weiterzuleitende Informationen je Knoten zur

Verfügung steht. Insbesondere bei Topologieänderungen durch Ausfall von über 80% der Sensorknoten, kann ein überwiegender Anteil der verbleibenden Kommunikationsdaten durch eine vorhandene Senke weiterverwendet werden.

7.2 Ausblick

Die Ergebnisse dieser Arbeit konzentrieren sich auf die Kombination des XOR-Codierungsschemas mit opportunistischer Weiterleitung. Zahlreiche Neu- und Weiterentwicklungen von Codierungsverfahren wie Raptor-Codes legen eine vergleichende Untersuchung unterschiedlicher Codierungsverfahren für eine weitere Optimierung von Multisink-Multicast Kommunikationsszenarien in drahtlosen Sensornetzen nahe. ROSMARIN bietet als Rahmenwerk bereits ein austauschbares Modul zur Netzwerkcodierung, das zukünftige Evaluationen mit anderen Codierungsverfahren stark vereinfachen kann. Insbesondere auf Sensorknoten mit einer deutlich größeren Leistungsfähigkeit und Speicherausstattung, als auf dem hier genutzten Knotentyp, versprechen komplexere Codierungsverfahren weitere Optimierungsmöglichkeiten für effizienten Datentransport mit Netzwerkcodierung.

Ebenso lässt sich das in der Evaluation vorrangig verwendete Medienzugriffsverfahren CSMA durch andere energieeffizientere Verfahren, wie TDMA oder YA-MAC [119, 120] austauschen oder durch ein Medienzugriffsverfahren, das weniger zur Kollision von Datenpaketen neigt, ersetzen. Insbesondere die Rekonstruktionsraten können hier von weiteren Verbesserungen des Medienzugriffs profitieren.

Ein weiterer Aspekt der in dieser Arbeit nur im Zusammenhang mit opportunistischer Kommunikation angerissen wurde, ist die Nutzung von mobilen Knoten in einem Multisink-Multicast-Kommunikationsszenario. Diese könnten im Rahmen weiterer Forschungsarbeiten, sowohl für eine gezieltere Weiterleitung, als auch für eine Optimierung der Decodierungsmöglichkeiten eingesetzt werden.

Literaturverzeichnis

[1] Jamshid Abouei, Siavash Fazeli Dehkordy, Konstantinos N. Plataniotis, and Subbarayan Pasupathy. Raptor codes in wireless body area networks. In *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 2143–2147. IEEE, sep 2011.

[2] Rudolf Ahlswede, Ning Cai, Shuo-yen Robert Li, and Raymond W Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, jul 2000.

[3] Mert Akdere, Cemal Çagatay Bilgin, Ozan Gerdaneri, Ibrahim Korpeoglu, Özgür Ulusoy, and Ugur Çetintemel. A comparison of epidemic algorithms in wireless sensor networks. *Computer Communications*, 29:2450–2457, 2006.

[4] Muhammad Hamad Alizai, Olaf Landsiedel, Jó Ágila Bitsch Link, Stefan Götz, and Klaus Wehrle. Bursty traffic over bursty links. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems - SenSys '09*, page 71, New York, New York, USA, 2009. ACM Press.

[5] Salah A Aly, Ahmed E Kamal, and Osameh M. Al-Kofahi. Network protection codes: Providing self-healing in autonomic networks using network coding. *Computer Networks*, 56(1):99–111, jan 2012.

[6] Salah A Aly, Zhenning Kong, and Emina Soljanin. Raptor Codes Based Distributed Storage Algorithms for Wireless Sensor Networks. In *ISIT 2008. IEEE International Symposium on Information Theory*, pages 2051 – 2055, Toronto, jul 2008. IEEE.

[7] Salah A Aly, Zhenning Kong, and Emina Soljanin. Fountain Codes Based Distributed Storage Algorithms for Large-scale Wireless Sensor Networks. In *International Conference on Information Processing in Sensor Networks, 2008. IPSN '08*, pages 171 – 182, St. Louis, feb 2009. IEEE.

[8] Antonios Argyriou. Wireless Network Coding with Improved Opportunistic Listening. *IEEE Transactions on Wireless Communications*, 8(4):2014–2023, apr 2014.

[9] Rocío Arroyo-Valles, Antonio G. Marques, and Jesús Cid-Sueiro. Energy-efficient Selective Forwarding for Sensor Networks. In *DCOSS 2008. International Conference on Distributed Computing in Sensor Systems and Workshops*, 2008.

[10] Atmel. Atmel 8-bit AVR Instruction Set, revision J, updated: 07/2014, jul 2014.

[11] Christopher L Barrett, Stephan J Eidenbenz, Lukas Kroc, Madhav Marathe, and James P Smith. Parametric Probabilistic Routing in Sensor Networks. *Mobile Networks and Applications*, 10(4):529–544, aug 2005.

[12] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. *ACM SIGCOMM Computer Communication Review*, 34(1):69, jan 2004.

[13] Rui Cao and Liuqing Yang. Short Paper: Reliable Transport and Storage Protocol with Fountain Codes for Underwater Acoustic Sensor Networks. In *WUWNet'10. Proceedings of the Fifth ACM International Workshop on UnderWater Networks*, page Article No. 14, Woods Hole, 2010. ACM.

[14] Mary-Luc Champel, Kévin Huguenin, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. LT Network Codes: Low Complexity Network Codes. Technical report, INRIA Rennes, Rennes, France, 2009.

[15] Roja Chandanala and Radu Stoleru. Network coding in duty-cycled sensor networks. In *INSS 2010. Seventh International Conference on Networked Sensing Systems*, pages 203–210, Kassel, jun 2010. IEEE.

[16] Tao Chen, Deke Guo, Yuan He, Honghui Chen, Xue Liu, and Xueshan Luo. A Bloom filters based dissemination protocol in wireless sensor networks. *Ad Hoc Networks*, 11(4):1359–1371, jun 2013.

[17] Seong-Min Choi, Kyogu Lee, and Joon-Sang Park. Fast Parallel Implementation for Random Network Coding on Embedded Sensor Nodes. *International Journal of Distributed Sensor Networks*, 2014:1–8, 2014.

[18] S. K. Das, A. Di Saverio, G Ghidini, A Navarra, and C. M. Pinotti. Broadcast Analysis in Dense Duty-Cycle Sensor Networks. In *ICUIMC' 2012. Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, page Article No. 16, Kuala Lumbur, 2012. ACM.

[19] Rodolfo de Paz Alberola and Dirk Pesch. AvroraZ: Extending Avrora with an IEEE 802.15.4 Compliant Radio Chip Model. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks - PM2HW2N '08*, pages 43–50, New York, New York, USA, oct 2008. ACM Press.

[20] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. ReInForM: Reliable information forwarding using multiple paths in sensor networks. In *28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN '03.*, pages 406–415. Ieee, 2003.

[21] Nildo dos Santos Ribeiro Júnior, Marcos A. M. Vieira, Luiz F. M. Vieira, and Omprakash Gnawali. CodeDrip : Data Dissemination Protocol with Network Coding for Wireless Sensor Networks. In *11th European Conference, EWSN 2014*, pages pp 34–49, Oxford, UK, 2014. Springer.

[22] Falko Dressler, Reinhard German, and Bettina Krüger. Adaptive Data Dissemination in Sensor Networks Using WPDD. In *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, pages 827–832, 2007.

[23] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - A lightweight and flexible operating system for tiny networked sensors. In *Proceedings - Conference on Local Computer Networks, LCN*, pages 455–462, 2004.

[24] Shane B Eisenman, Nicholas D Lane, and Andrew T Campbell. Techniques for Improving Opportunistic Sensor Networking Performance. In *Lecture Notes in Computer Science Volume 5067: Distributed Computing in Sensor Systems*, pages 157–175. Springer Berlin Heidelberg, 2008.

[25] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, jan 1956.

[26] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. Network Coding : An Instant Primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, jan 2006.

[27] Christina Fragouli and Emina Soljanin. Network Coding Applications. *Foundations and Trends in Networking*, 2(2):135–269, 2007.

[28] Euhanna Ghadimi, Olaf Landsiedel, Pablo Soldati, Simon Duquennoy, and Mikael Johansson. Opportunistic Routing in Low Duty-Cycled Wireless Sensor Networks. *ACM Trans. Embedd. Comput. Syst*, 2013.

[29] Stephan M Günther, Maximilian Riemensberger, and Wolfgang Utschick. Efficient GF Arithmetic for Linear Network Coding using Hardware SIMD Extensions. In *2014 International Symposium on Network Coding (NetCod)*, pages 1–6. IEEE, jun 2014.

[30] Shuo Guo, Yu Gu, Bo Jiang, and Tian He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. In *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*, page 133, New York, New York, USA, 2009. ACM Press.

[31] Zheng Guo, Bing Wang, Peng Xie, Wei Zeng, and Jun-Hong Cui. Efficient error recovery with network coding in underwater sensor networks. *Ad Hoc Networks*, 7(4):791–802, jun 2009.

[32] Bernhard Haeupler. Analyzing Network Coding Gossip Made Easy. In *STOC '11. Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 293 – 302. ACM, oct 2011.

[33] Song Han, Zifei Zhong, Hongxing Li, Guihai Chen, Edward Chan, and Aloysius K. Mok. Coding-Aware Multi-path Routing in Multi-Hop Wireless Networks. In *IPCCC 2008. IEEE International Performance, Computing and Communications Conference*, pages 93–100. IEEE, dec 2008.

[34] Peter Hebden and Adrian R Pearce. Data-Centric Routing using Bloom Filters in Wireless Sensor Networks. In *2006 Fourth International Conference on Intelligent Sensing and Information Processing*, pages 72–77. IEEE, dec 2006.

[35] Wendi B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, and Mark A. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18(1):6–14, jan 2004.

[36] Anton Hergenröder and Jens Horneber. Facing challenges in evaluation of WSN energy efficiency with distributed energy measurements. In *IWCMC 2011 - 7th International Wireless Communications and Mobile Computing Conference*, pages 1004–1009, Istanbul, 2011. IEEE.

[37] Anton Hergenröder, Jens Horneber, Detlev Meier, Patrick Armbruster, and Martina Zitterbart. Distributed energy measurements in wireless sensor networks. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, page 299, 2009.

[38] Tracey Ho, Ralf Koetter, Muriel Médard, D.R. Karger, and Michelle Effros. The benefits of coding over routing in a randomized setting. *IEEE International Symposium on Information Theory, 2003. Proceedings.*, 2003.

[39] Tracey Ho, Muriel Médard, Ralf Koetter, David R Karger, Michelle Effros, Jun Shi, and Ben Leong. A Random Linear Network Coding Approach to Multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, oct 2006.

[40] Peter Adam Höher. *Grundlagen der digitalen Informationsübertragung*. Springer Fachmedien Wiesbaden, Wiesbaden, 2. auflage edition, 2013.

[41] Jens Horneber. Opportunistic forwarding of network-coded multicast traffic in wireless sensor networks. In *The 10th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 66–71, Larnaca, Cyprus, oct 2014. IEEE.

[42] Jens Horneber and Anton Hergenroder. A Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, 16(4):1820–1838, jan 2014.

[43] Wouter Horré, Sam Michiels, Wouter Joosen, and Pierre Verbaeten. DAVIM: Adaptable Middleware for Sensor Networks. *IEEE Distributed Systems Online*, 9(1):1–11, jan 2008.

[44] I-Hong Hou, Yu-En Tsai, Tarek F Abdelzaher, and Indranil Gupta. AdapCode: Adaptive Network Coding for Code Updates in Wireless Sensor Networks. In *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pages 1517–1525, Phoenix, apr 2008. IEEE.

[45] Jacek Ilow, Shreyas Rangappa, and Nauman Aslam. Energy efficient broadcasting in WSNs with cocasting and power control. In *2012 IEEE International Conference on Communications (ICC)*, pages 6288–6292. IEEE, jun 2012.

[46] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2 – 16, 2003.

[47] Nishant Jain, Sanjeev Sharma, and Santosh Sahu. Efficient Flooding for a Large Sensor Networks using Network Coding. *International Journal of Computer Applications*, 30(9):7–10, sep 2011.

[48] Raja Jurdak, Antonio G. Ruzzelli, Gregory M. P. O'Hare, and Russell Higgs. Directed broadcast with overhearing for sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):1–27, feb 2010.

[49] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. XORs in The Air : Practical Wireless Network Coding. In *ACM SIGCOMM Computer Communication Review - Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 243–254, Pisa, Italy, 2006. ACM.

[50] Majid Khabbazian, Fabian Kuhn, Nancy Lynch, Muriel Médard, and Ali ParandehGheibi. MAC Design for Analog Network Coding. In *Proceedings of the 7th ACM ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing - FOMC '11*, pages 42–51, New York, New York, USA, 2011. ACM Press.

[51] Zubair Khalid, Norsheila Fisal, and Mohd. Rozaini. A Survey of Middleware for Sensor and Network Virtualization. *Sensors*, 14(12):24046–24097, 2014.

[52] Junwhan Kim and Binoy Ravindran. Opportunistic Real-Time Routing in Multi-Hop Wireless Sensor Networks. In *Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09*, pages 2197 – 2201, New York, New York, USA, 2009. ACM Press.

[53] MinJi Kim, Daniel Lucani, Xiaomeng Shi, Fang Zhao, and Muriel Médard. Network coding for multi-resolution multicast. In *Proceedings - IEEE INFOCOM*, 2010.

[54] Kevin Klues, Gregory Hackmann, Octav Chipara, and Chenyang Lu. A component-based architecture for power-efficient media access control in wireless sensor networks. In *Proceedings of the 5th iIternational Conference on Embedded Networked Sensor Systems - SenSys '07*, pages 59–72, New York, New York, USA, 2007. ACM Press.

[55] Ralf Koetter and Muriel Médard. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, oct 2003.

[56] Dimitrios Koutsonikolas, Chih-Chun Wang, and Y Charlie Hu. Efficient Network-Coding-Based Opportunistic Routing Through Cumulative Coded Acknowledgements. *IEEE/ACM Transactions on Networking*, 19(5):1368–1381, oct 2011.

[57] Andreas Kuntz. *Dienstbasierte Kommunikation über unzuverlässige drahtlose Verbindungen für selbstorganisierende Sensor-Aktor-Netze*. KIT Scientific Publishing, Karlsruhe, 2011.

[58] Pradeep Kyasanur, Romit Choudhury, and Indranil Gupta. Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks. In *MASS 2006. IEEE International Conference on Mobile Ad Hoc and Sensor Sysetems*, pages 91–100. IEEE, oct 2006.

[59] Olaf Landsiedel, Euhanna Ghadimi, Simon Duquennoy, and Mikael Johansson. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In *IPSN '12 Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 185–196. ACM, 2012.

[60] Michael Langberg and Muriel Médard. On the Multiple Unicast Network Coding, Conjecture. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 222–227. IEEE, sep 2009.

[61] Jilin Le, John C S Lui, and Dah-ming Chiu. DCAR : Distributed Coding-Aware Routing in Wireless Networks. *IEEE Transactions on Mobile Computing*, 9(4):596–608, apr 2010.

[62] April Rasala Lehman and Eric Lehman. Complexity Classification of Network Information Flow Problems. In *SODA '04 Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 142–150, Philadelphia, 2004.

[63] Ilias Leontiadis, Christos Efstratiou, Cecilia Mascolo, and Jon Crowcroft. SenShare: Transforming Sensor Networks into Multi-application Sensing Infrastructures. In Gian Pietro Picco and Wendi Heinzelman, editors, *Lecture*

Notes in Computer Science Volume 7158: Wireless Sensor Networks (EWSN 2012), pages 65–81. Springer Berlin Heidelberg, 2012.

[64] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. In *Ambient Intelligence*, pages 115–148. Springer, 2005.

[65] Philip Levis. TinyOS Enhancement Proposal (TEP) 111: message_t.

[66] Philip Levis and David Culler. Mat{é}: a tiny virtual machine for sensor networks. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X 2002)*, volume 36, pages 85–95, 2002.

[67] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *NSDI'04 Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 2–2. USENIX Association Berkeley, 2004.

[68] Li Li, Ramachandran Ramjee, Milind Buddhikot, and Scott Miller. Network coding-based broadcast in mobile ad hoc networks. In *Proceedings - IEEE INFOCOM*, pages 1739–1747, 2007.

[69] Shan-Shan Li, Pei-Dong Zhu, Xiang-Ke Liao, Wei-Fang Cheng, and Shao-Liang Peng. Energy Efficient Multipath Routing Using Network Coding in Wireless Sensor Networks. In T. Kunz and S.S. Ravi, editors, *Lecture Notes in Computer Science Volume 4104: Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW 2006)*, pages 114–127. Springer Berlin Heidelberg, 2006.

[70] Shuo-Yen Robert Li, Raymond W Yeung, and Ning Cai. Linear Network Coding. *IEEE Transactions on Information Theory*, 49(2):371–381, feb 2003.

[71] Zongpeng Li and Baochun Li. Network Coding: The Case of Multiple Unicast Sessions. In *Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.

[72] Kaisen Lin and Philip Levis. Data Discovery and Dissemination with DIP. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pages 433–444. IEEE, apr 2008.

[73] Ting Liu and Margaret Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. *ACM SIGPLAN Notices*, 38(10):107, oct 2003.

[74] Feng Lu, Liang-tien Chia, and Kok-Leong Tay. NBgossip - Neighborhood Gossip with Network Coding Based Message Aggregation. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007*, pages 1 – 12, Pisa, Italy, 2007. IEEE.

[75] Desmond S Lun, Tracey Ho, Niranjan Ratnakar, Muriel Médard, and Ralf Koetter. Network Coding in Wireless Networks: A survey of techniques for efficient operation of coded wireless packet networks. In Frank H. P. Fitzek and Marcos D. Katz, editors, *Cooperation in Wireless Networks: Principles and Applications*, chapter 5, pages 127–161. Springer, 2006.

[76] Markus Jung. *Integration von Iris-Sensorknoten in den Simulator Avrora*. Bachelorarbeit, Karlsruher Institut für Technologie, 2012.

[77] Takahiro Matsuda, Taku Noguchi, and Tetsuya Takine. Survey of Network Coding and Its Applications. *IEICE TRANS. COMMUN*, E94-B(3):698–717, mar 2011.

[78] Memsic. IRIS Wireless Measurement System.

[79] Sam Michiels, Wouter Horré, Wouter Joosen, and Pierre Verbaeten. DAViM: a dynamically adaptable virtual machine for sensor networks. In *Proceedings of the international workshop on Middleware for sensor networks - MidSens '06*, pages 7–12, New York, New York, USA, 2006. ACM Press.

[80] Dong Nguyen, Tuan Tran, Thinh Nguyen, and Bella Bose. Wireless Broadcast Using Network Coding. *IEEE Transactions on Vehicular Technology*, 58(2):914–925, feb 2009.

[81] Bin Ni, Naveen Santhapuri, Zifei Zhong, and Srihari Nelakuditi. Routing with opportunistically coded exchanges in wireless mesh networks. In *2006 2nd IEEE Workshop on Wireless Mesh Networks*, pages 157–159. IEEE, 2006.

[82] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking - MobiCom '99*, pages 151–162, 1999.

[83] Pouya Ostovari, Jie Wu, and Abdallah Khreishah. Network Coding Techniques for Wireless and Sensor Networks. In Habib M. Ammari, editor, *The Art of Wireless Sensor Networks - Volume 1: Fundamentals*, chapter 5, pages 129–162. Springer Berlin Heidelberg, 2013.

[84] Joon-sang Park, Mario Gerla, Desmond S Lun, Yunjung Yi, and Muriel Médard. Codecast: a network-coding-based ad hoc multicast protocol. *IEEE Wireless Communications*, 13(5):76–81, oct 2006.

[85] Yang Peng, WenZhan Song, Renjie Huang, Mingsen Xu, Bhrooz Shirazi, Richard LaHusen, and Guangyu Pei. Cacades: A reliable dissemination protocol for data collection sensor network. In *2009 IEEE Aerospace conference*, pages 1–10, Big Sky, MT, mar 2009. IEEE.

[86] Chrishan Perera. *Implementierung und Evaluierung eines Bloomfilters für Multipath-Forwarding in drahtlosen Sensornetzen*. Bachelorarbeit, Karlsruher Institut für Technologie, 2014.

[87] Ca Van Phan, Kikyung Baek, and Jeong Geun Kim. Opportunistic transmission for wireless sensor networks under delay constraints. In Osvaldo Gervasi and Marina L. Gavrilova, editors, *Computational Science and Its Applications - ICCSA 2007*, pages 858–871. Springer Berlin Heidelberg, 2007.

[88] Jalaluddin Qureshi, Chuan Heng Foh, and Jianfei Cai. Optimal solution for the index coding problem using network coding over GF(2). In *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 209–217, Seoul, jun 2012. IEEE.

[89] Marjan Radi, Behnam Dezfouli, Kamalrulnizam Abu Bakar, and Malrey Lee. Multipath routing in wireless sensor networks: survey and research challenges. *Sensors (Basel, Switzerland)*, 12(1):650–85, jan 2012.

[90] Rolf Riesen. Communication Patterns. In *Workshop on Communication Architecture for Clusters {CAC'06}*, 2006.

[91] Z. Rosberg, R. P. Liu, A. Y.dong, L. D.tuan, and S. Jha. ARQ with implicit and explicit ACKs in wireless sensor networks. In *2008 IEEE GLOBECOM - Global Telecommunications Conference*, pages 50–55, 2008.

[92] Gunnar Schaefer, François Ingelrest, and Martin Vetterli. Potentials of Opportunistic Routing in Energy-Constrained Wireless Sensor Networks. In Utz Roedig and Cormac J. Sreenan, editors, *Lecture Notes in Computer Science Volume 5432: Wireless Sensor Networks*, pages 118–133. Springer Berlin Heidelberg, 2009.

[93] Mingkai Shao, Xiaolin Wu, and Nima Sarshar. Rainbow Network Flow with Network Coding. *2008 Fourth Workshop on Network Coding, Theory and Applications*, pages 1–6, 2008.

[94] Amin Shokrollahi. Raptor codes. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, pages 36–36, Chicago, jun 2004. Ieee.

[95] Hnin Yu Shwe, Xiaohong Jiang, and Susumu Horiguchi. Efficient network coding for power saving in wireless sensor networks. In *2009 15th Asia-Pacific Conference on Communications*, pages 36–39. IEEE, oct 2009.

[96] Jorge M. Soares, Mirko Franceschinis, Rui M. Rocha, Wansheng Zhang, and Maurizio a. Spirito. Opportunistic Data Collection in Sparse Wireless Sensor Networks. *EURASIP Journal on Wireless Communications and Networking*, 2011:1–20, 2011.

[97] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait : An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05*, pages 252–259, 2005.

[98] Fred Stann, John Heidemann, Rajesh Shroff, and Muhammad Zaki Murtaza. RBP: robust broadcast propagation in wireless networks. In *Proceedings of the 4th ACM Conference on Embedded Network Sensor Systems - SenSys '06*, pages 85–98. ACM Press, 2006.

[99] Yi-Yu Su, Shiow-Fen Hwang, and Chyi-Ren Dow. An Efficient Multi-Source Multicast Routing Protocol in Mobile Ad Hoc Networks. In *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, volume 1, pages 8–14. IEEE, jul 2005.

[100] Jay Kumar Sundararajan, Devavrat Shah, and Muriel Médard. ARQ for network coding. In *2008 IEEE International Symposium on Information Theory*, volume 0627021, pages 1651–1655. IEEE, jul 2008.

[101] Nikolaos Thomos and Pascal Frossard. Raptor network video coding. In *Proceedings of the international workshop on Workshop on mobile video - MV '07*, page 19, New York, New York, USA, 2007. ACM Press.

[102] B.L. Titzer, D.K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks*, pages 477–482. IEEE, 2005.

[103] Todor Konstadinov Mechkov. *A Publisher/Subscriber Monitoring Service for Network Coding-Based Wireless Sensor Networks*. Diplomarbeit, Karlsruher Institut für Technologie, 2015.

[104] Shun Tokuyama, Tatsuhiro Tsuchiya, and Tohru Kikuno. Gossiping with Network Coding. In *2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing*, pages 63–70. IEEE, dec 2011.

[105] Alberto Lopez Toledo and Xiaodong Wang. Efficient Multipath in Sensor Networks using Diffusion and Network Coding. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 87–92, Princeton, NJ, 2006. IEEE.

[106] Gilman Tolle and David Culler. Design of an application-cooperative management system for wireless sensor networks. In *EWSN 2005. Proceeedings of the Second European Workshop on Wireless Sensor Networks*, pages 121–132. IEEE, 2005.

[107] Stefan Valentin, Hermann S Lichte, Holger Karl, and Guillaume Vivier. Cooperative wireless networking beyond store-and-forward : Perspectives in PHY and MAC design. *Wireless Personal Communications*, pages 1–24, mar 2008.

[108] Thiemo Voigt, Utz Roedig, Olaf Landsiedel, Kasun Samarasinghe, and Mahesh Bogadi Shankar Prasad. Practical Network Coding in Sensor Networks: Quo Vadis? In *The Third International Workshop on Networks of Cooperating Objects (CONET2012)*, pages 79 – 82, Beijing, China, apr 2012.

[109] Chieh Y. Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. Pump-Slowly, Fetch-Quickly (PSFQ): A reliable transport protocol for sensor networks. *IEEE Journal on Selected Areas in Communications*, 23:862–872, 2005.

[110] Feng Wang and Jiangchuan Liu. Duty-Cycle-Aware Broadcast in Wireless Sensor Networks. In *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, pages 468–476. IEEE, apr 2009.

[111] Lei Wang, Yuwang Yang, and Wei Zhao. Network coding-based multipath routing for energy efficiency in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):115, mar 2012.

[112] Lei Wang, Yuwang Yang, Wei Zhao, Lei Xu, and Shaohua Lan. Network-Coding-Based Energy-Efficient Data Fusion and Transmission for Wireless Sensor Networks with Heterogeneous Receivers. *International Journal of Distributed Sensor Networks*, 2014:1–13, mar 2014.

[113] Qiang Wang, Yaoyao Zhu, and Liang Cheng. Reprogramming wireless sensor networks: Challenges and approaches. *IEEE Network*, 20(3):48–55, 2006.

[114] Shuai Wang, Athanasios Vasilakos, Hongbo Jiang, Xiaoqiang Ma, Wenyu Liu, Kai Peng, Bo Liu, and Yan Dong. Energy Efficient Broadcasting Using Network Coding Aware Protocol in Wireless Ad Hoc Network. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, jun 2011.

[115] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, sep 1991.

[116] Jörg Widmer and Jean-Yves Le Boudec. Network Coding for Efficient Communication in Extreme Networks. In *Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05*, pages 284–291, New York, New York, USA, 2005. ACM Press.

[117] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks, 2003.

[118] Feng Xue and P.R. Kumar. The Number of Neighbors Needed for Connectivity of Wireless Networks. *Wireless Networks*, 10(2):169–181, mar 2004.

[119] Poonam Yadav and Julie A. McCann. EBS: decentralised slot synchronisation for broadcast messaging for low-power wireless embedded systems. In *Proceedings of the 5th International Conference on Communication System Software and Middleware - COMSWARE '11*, pages 1–6, New York, New York, USA, 2011. ACM Press.

[120] Poonam Yadav and Julie A. McCann. YA-MAC: Handling unified unicast and broadcast traffic in Multi-hop Wireless Sensor Networks. In *DCOSS 2011. International Conference on Distributed Computing in Sensor Systems and Workshops*, pages 1–9. IEEE, jun 2011.

[121] Raymond W Yeung, Shuo-Yen Robert Li, Ning Cai, and Zhen Zhang. Network Coding Theory. *Foundations and Trends in Communications and Information Theory*, 2(4/5), 2006.

[122] Fang Zhao and Muriel Médard. On analyzing and improving COPE performance. *2010 Information Theory and Applications Workshop (ITA)*, pages 1–6, jan 2010.

[123] Jinyi Zhou, Shutao Xia, Yong Jiang, and Haitao Zheng. Decoding buffer management in practical wireless network coding. In *2011 International Symposium on Network Coding, NETCOD 2011 - Proceedings*, 2011.

[124] Haojie Zhuang, Hwee Pink Tan, Alvin Valera, and Zijian Bai. Opportunistic ARQ with bidirectional overhearing for reliable multihop underwater networking. In *OCEANS'10 IEEE Sydney, OCEANSSYD 2010*, 2010.