Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Gutachter: Prof. Jens-Peter Redlich
Prof. Miroslaw Malek
Berlin, den 18. Juni 2008

Network Coding zum Ausgleich von Bitfehlern in drahtlosen Netzwerken

Diplomarbeit vorgelegt von

Name: Ulf Hermann
Matrikelnummer: 185523
E-Mail: uhermann@informatik.hu-berlin.de

Zusammenfassung

Im Vergleich zu kabelgebundener Kommunikation ist die drahtlose Übertragung auf der physikalischem Ebene erheblich fehleranfälliger. Im IEEE 802.11 Standard für Drahtlosnetzwerke sind daher *forward error correction* (FEC) und *automatic repeat request* (ARQ) vorgesehen, um Übertragungsfehler auf ein für höhere Protokollschichten zumutbares Maß zu begrenzen. Beide Techniken sind bei Unicast-Übertragungen effektiv, wenn auch nicht optimal. Allerdings genügen sie nicht den Anforderungen von neuen Routing-Ansätzen wie Opportunistischem Routing, da das Anycast Link Layer Primitiv nicht abgebildet wird.

Ziel der Diplomarbeit ist es, für das Anycast Primitiv ein ARQ-Schema mit inkrementeller Redundanz zu entwerfen. Zu diesem Zweck werden Pakete in Fragmente eingeteilt, die unabhängig von einander auf Fehler geprüft und akzeptiert oder verworfen werden. Auf der anderen Seite werden Pakete in Batches logisch zusammengefasst. Die Fragmente eines Batches bilden die Basis für eine Codierung mittels *random linear network codes* (RLNC). Durch die Codierung wird die übertragene Redundanz optimiert. Übertragungen finden grundsätzlich opportunistische entlang einer durch ein externes Routing-Protokoll, beispielsweise *dynamic source routing* (DSR), vorgegebenen Route statt. Voraussetzung dafür ist ein Verfahren zur Bestimmung der Senderate für die codierten Anycast-Pakete, mit dem der Medienzugriff für die Knoten einer Route gemäß ihrer Nützlichkeit für die Route aufgeteilt werden kann. Um Empfangsfehler auf Datenübertragungs- und Sicherungsschicht in Simulationen darzustellen werden schließlich noch einige Bitfehlermodelle auf der Basis der JiST/SWANS-Simulationsumgebung bereitgestellt.

Inhaltsverzeichnis

1 Einleitung

Ein grundlegendes Charakteristikum von Drahtlosnetzwerken ist die implizite Broadcast-Eigenschaft. Wird ein Paket gesendet, so sind meist mehrere Netzwerk-Knoten in der Lage, es gleichzeitig und ohne zusätzliche Kosten zumindest teilweise zu empfangen. Diese Eigenschaft unterscheidet drahtlose von den meisten modernen drahtgebundenen Netzwerken, wo dezidierte Punkt-zu-Punkt-Verbindungen zwischen je zwei Knoten favorisiert werden. Trotzdem ist in drahtlosen Kanälen effiziente Kommunikation schwieriger zu bewerkstelligen als in drahtgebundenen. Ein Grund dafür ist, dass in standardisierten Protokollen, wie der IEEE 802.11 Protokoll-Familie, die implizite Broadcast-Eigenschaft kaum genutzt wird.

Ein anderer Grund ist die Tatsache, dass im drahtlosen Kanal üblicherweise mehr und stärker variierende Fehler auftreten. Die Kontrolle und der Ausgleich dieser Fehler ist demnach von vitalem Interesse für Kommunikationssysteme, die auf drahtlosen Kanälen arbeiten. Der IEEE 802.11 Standard stellt zwei Mechanismen zur Fehlerkontrolle bereit. Zunächst findet *forward error correction* (FEC) auf der Datenübertragungsschicht statt. Dabei werden redundante Bits in den Datenstrom eingebettet, so dass im Falle einzelner fehlerhafter Bits eine sofortige Korrektur stattfinden kann. Um jedoch annähernd fehlerfreien Empfang nur mittels FEC zu gewährleisten, müssten, dank starker Schwankungen der Fehlerrate, so viele redundante Bits eingebettet werden, dass der Kanal nur äußerst schlecht ausgenutzt würde.

Um dies zu vermeiden, ist im IEEE 802.11 Standard *automatic repeat request* (ARQ) vorgesehen. Jedes Paket wird dabei in einem Stop-And-Wait Verfahren bei erfolgreicher Annahme mit einer *acknowledgement* (ACK) quittiert. Kommt beim Sender kein ACK an, so wird das Paket erneut übertragen. Der Empfänger hingegen wird ein Paket nur bestätigen, wenn er es korrekt empfangen hat. Die redundanten Sendeoperationen stellen also eine weitere Methode der Fehlerkorrektur auf höherer Ebene dar. Für explizite Multicast- und Broadcast-Operationen ist ARQ jedoch nicht verfügbar.

In neueren Forschungsergebnissen wird ein weiteres, verwandtes Link-Layer Primitiv, *Anycast* (vgl. Choudhury und Vaidya 2004), vorgeschlagen, um unter Nutzung der impliziten Broadcast-Eigenschaft des Kanals das Problem der Kanalfehler zu bekämpfen. Eine Sendeoperation soll dabei erfolgreich sein, wenn aus einer Gruppe von Empfängern mindestens einer einen gesendeten Frame fehlerfrei empfängt. Biswas und Morris (2004)[1] beschreiben eine Implementierung dieses Prinzip mit sofortigen orthogonalen

[1] Diese Form der Quellenangabe wird swowohl verwendet, um das Werk zu referenzieren, als auch um die Autoren selbst indirekt zu zitieren.

ACKs. Gibt es also mehrere sich überschneidende Routen zwischen zwei Knoten in einem drahtlosen Netzwerk, so erlaubt eine Anycast-Operation mehrere alternative Pfade der Übertragung. Die Wahrscheinlichkeit, dass auf einem beliebigen dieser Pfade eine vollständige und fehlerlose Übertragung stattfindet ist höher als die Wahrscheinlichkeit, dass dies auf einem vorher festgelegten Pfad geschieht. Ein ARQ-Mechanismus für das Anycast-Primitiv ist, ähnlich wie bei Broadcast und Multicast, entweder nicht trivial oder mit zusätzlichem Overhead verbunden. Davon abgesehen ist das Anycast-Primitiv in der Formulierung auf Frameebene noch nicht optimal. Selbst wenn alle Empfangskandidaten einen Frame fehlerhaft empfangen, so kann es sein, dass durch Kombination der bei verschiedenen Knoten korrekt empfangenen Fragmente der komplette Frame wieder rekonstruierbar ist. Selbst wenn letzteres nicht der Fall ist, so fehlen mit großer Wahrscheinlichkeit nur kleine Teile des Frames. In beiden Fällen ist es nicht notwendig, den kompletten Frame erneut zu senden.

In diesem Zusammenhang stellt sich die Frage, wie ein Paket weitergeleitet wird, das von mehreren Knoten empfangen wurde und somit auf mehreren Pfaden transportiert werden kann. Verallgemeinert ist diese Frage das Ratenkontroll- oder Schedulingproblem. Scheduling in diesem Sinne ist die Aufteilung der Zuständigkeit für die Weiterleitung von Paketen oder auf niedrigerer Ebene die Aufteilung der vorhandenen Sendemöglichkeiten an Knoten im Drahtlosnetzwerk.

Ausgehend von diesen Feststellungen, zeigt diese Arbeit Möglichkeiten auf, mittels inkrementeller Redundanz und einem *hybriden* ARQ-Schema das Anycast-Primitiv effizienter zu gestalten. Zu diesem Zweck, wird im nächsten Abschnitt zunächst die bestehende Literatur zum Thema vorgestellt und ein begrifflicher und thematischer Rahmen geschaffen. Damit zusammenhängend wird in Abschnitt 3 auf das spezielle Problem des Scheduling und bestehende Lösungsansätze dafür eingegangen. Abschnitt 4 stellt Messungen von Bitfehlerraten und -Verteilungen vor, die die Idee der Fragmentierung von Frames rechtfertigen. Abschnitt 5 enthält dann eine Präzisierung des Problems und der Aufgabenstellung, Abschnitt 6 das konzeptionelle Design der Lösung und Abschnitt 7 die architektonischen Details. Die mit Hilfe der Implementation gewonnenen Simulationsergebnisse werden in Abschnitt 8 vorgestellt und abschließende Erkenntnisse über mögliche Weiterentwicklungen in Abschnitt 9.

2 Thematische Einordnung

Dem Thema verwandte Arbeiten können grob in vier Kategorien eingeteilt werden. Zunächst bilden theoretische Modelle des drahtlosen Kanals sowie Messungen darin den allgemeinen Hintergrund. Weiterhin sind bereits bestehende Verfahren zur Feststellung und Korrektur von Fehlern von Bedeutung. Drittens müssen opportunistische Routing-Protokolle betrachtet werden und schließlich spielen neuere Entwicklungen zu Network Coding für drahtlose Kommunikation eine wichtige Rolle. Diese Themen werden in den Abschnitten 2.2 bis 2.5 behandelt. Zu deren Verständnis ist es zunächst nötig, einige Grundbegriffe zu Drahtlosnetzwerken zu definieren, was im folgenden Abschnitt geschieht.

2.1 Begriffe und Konzeptionen für Drahtlosnetzwerke

Zunächst werden in den Abschnitten 2.1.1 und 2.1.2 einige Aspekte des IEEE 802.11 Standards vorgestellt, die sich der Datenübertragungs- und der Sicherungsschicht zuordnen lassen. Maschennetzwerke stellen dabei ein spezielles Gebiet dar, dessen Grundbegriffe im Abschnitt 2.1.3 kurz umrissen werden. Schließlich spielt insbesondere die Analyse der verschiedenen Fehlerfälle beim Datenempfang eine besondere Rolle und damit zusammenhängend werden im Abschnitt 2.1.4 kurz verschiedene Möglichkeiten, den drahtlosen Kanal zu charakterisieren aufgezeigt. Hierbei soll es nur darum gehen, eine Grundlage zum Verständnis der aktuelleren Entwicklungen zu schaffen, die in den nachfolgenden Abschnitten erläutert werden.

2.1.1 Datenübertragungsschicht

Auf der physischen Ebene gilt zunächst für alle besprochenen Systeme, dass Daten in Form von Bits auf elektromagnetische Signale moduliert und so über einen Kanal übertragen werden. Verschiedene Modulationsverfahren manipulieren dabei ein Trägersignal nach Frequenz, Amplitude oder Phase und können dementsprechend unterschiedliche Bitraten modulieren. Ein Signal wird mit einer bestimmten Leistung an der Antenne des Senders angelegt. Der Empfänger erkennt das Signal anhand der Intensität der an seiner Antenne absorbierten elektromagnetischen Wellen. Gesprochen wird von gesendeter und empfangener Signalstärke. Damit ein Empfänger ein Signal wiedererkennt muss es sich gegen das Hintergrundrauschen des Kanals abheben. Inwieweit dies der Fall ist, gibt die *signal to noise ratio* (SNR), als Quotient aus der Stärke des gesendeten Signals einerseits und der Stärke aller Störsignale andererseits an. Überlagern sich mehrere Signale,

so wird von Interferenz gesprochen. (vgl. Goldsmith 2005, Kap. 5)

Voraussetzung für jeden Empfang ist die Erkennung des Trägersignals mittels Trägerprüfung durch den Empfänger. Die Trägerprüfung wird auch genutzt um Mehrfachzugriff auf den Kanal zu regeln. Grundsätzlich prüft ein Sender den Kanal auf bereits anliegende Signale bevor er sendet, um Kollisionen zu vermeiden. Hierbei wird von *carrier sense multiple access* (CSMA) gesprochen. Beschränkt sich das System auf die prophylaktische Prüfung so handelt es sich um *carrier sense multiple access with collision avoidence* (CSMACA). Ein häufiges Problem bei diesem Verfahren sind versteckte Knoten (*hidden nodes*). Versteckte Knoten sind für den Sender nicht erkennbar, beispielsweise wenn sie zu weit entfernt sind, stören aber das Signal beim Empfänger. Eine weitere Methode der gemeinsamen Nutzung eines Kanals ist die probabilistische Spreizung von Signalen, die dadurch für Parteien, die die Codierung nicht kennen als weißes Rauschen erscheinen. Auf diese Weise können mehrere Signale gleichzeitig überlagernd gesendet werden. Dieses Verfahren heißt *code division multiple access* (CDMA) (vgl. Goldsmith 2005, Kap. 13, 14).

Unterschieden wird häufig zwischen Verbindungen mit Sichtkontakt - *line of sight* (LOS) - und solchen ohne - *no line of sight* (NLOS) -, da Objekte, die für Licht undurchdringlich sind meist auch für Radiosignale ein Hindernis darstellen. Jegliche teilnehmenden Geräte, egal ob Sender oder Empfänger on Radiosignalen, werden allgemein als *Knoten* bezeichnet.

2.1.2 Sicherungsschicht

Im Rahmen der Sicherungsschicht sind *Frames* die grundlegenden Dateneinheiten. Frames sind Sequenzen von Bits, die am Stück übertragen werden. Allgemein wird bei solchen Einheiten von *Paketen* gesprochen, durch Verwendung des Wortes *Frame* wird hervorgehoben, dass das Paket der Sicherungsschicht zuzuordnen ist. Der 802.11-Standard definiert ACK-Frames, mit denen der Empfang eines Unicast-Frame gegenüber dem Sender quittiert wird. Jeder Frame wird zudem mit einer durch *cyclic reduncancy check* (CRC) gewonnenen Prüfsumme ausgestattet, die es erlaubt, fehlerhaft übertragene Frames beim Empfänger zu identifizieren. Als Maßnahme zur Verhinderung von Bitfehlern kann eine Fehlerkorrektur angewandt werden, wobei mittels FEC redundante Bits in den Bitstrom eingefügt werden. Die hier verwendete FEC ist ein Faltungscode. Ausgehend von der Trägerprüfung, die die Datenübertragungsschicht bereitstellt, wird zudem ein Mechanismus der Kollisionsvermeidung spezifiziert - die *distributed coordination function* (DCF). Ein potenzieller Sender prüft dabei das Medium auf Aktivität. Liegt kein

Trägersignal vor, so kann er senden. Falls doch, so wartet er eine Weile und versucht es noch einmal (*backoff*). Diese Zeit, das *contention window*, wächst exponenziell mit jedem Versuch. Sie wird zudem mit einem Zufallsfaktor versehen, damit die Wahrscheinlichkeit, dass zwei Knoten gleichzeitig das Medium als frei erkennen und senden, sinkt. Im IEEE 802.11 Standard nehmen die Knoten auf der Sicherungsschicht verschiedene Rollen an. Der *access point* (AP) vermittelt zwischen dem drahtlosen und anderen Netzen. Die *station* (STA) nutzt diesen Service und ist zu jedem Zeitpunkt mit höchstens einem AP assoziiert. Bewegt sich eine STA von einem AP zu einem anderen, so findet Mobilität statt und ein Handoff-Mechanismus regelt die Reassoziation. Zusätzlich gibt es einen Ad-Hoc-Modus der ohne AP auskommt (vgl. 802.11 2007).

2.1.3 Maschennetzwerke

Von Maschennetzwerken wird gesprochen, wenn keine zentralisierte Struktur, wie beispielsweise ein System von APs vorliegt. Jeder Knoten im Netz erfüllt damit alle Funktionen, die in zentralisierten Netzen zwischen STA und AP aufgeteilt werden. Dies führt dazu, dass Daten meist über mehrere Knoten hinweg übertragen werden müssen. Jede (Einzel-)Übertragung wird dabei auch ein *Hop* genannt. Der gesamte Vorgang des Transports von Paketen von einer Quelle zu einem Ziel über mehrere Hops hinweg ist eine End-To-End- oder Multihop-Übertragung. Im Gegensatz zur zentralisierten Struktur eines 802.11-konformen Netzwerks kann hier die Tatsache gezielt ausgenutzt werden, dass Radiowellen mit der Entfernung vom Sender immer schwächer werden. Weit voneinander entfernte Knoten können gleichzeitig senden, ohne sich gegenseitig zu stören. Dieser Effekt wird *spacial diversity* genannt. Auf der anderen Seite gibt es oft auch mehrere potenzielle Empfänger, für die das selbe Datenpaket nützlich ist, oder mehrere Sender, die durch gleichzeitiges Senden das Signal verstärken und somit weiter entfernte Knoten erreichen können. Dieser Effekt wird *Macrodiversity* genannt. Komplementär dazu ist die *temporal diversity*, die genutzt wird, wenn ein Datenpaket mehrmals hinter einander - vollständig oder teilweise, im Klartext oder codiert - gesendet wird. Die Gesamtheit der Pakete, die von einem bestimmten Knoten A zu einem anderen Knoten B in einem Zeitabschnitt übertragen werden, sind der *Datenstrom* oder *Strom* von A nach B. Ein Datenstrom in einem echten oder simulierten Netzwerk kann in bestimmten Fällen als ein *Fluss* in einem graphentheoretischen Netzwerkmodell dargestellt werden, diese beiden Begriffe sind jedoch nicht gleichbedeutend.

2.1.4 Fehlerfälle

Beim Empfang von Signalen im drahtlosen Kanal treten unweigerlich Fehler auf. Diese können auf verschiedenen Ebenen betrachtet werden. Zunächst können die einzelnen Bits untersucht werden. Die *bit error rate* (BER) ist der Quotient aus der Zahl der fehlerhaft angekommenen Bits und der Zahl der insgesamt übertragenen Bits. Eine Stufe höher kann analog die *packet error rate* (PER) betrachtet werden. Ein Paketfehler tritt auf, wenn eventuelle Fehlerkorrektur-Mechanismen auf Bitebene versagt haben und ein Paket fehlerhaft ausgeliefert oder verworfen wird. Wiederum eine Ebene höher kann der Erholungsmechanismus für Paketfehler betrachtet werden. Wird davon ausgegangen, dass die Reaktion auf einen Paketfehler eine erneute Sendeoperation ist, so können die *extpected transmissions* (ETX) als Metrik genutzt werden. Die ETX-Metrik gibt die erwartete Zahl von Sendeoperationen an, die für ein Paket nötig sind, um eine Einzelübertragung korrekt durchzuführen. Die ETX-Metrik kann auch über mehrere Übertragungen hinweg aggregiert werden und gibt dann die erwartete Gesamtzahl der Sendeoperationen an, die nötig sind, um ein Paket auf einer bestimmten Route zu übertragen.

Die Ursachen der Übertragungsfehler werden häufig in drei Kategorien eingeteilt. Erstens spielt die Entfernung zwischen Sender und Empfänger eine entscheidende Rolle für die Häufigkeit der Fehler. Die Signalstärke nimmt mit der Entfernung exponenziell ab, dieser Effekt heist Pfadverlust. Zweitens kann durch Bewegungen von Sender und Empfänger gegeneinander durch den Dopplereffekt eine Signalverschiebung stattfinden. Drittens schließlich gibt es üblicherweise eine ganze Reihe von Umwelteinflüssen, die sich als Zufallsprozesse darstellen lassen - das *fading*. Insbesondere *multipath* Fading ist in diesem Zusammenhang von Bedeutung. Es entsteht, indem das selbe Signal durch Reflektion auf verschiedenen Wegen, und damit phasenverschoben beim Empfänger ankommt. Je nach Umfang der Phasenverschiebung kann eine Dämpfung oder eine Verstärkung des Signals stattfinden.

2.2 Kanalmodelle und -messungen

Aguayo u. a. (2004) untersuchen Link-Level Messungen in einem IEEE 802.11b Netzwerk im Außenbereich. Sie stellen fest, dass Paketfehler bei drahtloser Kommunikation häufig sind. Die meisten der untersuchten Verbindungen waren deshalb von mittlerer Qualität, was mit dem Effekt des Multipath-Fading begründet wird. Bestätigt wird diese Feststellung durch Simulationen mit einem Kanal-Emulator. De Couto u. a. (2003) zeigen anhand ihrer Ausführungen zur ETX-Metrik, dass eben diese Verbindungen von mitt-

lerer Qualität entscheidende Bedeutung für das Zustandekommen von hohem Durchsatz in ungeplanten Maschennetzwerken tragen.

Dass die IEEE 802.11-Sicherungsschicht mit solchen Verbindungen nicht optimal umgehen kann, zeigen unter anderem Rayanchu u. a. (2008). Sie stellen fest, dass die Reaktion der Sicherungsschicht auf ein inkorrekt empfangenes Paket - exponentielles Backoff und erneutes Senden - im Falle von Verlust aufgrund zu schwacher Signale sehr verschwenderisch ist. Ist das empfangene Signal zu schwach, so macht es keinen Sinn, vor dem nächsten Sendeversuch längere Zeit zu warten. Im Gegenteil, um über den ohnehin schlechten Kanal eine möglichst hohe Datenrate zu übertragen, sollte dieser möglichst intensiv genutzt werden. Anders ist dies natürlich bei kollisionsinduziertem Paketverlust. Senden mehrere Stationen gleichzeitig, so ist der Backoff eine sinnvolle Maßnahme, um Kollisionen zu vermeiden. Somit ist es vorteilhaft, zwischen Kollisionen und zu schwachem Signal als Grund der Empfangsfehler zu unterscheiden.

Theoretische Modelle legen nahe, dass es einen engen Zusammenhang zwischen momentaner SNR und Paketfehlerrate gibt (vgl. Goldsmith 2005, Kap.6). Der Übergangsbereich von keinem zu vollständigem Paketverlust sollte für die im IEEE 802.11 Standard aufgeführten Modulationstechniken relativ klein sein. Aguayo u. a. (2004) zitieren beispielsweise die Spezifikation eines Radiochips, der von etwa 3 bis 4 dB ausgeht. Tatsächliche Messungen der selben Autoren in innerstädtischen Maschennetzwerken bestätigen dies jedoch nicht. Andererseits stimmen weitgehend interferenzlose Langstreckenverbindungen in ländlichen Gegenden nach Chebrolu u. a. (2006) sehr gut mit diesem Modell überein. Hieraus nährt sich die Vermutung, dass Interferenz den Zusammenhang zwischen SNR und PER aufbricht. Tatsächlich zeigen Souryal u. a. (2006) Messergebnisse, in denen der SNR-PER Graph unter Einfluss von Interferenz deutlich flacher ausfällt als ohne. So weisen Verbindungen mit einer SNR von 30 dB beispielsweise ohne Interferenz eine sehr niedrige, unter moderater Interferenz eine Paketfehlerrate von 0 bis 20 Prozent und unter hoher Interferenz von 10 bis 50 Prozent auf. Daraus ergibt sich, dass Interferenz ein Hauptgrund des Paketverlusts in drahtlosen Netzwerken nach IEEE 802.11 ist. Auch verkürzte Frames sind ein weit verbreitetes Problem auf IEEE 802.11 konformer Hardware. Obwohl über deren Ursachen wenig Details bekannt sind, ist eine plausible Annahme, dass auch dieser Effekt zumindest teilweise durch interferenzbedingte Kollisionen hervorgerufen wird. Insbesondere der *capture*-Effekt, bei dem das Signal eines schon teilweise empfangenen Frames vom einsetzenden Signal eines weiteren Knotens komplett verdeckt wird, kann hier eine Rolle spielen.

Deutlich werden diese Zusammenhänge auch bei Lee u. a. (2007). Diese betrachten die

Durchsatzraten verschiedener interferierender Verbindungen in einem 802.11a-Netzwerk. Sie senden dabei bei einer Bitrate von 6Mbps und stellen die Knoten so auf, dass verschiedene Kombinationen zweier Sender ihre Trägersignale empfangen können beziehungsweise deren Signale an zwei Empfängern interferieren. Dabei zeigt sich, dass die Durchsatzraten, je nach Interaktion der Verbindungen, der Verwendung von Broadcast- oder Unicast-Modus, sowie der verwendeten Paketrate stark unterschiedlich ausfallen. Insbesondere zeigen sie beispielsweise, dass bei voller Auslastung des Kanals durch zwei Broadcast-Sender, die ihr jeweiliges Trägersignal nicht erkennen können, aber an beiden Empfängern interferieren, praktisch kein Durchsatz erzielt werden kann. Wird hingegen die Paketrate so gedrosselt, dass bei beiden die Hälfte der Kanalkapazität genutzt wird, so können die Empfänger je etwa 40% der für sie gedachten Pakete empfangen. Weiterhin stellen sie fest, dass unter bestimmten Bedingungen eine stark unfaire Verteilung der Durchsatzraten zwischen den beiden Verbindungen stattfindet.

Auf Symbolebene charakterisiert Goldsmith nach Shannon einen zeitdiskreten Kanal mit additivem weißen Rauschen (*additive white gaussian noise* (AWGN)) (vgl. Goldsmith 2005, Kap. 4). Dieses theoretische Modell bietet eine einfache Handhabe zur Voraussage von Erfolgswahrscheinlichkeiten für einzelne Hops. Jedes Bit ist statistisch von der selben Fehlerwahrscheinlichkeit betroffen. Damit sind auch die Bitfehler im AWGN-Kanal über die Länge der Pakete gleichverteilt. Der AWGN-Kanal ist jedoch ein für viele Zwecke zu stark vereinfachtes Modell terrestrischer Systeme, da Effekte wie Multipath-Fading, Hindernisse, Mobilität und Interferenz unberücksichtigt bleiben. Insbesondere Fading spielt, wie beispielsweise von Aguayo u. a. (2004) festgestellt, in echten Umgebungen häufig eine große Rolle. Für das Fading gibt es zwei allgemein akzeptierte Modelle. Rayleigh Fading für Umgebungen ohne Sichtkontakt und Rician Fading für solche mit Sichtkontakt. Fading kann den Kanal mit einem *Gedächtnis* versehen, so dass die Gleichverteilungsannahme nicht mehr gilt. Hierbei ist die Kohärenzzeit des Kanals ein wichtiger Parameter. Die Kohärenzzeit ist das Zeitintervall, nach dem die Effekte des zu Grunde liegenden Fadingprozesses wieder zu den vorher beobachtbaren unkorreliert sind. Ist die Kohärenzzeit kleiner als die Symboldauer, so gleichen sich die Variationen des Fadings auf Bitebene aus und die Gleichverteilungsannahme bleibt sinnvoll. Ist andererseits die Kohärenzzeit größer als die Dauer eines Frames, so manifestiert sich das Fading auf Bitebene und trägt zur lokalen Häufung von Bitfehlern bei (vlg. Tse und Viswanath 2005, Kap. 2).

Verschiedene Arbeiten zeigen, dass Bitfehler bei drahtloser Kommunikation lokal gehäuft auftreten, also nicht gleichverteilt über einen längeren Datenstrom (vgl. Tanen-

baum 2002, Kap. 3). Willig u. a. (2002) untersuchen das Verhalten von IEEE 802.11b Netzwerken in einer industriell geprägten Umgebung. Sie zeigen, dass sowohl Paket-, als auch Bitfehler lokal gehäuft auftreten. Weiterhin argumentieren sie, dass ein Markov-Modell mit zwei Zuständen nach Gilbert und Elliot (vgl. Gilbert 1960) den Resultaten besser entspricht, als ein Modell, das von gleichverteilt auftretenden Bitfehlern ausgeht. Dubois-Ferriere u. a. (2005) beobachten ebenfalls lokal gehäufte Bitfehlermuster in Verbindungen mit hoher Bitfehlerrate im Innenbereich. Sie arbeiten dabei mit Schmalband-Sensorknoten. Miu u. a. (2005) schließlich weisen auf Häufungen von Bitfehlern im Rahmen von bis zu 100 Bits in IEEE 802.11a Netzwerken hin. Sie beobachten ebenfalls, dass Paketverlust bei verschiedenen Empfängern unabhängig auftritt. Allgemein lässt sich feststellen, dass auf theoretischer Ebene die vorgestellten Fading-Modelle die Verteilung der Bitfehler nicht vollständig erklären können.

2.3 Fehlerkontrolle auf der Sicherungsschicht

Die beiden Fehlerkontroll-Mechanismen im 802.11-Standard sind komplementär. FEC kann einzelne Bitfehler durch unmittelbare Redundanz ausgleichen, ARQ springt ein, wenn das fehlschlägt. IEEE 802.11g und 802.11a nutzen für die FEC einen Faltungscode der festen Länge 7 (vgl. 802.11 2007). Beim Entwurf dieses Faltungscodes wurde von einem AWGN-Kanal mit gleichverteilten Fehlern ausgegangen. Sobald Fading auftritt oder aus anderen Gründen gehäufte Fehler auftreten, fällt die Leistung des Codes bei der Korrektur der Fehler deutlich ab (vgl. Goldsmith 2005, Kap. 6). Riemann und Winstein (2005) beschreiben deshalb Experimente mit einem über der Link-Layer angesiedelten *Reed-Solomon* (RS) Code, die deutliche Steigerungen der maximalen Reichweite von IEEE 802.11-Netzwerken durch Reparatur der sonst auftretenden Fehler zeigen. RS-Codes sind speziell zur Korrektur lokal gehäufter Fehler gut geeignet. Daher wurde auch im IEEE 802.16e *mobile WiMAX* Standard ein äußerer RS- zusätzlich zu einem inneren Faltungscode übernommen (vgl. 802.16e 2005).

ARQ korrigiert Fehler in übertragenen Frames, nachdem sie beim Empfänger aufgetreten sind, durch erneutes Senden des selben Frames. Um die Rückmeldung zum Empfangsstatus des Frames zu übermitteln, ist ein Signalisierungskanal vom Empfänger zum Sender nötig. Im IEEE 802.11 Standard wird ein einfaches Stop-and-Wait Protokoll spezifiziert (vgl. 802.11 2007). Hierbei muss jeder korrekt empfangene Frame sofort vom Empfänger bestätigt werden. Bleibt die Bestätigung aus, so wird der Frame erneut gesendet. Im IEEE 802.11 Standard wird ein ausbleibendes ACK grundsätzlich als Kollision interpretiert. Der Sender reagiert, um weitere Kollisionen zu vermeiden, mit einer Ver-

doppelung des Contention Windows, also der Zeit, die er wartet, bevor er erneut versucht zu senden. Zudem versucht der Sender nicht ewig, einen Frame durch erneutes Senden zu korrigieren. Üblicherweise wird nach 7 bis 11 redundanten Übertragungsversuchen aufgegeben. Fortgeschrittenere ARQ-Protokolle nutzen einen Sliding-Window-Ansatz, mit dem, abhängig von der Umgebung, deutliche Leistungsgewinne über Stop-and-Wait erzielt werden können. Sundararajan u. a. (2008) skizzieren beispielsweise ein solches Modell.

hybrid ARQ (HARQ) kombiniert die Mechanismen der Kanalcodierung und des ARQ. Hierbei wird das Prinzip der inkrementellen Redundanz verwendet. Nur bei schlechten Kanalbedingungen ist Redundanz nötig. HARQ nutzt dabei sofortige Rückmeldungen des Empfängers um zu bestimmen, ob die bisher übertragenen redundanten Daten ausreichen oder ob weitere Redundanz übertragen werden muss. Damit kann auf Schwankungen der Kanalqualität unmittelbar reagiert werden. Subramanian u. a. (2007) verwenden HARQ auf verschiedenen Schichten für point-to-point Verbindungen und erzielen dabei vielversprechende Ergebnisse. Neuere Ansätze passen HARQ zur Anwendung in Systemen mit mehreren Knoten an. Ein Beispiel dafür ist *layered packet coded cooperative system* (LPCCS) (vgl. Levorato u. a. 2006). Mit LPCCS wird ein Frame in mehreren FEC-Phasen gesendet. Jede dieser Phasen wird durch eine negative oder positive Rückmeldung des Empfängers beendet. Nachbarknoten, die einen Frame zufällig mithören, können sich anhand von SNR-Messungen entscheiden, bei der Übertragung mitzuhelfen. Allerdings sind für LPCCS Systeme mit mehreren Antennen nötig, so dass mittels CDMA orthogonaler Kanalzugriff möglich wird. Dieser Umstand führt dazu, dass LPCCS nur auf deutlich aufwändigerer Hardware anwendbar ist, als momentan kommerziell verfügbar ist.

Eine weitere praktische Herangehensweise an die HARQ-Idee ist Paketkombination. Bei einfacher Paketkombination, wie Dubois-Ferriere u. a. (2005) sie vorschlagen, heben Empfänger fehlerhaft empfange Frames auf, anstatt sie zu verwerfen. Anschließend wird durch Kombination mit weiteren fehlerhaften Kopien des selben Frames versucht, das korrekte Paket zu rekonstruieren. Kombiniert werden dabei jeweils Fragmente verschiedener empfangener Frames ohne spezielle Codierung. Falls in jedem Frame nur wenige Fragmente defekt sind, führt dies mit hoher Wahrscheinlichkeit schnell zum Erfolg. Jamieson und Balakrishnan (2007) stellen mit *Partial Packet Recovery* ein ähnliches System vor. Dabei werden die Konfidenzwerte zu den einzelnen Bits auf der Datenübertragungsschicht genutzt, um gezielt nur diejenigen Teile eines inkorrekt empfangenen Frames neu anzufordern, die mit großer Wahrscheinlichkeit auch wirklich defekt sind.

Auf ähnliche Art erzielen Miu u. a. (2005) mit ihrem *multi radio diversity* (MRD) System beeindruckende Leistungssteigerungen in einer realen Testumgebung. MRD verlässt sich dabei auf die Effekte der Spacial- und Macrodiversity in 802.11-Infrastruktur-Netzwerken. In typischen Anordnungen befindet sich ein STA-Knoten in der Nähe von mehreren AP-Knoten. Trotzdem schreibt der IEEE 802.11-Standard vor, dass jeder STA-Knoten zu jedem Zeitpunkt nur mit einem AP-Knoten assoziiert ist. Um die brach liegende Empfangskapazität der anderen AP-Knoten zu nutzen und dadurch von der Diversity zu profitieren, wird diese Festlegung durch MRD aufgehoben. Hier kooperieren die APs beim Empfang von Frames über einen drahtgebundenen Backbone miteinander. Die Wahrscheinlichkeit, dass ein beliebiger AP den gesendeten Frame korrekt empfängt ist wiederum größer als die Wahrscheinlichkeit, dass ein festgelegter AP dies schafft. Zusätzlich wird auch hier Paketkombination angewandt, um aus mehreren unterschiedlich defekten Kopien des selben Frames einen korrekten Frame zu rekonstruieren.

Letztendlich ist Paketkombination jedoch immer eine Form der Codierung mittels Wiederholung und effizientere Codes sind bereits bekannt (vgl. Tse und Viswanath 2005, Kap. 3).

2.4 Multi-User Diversity und Opportunistisches Routing

Multi-User Diversity und Opportunistisches Routing in Maschennetzwerken ist Thema einer Reihe von Publikationen. Der erste Versuch, ein entsprechendes Protokoll zu entwickeln wird von Larsson (2001) vorgestellt. Das Ergebnis, *selection diversity forwarding* (SDF), baut auf der zentralen Idee auf, statt einem einzelnen Knoten eine Menge von Kandidaten für den nächsten Hop zur Übertragung eines Paketes in Betracht zu ziehen. Sobald ein Frame mittels Broadcast von dem momentanen Knoten abgegeben wurde, müssen diejenigen Kandidaten, die den Frame empfangen haben, sich darüber einig werden, welcher ihn weitersenden soll. Zu diesem Zweck wird ein relativ teurer 4-Wege-Handshake implementiert (Daten, ACK, FO, FOA). Andere Autoren, beispielsweise Valenti und Correal (2003) oder Biswas und Morris (2004, 2005), stellen Verbesserungen des SDF-Protokolls vor, die mit weniger aufwändigen 2-Wege-Handshakes aus Daten und in festen Zeitrahmen zu sendenden (*slotted*) ACKs auskommen.

Das von Biswas und Morris (2004, 2005) beschriebene *extreme opportunistic routing* (ExOR) Protokoll für Maschenneztwerke arbeitet somit ähnlich wie SDF. Um Zwischenknoten auf dem Weg zum Zielknoten zu überspringen und so Hops einzusparen bestimmt auch ExOR bei jeder Sendeoperation eine Gruppe von Empfangskandidaten für den nächsten Hop. Die Auswahl des Knotens, der weiter für den Frame verantwortlich

ist und die Koordination der Kandidaten stützt sich auf ein slotted ACK. Kandidaten bestimmen den Zeitpunkt für ihr jeweiliges ACK anhand ihrer Priorität, die durch die Position in der Kandidatenliste ausgedrückt wird. Von allen Knoten, die den Frame erfolgreich empfangen haben, sendet der mit der höchsten Priorität das erste ACK und ist damit auch verantwortlich für die Weitervermittlung. Leider gibt es keine Garantie dafür, dass alle Kandidaten dieses ACK empfangen und somit kann es zu redundanten Übertragungen und Duplikaten kommen (vgl. Zubow u. a. 2006). Es existieren mehrere Ansätze zur Lösung dieses Problems. Nennenswert sind insbesondere das von Biswas und Morris (2005) selbst vorgestellte *batch processing*, robustere (beispielsweise passiv mit anderen Frames mitgeschickte) ACKs (vgl. Zhong und Nelakuditi 2007) und komprimierte slotted ACKs (vgl. Zubow u. a. 2006).

Larsson und Johansson (2005) beschreiben eine verbesserte Version des SDF-Protokolls, names *multiuser diversity forwarding* (MDF). Die grundlegende Idee stellt sich wie folgt dar: Der Sender sendet ein Testpaket per Multicast an eine Menge von Kandidaten, die jeweils anhand dieses Pakets die Qualität des Kanals auswerten. Die Information über die Kanalqualität wird dann an den ursprünglichen Sender zurück geschickt. Dieser bestimmt anhand der aggregierten Kanalmessungen den momentan wichtigsten Datenstrom, den besten nächsten Hop für ein momentan zu sendendes Paket und die für diesen Hop günstigste Senderate. Dieser nächste Hop wird dann im Voraus für das Paket ausgewählt. Damit geht MDF implizit davon aus, dass die Kohärenzzeit des Kanals länger ist als die Zeit, die das Protokoll zur Erstellung und Aggregation der Kanalmessungen braucht.

Auf die erwähnten Protokolle aufbauend betonen Choudhury und Vaidya (2004) die Bedeutung der Protokollschichtung. Sie führen deshalb *Anycast* als neues Primitiv der Sicherungsschicht ein, das dann als Baustein für opportunistische Protokolle genutzt werden kann. Damit wird die Sicherungs- und Vermittlungsschicht besser getrennt.

2.5 Network Coding

Network Coding kann als Methode zur Optimierung des Durchsatzes in Netzwerken angesehen werden. Zunächst soll hier Network Coding im Allgemeinen sowie die theoretische Herleitung dieser Sichtweise charakterisiert werden. Anschließend werden verschiedene Möglichkeiten aufgezeigt, wie Network Coding im Umfeld der drahtlosen Maschennetzwerke nutzbar gemacht werden kann.

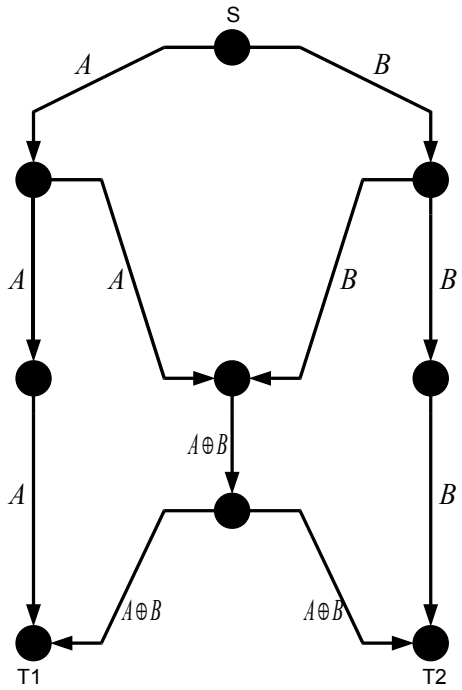


Abbildung 1 — Einfaches XOR-Coding-Schema

2.5.1 Graphentheoretische Betrachtung

Traditionell werden Netzwerke als ungerichtete gewichtete Graphen dargestellt, über deren Kanten Unicast-Übertragungen stattfinden. Pro Zeiteinheit können der Kapazität der Kanten entsprechend viele Übertragungen getätigt werden. Die maximale (Multipath-)Übertragungsrate zwischen zwei Knoten wird in diesem Fall durch den minimalen Schnitt zwischen diesen begrenzt, wie Ford und Fulkerson (1956) und Elias u. a. (1956) festgestellt haben.

Für Multicast-Übertragungen, bei denen ein Quellknoten an mehrere Zielknoten gleichzeitig sendet, ist diese Übertragungsrate nicht unbedingt erreichbar. Ahlswede u. a. (2000) zeigen, dass das wesentliche Hindernis in diesem Fall das traditionelle Verständnis von Routing als einfache Kopier- und Relay-Operation an den Knoten im Netzwerk ist. Dies wird an einem einfachen und vielzitierten (vgl. z. B. Ahlswede u. a. 2000) Beispiel deutlich. In Abbildung 1 haben alle Kanten Einheitskapazität. Der minimale Schnitt zwischen Quellknoten S und jedem der Zielknoten T1 und T2 ist offensichtlich 2. Der maximale mittels einfachem Routing erreichbare Durchsatz jedoch insgesamt 3. Eine einfache XOR-Codierung, bei der an den mittleren Knoten je eine Kombination der

empfangenen Pakete gesendet wird, kann jedoch für beide Empfänger einen Fluss von 2 erreichen. Daher schlagen Ahlswede u. a. (2000) als Gegenentwurf und Erweiterung zum traditionellen Routing allgemein *network coding* als ein Verfahren vor, bei dem jeder Knoten im Netzwerk verschiedene Pakete nach einem Codierungsschema miteinander kombiniert. Auf diese Art kann auch in Multicast-Szenarien eine dem minimalen Schnitt zwischen der Quelle und jedem Ziel entsprechende Übertragungsrate erreicht werden.

Li u. a. (2003) zeigen, dass lineare Transformationen, zum Beispiel in einem endlichen Feld, als Codierungsschema ausreichend sind, um diese Übertragungsrate zu erreichen. Ein Quellknoten S generiert dabei einen d-dimensionalen Vektorraum Ω und jedem Knoten $T \neq S$ wird ein Subraum $v(T)$ von Ω zugewiesen. Jede Kante wird in gerichtete *Kanäle* von je Einheitskapazität eingeteilt. Ein zum Fluss beitragender Kanal XY von Knoten X nach Knoten Y erhält nun einen Spaltenvektor $v(XY) \in v(X)$. Außerdem gilt, dass $v(Y)$ für einen Knoten Y von den Vektoren aller eingehenden Kanäle $v(XY)$ aufgespannt wird. S erzeugt die zu übertragenden Daten als einen d-dimensionalen (Zeilen-)*Informationsvektor*. Auf jeder Kante XY wird letztendlich das Produkt aus Informationsvektor und $v(XY)$ übertragen. Bewiesen wird nun, dass es Vektoren $v(XY)$ gibt, so dass $dim(v(T)) = maxflow(T)$ für alle Knoten $T \neq S$. Dies führt dazu, dass jeder Knoten T einen $v(T)$-dimensionalen Informationsvektor vollständig decodieren kann. Ein Informationsvektor, der nach den minimalen Schnitt einer Quelle zu allen Empfängern einer Multicast-Übertragung dimensioniert ist, kann also den maximalen Fluss nach Ford und Fulkerson (1956) realisieren.

Koetter und Médard (2003) erweitern dieses Ergebnis und formulieren das algebraische Problem der Netzwerkkommunikation. Es wird weiterhin ein gerichteter, gewichteter (Multi-)Graph $G = (V, E)$ mit Einheitskapazitäten für alle, potenziell parallelen Kanten $e \in E$ betrachtet. Jede Kante $e = (v, v')$ hat einen Ursprung $tail(e) = v$ und ein Ziel $head(e) = v'$. An jedem Knoten v sind dabei Zufallsprozesse $\mathscr{X}(v) = \{X(v,1), X(v,2), X(v,3)...X(v,\mu(v))\}$ beobachtbar. Diese entsprechen der Quelle eines Flusses. Die Entropierate jedes Prozesses entspricht der Kapazität der Kanten. Quellen, die eine höhere Rate erzeugen, werden als mehrere parallele Prozesse modelliert. Eine Teilmenge $\mathscr{X}(v, v') \subseteq \mathscr{X}(v)$ davon soll an einem anderen Knoten v' empfangen werden. Dieser Umstand beschreibt eine *Verbindung* als Tripel $(v, v', \mathscr{X}(v, v'))$. Zusätzlich gibt es an jedem Knoten eine Ausgabe, wiederum als Zufallsprozesse $\mathscr{Z}(v) = \{Z(v,1), Z(v,2), ...Z(v,\nu(v))\}$. Eine Verbindung $(v, v', \mathscr{X}(v, v'))$ heißt erfolgreich, wenn $\mathscr{X}(v, v') \subseteq \mathscr{Z}(v')$. Über jede Kante e wird ein Zufallsprozess $Y(e)$ übertragen. Dabei

gilt:

$$Y(e) = \sum_{l=1}^{\mu(v)} \alpha_{e,l} X(v,l) + \sum_{e':head(e')=tail(e)} \beta_{e',e} Y(e')$$

sowie:

$$Z(v,j) = \sum_{e':head(e')=v} \epsilon_{e',j} Y(e')$$

$\alpha_{e,l}$, $\beta_{e',e}$ und $\epsilon_{e',j}$ sind Elemente eines endlichen Feldes $\mathbb{F}_{2^m}$, wobei m die Länge der Elemente von $X(v,l)$, $Y(e)$ und $Z(v,l)$ ist. Das Problem der Netzwerkkommunikation in seiner einfachsten Form ist nun, für einen Graphen G und eine Menge Verbindungen $\mathscr{C}$ eine Belegung der $\alpha_{e,l}$, $\beta_{e',e}$ und $\epsilon_{e',j}$ zu finden, so dass alle Verbindungen in $\mathscr{C}$ erfolgreich sind. Es ist leicht zu sehen, dass es sich hier um eine Verallgemeinerung des von Li u. a. (2003) vorgestellten Modells auf beliebig viele Multicast-Flüsse handelt. Wird das Modell auf nur einen Multicast-Fluss eingeschränkt, also

$$\exists\, x \in V : \forall\, (v, v', \mathscr{X}(v,v')) \in \mathscr{C} : v = x,$$

so gelten die Ergebnisse von Li u. a. (2003) weiter. Koetter und Médard (2003) zeigen unter anderem diese noch einmal unabhängig.

Ho u. a. (2006) führen diesen Gedanken weiter aus und zeigen, dass es bei großem m reicht, $\alpha_{e,l}$ und $\beta_{e',e}$ zufällig und gleichverteilt aus $\mathbb{F}_{2^m}$ zu wählen, um bei lösbaren Problemen $(G, \mathscr{C})$ mit hoher Wahrscheinlichkeit eine Lösung zu erreichen. $\epsilon_{e',j}$ sind dann durch $\alpha_{e,l}$ und $\beta_{e',e}$ determiniert. Die Wahrscheinlichkeit, eine Lösung zu erreichen beträgt für eine einzelne Multicast-Verbindung mindestens $(1-d/q)^\eta$, wobei d die Zahl der Empfänger, η die Zahl der verwendeten Kanten und $q = 2^m$ ist. Sie stellen fest, dass die Wahrscheinlichkeit einer fehlerhaften Belegung mit steigendem m exponentiell abnimmt. Es bleibt jedoch anzumerken, dass die Wahrscheinlichkeit andererseits mit der Anzahl der verwendbaren Kanten exponenziell zunimmt. In einem größeren Graphen mit mehreren Flüssen, muss q demnach entsprechend groß gewählt werden. Bei Abwesenheit von globalem Wissen der Knoten müssten zudem jedem potenziellen Zielknoten eines Flusses die Koeffizienten aller am Fluss beteiligten Knoten und Kanten in einem Initialisierungsverfahren mitgeteilt werden. Als Vorteil der statischen Belegung der Koeffizienten kann jedoch festgehalten werden, dass außer dieser Initialisierung kein Overhead nötig ist.

Ist im laufenden Betrieb ein gewisser Overhead akzeptabel, so können die Koeffizienten für jede Sendeoperation einzeln neu bestimmt werden. Lun u. a. (2006) wählen

dieses Verfahren. Zunächst wird hier das Konzept des Pakets als Zusammenfassung einer Anzahl gleich codierter Bits eingeführt. Jedes Paket muss nun mit seinem globalen Codierungsvektor γ, einer kondensierten Form der $\alpha_{e,l}$ und $\beta_{e',e}$ versehen werden, damit der Zielknoten die nötigen Decodierungskoeffizienten berechnen kann. Der damit verbundene Overhead sinkt anteilsmäßig mit steigender Länge der Pakete. Zusätzlich wird ein Hilfsvektor $\beta = (\beta_1, \beta_2 ... \beta_n)$ definiert, mit dem ein Paket x im Netzwerk als Linearkombination der ursprünglich vom Quellknoten gesendeten Pakete $v_1, v_2, ... v_n$ ausgedrückt werden kann:

$$x = \sum_{l=1}^{n} \beta_l v_l$$

In diesem Zusammenhang führen Lun u. a. (2006) als eine zentrales Konzept die *Innovativität* eines Paketes ein. Ein an einem Knoten empfangenes Paket heißt innovativ, wenn sein Hilfsvektor nicht in dem Vektorsubraum liegt, der von den Hilfsvektoren der bisher dort empfangenen Pakete aufgespannt wird. Ein innovatives Paket ist also linear unabhängig von anderen Paketen des selben Flusses am selben Knoten. Lun u. a. (2006) ziehen nun Deb u. a. (2006) heran, um zu zeigen, dass mit gleichverteilt generierten Koeffizienten codierte Pakete mit hoher Wahrscheinlichkeit innovativ sind. Sind V_1 und V_2 zwei Mengen von Vektoren aus $\mathbb{F}_q^n$ und β eine Linearkombination der Vektoren aus V_1 mit zufällig gleichverteilten Koeffizienten aus $\mathbb{F}_q$. Dann gilt[2]:

$$Pr(\beta \notin span(V_2) | span(V_1) \nsubseteq span(V_2)) \geq 1 - 1/q$$

Dieses Ergebnis ist von entscheidender Bedeutung. Wird an einem Knoten aus vorhandenen Paketen unter zufälliger Koeffizientenwahl ein codiertes Paket erzeugt und gesendet, dann ist dieses für einen Empfänger, unabhängig vom Umfang der jeweiligen vorhandenen Subräume, mit hoher Wahrscheinlichkeit innovativ. Einzige Bedingung ist, dass der Subraum des Empfängers kleiner ist als der des Senders, dass es also überhaupt noch Koeffizienten gibt, mit denen der Sender ein innovatives Paket erzeugen kann. Lun u. a. (2006) konstruieren anhand dieses Ergebnisses ein Codierungs-Schema, das ein Netzwerkkommunikationsproblem mit der Einschränkung auf einen Unicast- oder Multicast-Fluss und unter spezieller Berücksichtigung verlustbehafteter Kanten annähernd optimal löst.

Die von Ahlswede u. a. (2000), Li u. a. (2003), Koetter und Médard (2003), Ho u. a. (2006) sowie Lun u. a. (2006) vorgestellten Multicast-Szenarien spielen im Folgenden kei-

[2]Die Formulierung in Lun u. a. (2006) lässt den Fall $span(V_1) = span(V_2)$ in der Bedingung zu. Das wurde unter Berücksichtigung der ursprünglichen Quelle (Deb u. a. 2006) hier korrigiert.

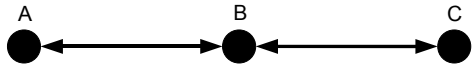


Abbildung 2 — Gegengerichtete Flüsse

ne direkte Rolle. Li und Li (2004) zeigen, dass Coding zwischen verschiedenen Unicast-Flüssen in bestimmten Fällen ebenfalls günstig ist. Dieser Aspekt wird im nächsten Absatz kurz besprochen, bleibt jedoch auch hauptsächlich außen vor. Der Fokus liegt auf Coding innerhalb eines Unicast-Flusses. Allerdings ist ein Unicast-Multipath-Fluss ein Sonderfall eines Multicast-Flusses. Insofern sind die genannten Ergebnisse direkt übertragbar. Zwar kann in diesem Fall in einem gewichteten, gerichteten Graphen der maximale Fluss auch ohne Network Coding erreicht werden, jedoch ist nun gezeigt, dass auch mit Network Coding - und insbesondere mit zufälligen Linearkombinationen - nur wenig Kapazität verloren geht. Die folgenden Arbeiten zeigen verschiedene praktische Vorteile des Network Coding im Unicast-Fall und damit auch Schwächen der Modellierung von Drahtlosnetzwerken als gewichtete Graphen.

2.5.2 Coding zwischen verschiedenen Strömen

Eine offensichtliche Anwendung des Network Coding ist die Kombination verschiedener Unicast-Datenströme unter Nutzung der impliziten Broadcast-Eigenschaft des Mediums. Im Gegensatz zum Modell des gewichteten Graphen gibt es in Drahtlosnetzwerken keine dedizierten Kanten zwischen Knoten. Was ein Knoten sendet, wird unweigerlich über alle Kanten übertragen, die mit dem Knoten verbunden sind. Auf diesen Kanten kann parallel nichts anderes übertragen werden. Dies kann mittels Network Coding zur Durchsatzsteigerung genutzt werden. Der Gedanke wird in Abbildung 2 deutlich. Es soll hier ein Paket a von A nach C und ein Paket c von C nach A gesendet werden. Unter Nutzung eines klassischen Routingschemas wären dazu vier Sendeoperationen nötig. A und C senden je einmal, B zweimal. Da jedes von B gesendete Paket jedoch sowohl bei A als auch bei B ankommt, kann hier eine Sendeoperation eingespart werden. B kann statt a und c einzeln die Kombination $a \oplus c$ senden. Da A und C jeweils das eigene Paket schon kennen, können sie wieder mittels einer XOR-Operation das je andere Paket aus der Kombination gewinnen. Dieses Prinzip nutzen Katti u. a. (2006) mit ihrem *COPE*-Protokoll. Als zusätzlicher Gewinn stellt sich dabei heraus, dass die Unzulänglichkeiten

der IEEE 802.11 MAC-Schicht bezüglich der Fairness (vgl. Abschnitt 3.1), unter denen insbesondere Multihop-Übertragungen leiden, teilweise umgangen werden können. Insgesamt kann dieses Verfahren jedoch nur angewandt werden wenn einerseits gegengerichtete Datenströme vorliegen, die sich an einem Knoten schneiden und andererseits Pakete an diesem Knoten so lange aufgehoben werden, dass eine Codierungsmöglichkeit entsteht. Diese Bedingungen sind sehr restriktiv, daher wird die im nächsten Abschnitt vorgestellte Codierung innerhalb eines Stroms präferiert.

2.5.3 Coding innerhalb des selben Stroms

Verschiedene weitere Werke zeigen einen weiteren Gewinn der mit Network Coding in drahtlosen Netzwerken erzielt werden kann: Der größte Teil der Statusrückmeldungen, die beispielsweise in Form von ACKs von Empfängern an Sender übertragen werden, kann durch geschickte Nutzung der inkrementellen Redundanz eliminiert werden. Als Beispiel kann ein einfaches System zur Datenverteilung durch einen Sender unter einer Gruppe von direkt erreichbaren Empfängern betrachtet werden. Ohne Network Coding muss jeder Empfänger den Sender informieren, welche Pakete bereits angekommen sind und welche noch fehlen, damit dieser sie neu sendet. Unter Nutzung von Network Coding kann dies umgangen werden. Indem der Sender ununterbrochen zufällige Kombinationen aller zu übertragenen Pakete sendet, sorgt die Innovations-Eigenschaft des Codes dafür, dass mit hoher Wahrscheinlichkeit jeder Empfänger jedes Paket zur Decodierung verwenden kann, solange es ein Informationsgefälle zwischen Sender und Empfänger gibt (vgl. Lun u. a. 2006; Deb u. a. 2006). Nötig ist lediglich ein *Schlusspunkt*, der den Sender über die erfolgreiche Decodierung aller Pakete informiert. Fragouli u. a. (2007) stellen jedoch fest, dass in vielen Fällen zusätzliche Rückmeldungen trotzdem sinnvoll sind. Ohne zusätzliche Rückmeldungen können Knoten nie wissen, welche Pakete bereits übertragen sind und müssen alle empfangenen Pakete im Speicher vorhalten. Dies kann bei Knoten mit wenig Speicherkapazität problematisch sein. Rückmeldungen über die empfangenen Pakete können genutzt werden, um bei sendenden und weiterleitenden Knoten Speicher freizugeben. In diesem Rahmen skizzieren Sundararajan u. a. (2008) ein ARQ-Schema, das einerseits den Speicherbedarf der Knoten, andererseits auch die durch Decodierung erzeugte Verzögerung oder *burstiness* der Pakete verringert. Allgemein wird häufig das Prinzip der *Generationen* oder *Batches* angewandt, um Zahl der gleichzeitig verarbeiteten Pakete zu reduzieren.

Ein solcher Ansatz wird von Chachulski u. a. (2006, 2007a) mit *MORE* vorgestellt. MORE partitioniert einen Datenstrom an seiner Quelle in Batches, die jeweils aus meh-

reren Paketen bestehen. Aus den Paketen eines Batches werden dann zufällige Linearkombinationen erzeugt und gesendet. Jeder Knoten auf der Route erzeugt dabei neue Linearkombinationen aus den empfangenen Paketen, und sendet diese weiter. Sobald der Zielknoten genug linear unabhängige Kombinationen der Pakete eines Batches empfangen hat, um den Batch zu decodieren, sendet er ein ACK per Unicast in der entgegengesetzten Richtung zurück zur Quelle. Durch die so eingeführte inkrementelle Redundanz müssen wesentlich weniger ACK-Pakete gesendet werden, was den Durchsatz an Datenpaketen steigert. Zusätzlich können hier auch opportunistisch empfangene Pakete und Multipath-Routen ohne großen Koordinationsaufwand verwendet werden. Andere opportunistische Protokolle wie ExOR haben einen hohen Bedarf an Koordination. Dort muss für jedes Paket einzeln festgelegt werden, über welche Route es gesendet wird. Um redundante Übertragungen zu vermeiden, muss diese Festlegung allen Kandidaten für die Weiterleitung mitegeteilt werden. Bei MORE ist das nicht nötig. Jeder Knoten sendet zufällige Linearkombinationen der empfangenen Pakete und in Übereinstimmung mit den von Lun u. a. (2006) vorgestellten theoretischen Resultaten sind diese meist für nachfolgende Knoten nützlich. Als zusätzliches Problem tritt nun jedoch die Frage auf, wie die Senderaten der einzelnen Knoten bestimmt werden. Hier tritt ein weiterer Aspekt in Erscheinung, der im Modell des gewichteten Graphen nicht berücksichtigt ist. Im drahtlosen Medium konkurrieren Knoten um Sendezeit. Dies geschieht nicht nur zwischen Sender und Empfänger eines Pakets, sondern zwischen allen Knoten, die das Trägersignal eines potenziellen Senders empfangen können. Daher ist es ungünstig, einen Knoten, der nur wenig zum Fluss beiträgt, ununterbrochen senden zu lassen. Dies verringert die Senderaten der anderen, wichtigeren Knoten und somit den Gesamtdurchsatz. Dieses Problem wird in Abschnitt 3 näher untersucht.

Radunovic u. a. (2007b, a) schließlich, nehmen sich der Idee von MORE noch einmal an und optimieren sie unter expliziter Berücksichtigung der Fairness zwischen verschiedenen Datenströmen, der Sendeleistung, der Verfügbarkeit verschiedener Bitraten und der Scheduling-Bedingungen. So schlagen sie einen *optimal* Scheduling-Algorithmus vor, der sich nicht allein mit der Berechnung von Senderaten für die einzelnen Knoten begnügt, sondern schon auf der MAC-Schicht ansetzt und eine detaillierte Steuerung des Medienzugriffs voraussetzt. Eine Abwandlung des Algorithmus wird auch für *random* Scheduling, also den Einsatz des IEEE 802.11 MAC, präsentiert. All diese Aspekte werden in einem schichtübergreifenden Optimierungs-Framework berücksichtigt. Nach Optimierung der Parameter werden mittels Simulation erhebliche Durchsatzgewinne gegenüber MORE gezeigt. Allerdings wird dabei von einer verlust- und latenzfreien, dem

eigentlichen Medium externen, Signalisierungsschicht ausgegangen auf der beispielsweise ACK-Pakete übertragen werden. Zudem ist der optimale Scheduling-Algorithmus nicht vollständig dezentral, würde also weiteren Kommunikationsaufwand erfordern, sofern die Signalisierungsschicht im eigentlichen Medium implementiert würde.

2.5.4 Network Coding auf Symbolebene

Auch Katti u. a. (2007) sowie Katti und Katabi (2007) befassen sich mit Codierung innerhalb des selben Datenstroms. Hierbei werden nicht Pakete, sondern unmittelbar Radiosignale miteinander codiert. Es findet also eine Abbildung der Ideen von MORE auf die Datenübertragungsschicht statt . Durch dieses Vorgehen ist eine detailliertere Fehleranalyse möglich. So muss, wenn ein empfangenes Paket fehlerhaft ist, nicht das gesamte Paket verworfen werden, sondern die fehlerhaften Bits können mittels des Konfidenzwertes ermittelt und aussortiert werden. Zudem können Radiosignale bewusst interferierend gesendet und beim Empfänger durch *analog network coding* wieder etnziffert werden. Auf diese Art sind im Mittel deutlich weniger Sendeoperationen nötig, um eine festgelegte Menge an Daten zu übertragen, als dies bei MORE der Fall ist. Allerdings ist findet hier, wie auch bei MORE, eine End-zu-End-Codierung statt, bei der das ACK nach der vollständigen Decodierung durch den Empfänger die gesamte Route zurück reisen muss. Eine ähnliche Idee präsentieren Pu u. a. (2008) mit *continuous network coding* (CNC). Wie bei Katti u. a. (2007) wird der Wahrscheinlichkeitswert, mit dem jedes Bit demoduliert wird, ausgewertet. Allerdings geschieht dies nicht nur am Empfänger und nicht nur zur tatsächlichen Decodierung. Vielmehr nutzt jeder weiterleitende Knoten diese Inoformation und gibt sie mittels analoger Modulation in seinen Paketen weiter. Dabei wird eine speziell für diesen Zweck entwickelte XOR-Operation auf reellen Zahlen zwischen 0 und 1 verwendet, um mehrere von verschiedenen Quellen empfangene Pakete miteinander zu codieren und an deren gemeinsamen Empfänger wieder zu decodieren. Um verbleibende Bitfehler zu reparieren, wird ein *low density parity check code* (LDPC) verwendet. Pu u. a. (2008) zeigen anhand einer einfachen Beispieltopologie, in der es zwei Quellen, einen weiterleitenden Knoten und einen Zielknoten gibt erhebliche Durchsatzgewinne gegenüber anderen Methoden des Network Coding, die mit vollständig decodierten Bits arbeiten. Insgesamt leiden diese Verfahren unter dem praktischen Nachteil, dass sie nur mit spezialisierter Hardware realisierbar sind. Die IEEE 802.11 Datenübertragungsschicht ist normalerweise nicht für entsprechende Experimente zugänglich.

3 Senderatenkontrolle und Scheduling

Speziell im Umfeld des opportunistischen Routing wird die Zuordnung der Kapazität des Mediums an die verschiedenen potenziellen Sender - das Scheduling - zum Problem. Mittels Scheduling wird festgelegt, welche Knoten zu welchen Zeitpunkten senden dürfen. Ein eng verwandtes Thema ist die Fairness zwischen verschiedenen Datenströmen. Kreuzen sich zwei Datenströme an einer Stelle im Netzwerk, so stellt sich neben der Allokation des Mediums an Knoten auch die Frage der Allokation an Datenströme. Welcher Strom soll wie oft an welchen Knoten an die Reihe kommen und was soll damit erreicht werden? Im Fall eines einzelnen Stromes stellt meist der Datendurchsatz das Qualitätsmaß für eine diesem Strom lokale Optimierung dar. Unter Betrachtung der Fairness lassen sich demgegenüber verschiedene globale Optimierungsmöglichkeiten ausmachen. Der addierter Gesamtdurchsatz aller Ströme ist als Optimierungsziel denkbar, ebenso die möglichst gute Erfüllung der Durchsatzvorgaben jedes einzelnen Stroms. Hier wird oft mittels einer konkaven Funktion dem Durchsatz auf jedem Strom ein Nützlichkeitswert zugeordnet (vgl. Radunovic u. a. 2007b). Eine weitere Steigerung eines ohnehin hohen Durchsatzes wird dmenach als weniger nützlich betrachtet als eine gleiche Steigerung eines bisher niedrigen Durchsatzes. Über die Summe dieser Nützlichkeitswerte kann dann optimiert werden. Gänzlich unbeachtet bleibt häufig die Latenz der Datenübertragung, die jedoch bei vielen Anwendungen eine große Rolle spielt. Je länger der Weg, den ein Paket durch das Netzwerk nimmt, desto höher ist auch die End-To-End Latenz der Übertragung.

Auf einer einzelnen Route für einen einzelnen Unicast-Strom ohne Network Coding ist klar, dass jeder Knoten letztendlich gleich viele Pakete an seinen Nachfolger übermitteln muss. Gibt es hingegen mehrere mögliche Pfade, über die ein Paket übertragen werden kann, liegt also eine Multipath-Route vor, so ergeben sich hier bereits Wahlmöglichkeiten. Ein Knoten, der absichtlich weniger sendet als das Medium erlaubt, könnte damit für einen anderen, günstigeren Knoten zusätzliche Sendemöglichkeiten schaffen. Viele Anycast-Protokolle ohne Network-Coding umgehen dieses Problem, indem sie lokal für jedes empfangene Paket festlegen, welcher Empfänger es weiterleiten soll. ExOR (vlg. Biswas und Morris 2004, 2005) beispielsweise bestimmt dies anhand der ETX-Metriken des optimalen Pfades jedes Empfängers zum Ziel. Global betrachtet ist dieses Vorgehen nicht unbedingt optimal, wie am Beispiel in Abbildung 3 sichtbar wird. Die an den Kanten notierten Empfangswahrscheinlichkeiten zeigen, dass fast jedes bei S gesendete Paket sowohl von F1 als auch von F2 empfangen wird. ExOR wird für Übertragungen von S nach T jedoch praktisch ausschließlich den kürzeren Pfad über F1 wählen, obwohl

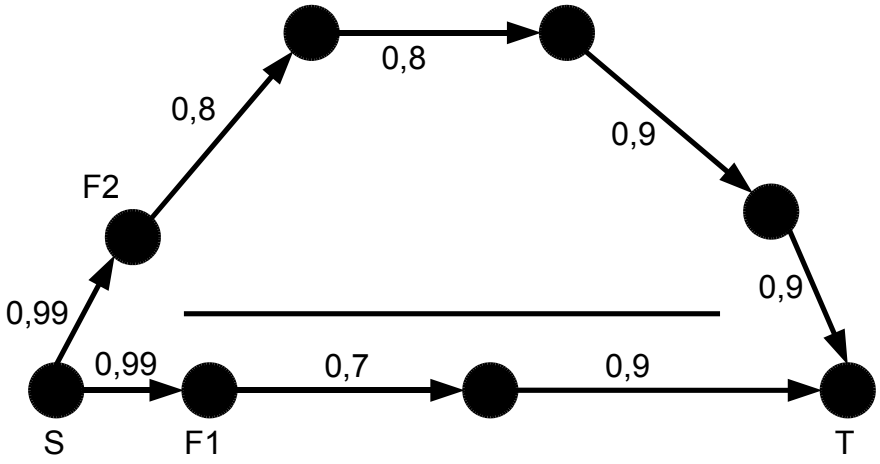


Abbildung 3 — Beispiel für Fehlfunktion des Scheduling bei ExOR

der Gesamtdurchsatz bei paralleler Nutzung des längeren Pfades über F2 deutlich höher wäre. Bei Nutzung von Network Coding ist die Allokation einzelner weiterzuleitender Pakete an einzelne Zwischenknoten nicht ohne Weiteres möglich, weshalb unter anderem Chachulski u. a. (2006) sowie Radunovic u. a. (2007b) sich mit dem Problem des Scheduling befassen. Zunächst ist das Scheduling aber implizit über die DCF des IEEE 802.11 Standards geregelt.

3.1 Die IEEE 802.11 DCF

Die im IEEE 802.11 Standard spezifizierte Trägerprüfung und das Backoffverfahren garantieren, dass jeder Knoten in einem Netzwerk im statistischen Mittel gleich viele Chancen erhält, das Medium für eigene Sendeoperationen zu nutzen (vgl. Bianchi 2000). Zudem sollen damit Kollisionen zwischen verschiedenen Sendeoperationen nach Möglichkeit vermieden werden und das Medium möglichst gut ausgelastet werden. Letztere Effekte treten nur in eingeschränktem Maße ein (vgl. Lee u. a. 2007). Falls die Trägerprüfung einer sendewilligen Station ergibt, dass das Medium belegt ist, so wird eine bestimmte Zeit gewartet und anschließend ein neuer Versuch unternommen. Diese Zeit ist das Backoff. Für Unicast-Übertragungen steigt das Backoff im Mittel exponentiell mit der Anzahl der erfolglosen Versuche, wobei ein Zufallsfaktor eingesetzt wird, um die Wahrscheinlichkeit zu minimieren, dass mehrere Stationen gleichzeitig den Träger als frei erkennen. Bianchi

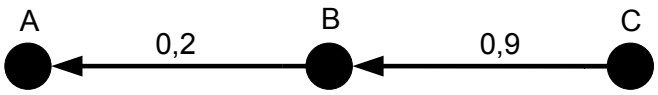


Abbildung 4 — Schlechtes Szenario für gleichverteilte Sendechancen

(2000) zeigen, dass dies reicht, um eine faire zeitliche Aufteilung des Mediums auf die Stationen im Netz zu erreichen. Für Broadcast- und Multicast-Übertragungen ist der Backoff jedoch konstant (vgl. 802.11 2007). Konkurrieren also Unicast- mit Broadcast- oder Multicast-Übertragungen um das Medium, so sind die Unicast-Verbindungen im Nachteil, da sie im Mittel länger warten, wenn das Medium belegt ist und somit eine Broadcast-Übertragung in der Zwischenzeit beginnen kann.

Zusätzlich ist eine Gleichverteilung der Sendechancen zwischen Knoten nicht immer erwünscht. Ein einfaches Beispiel ist ein Datenstrom über einen Zwischenknoten, der für eingehende und ausgehende Hops deutlich unterschiedliche Übertragungswahrscheinlichkeiten aufweist. Dies wird am Szenario in Abbildung 4 deutlich. Um ein Paket von C nach B zu übertragen sind bei den annotierten Empfangswahrscheinlichkeiten erwartete 1,1 Sendeoperationen nötig. Um das selbe Paket von B nach A weiterzuleiten sind erwartete 5 Sendeoperationen nötig. Der Zwischenknoten muss hier also deutlich mehr senden, als er selbst empfängt, um die optimale Durchsatzrate zu erreichen. Bei Unipath-Strömen, die das IEEE 802.11 ARQ-Protokoll verwenden ist dies kein Problem. Das Protokoll auf der Transportschicht, z.B. TCP, wird nach einer Weile keine neuen Pakete mehr zur Verfügung stellen, falls zu wenige Pakete ankommen. Daraufhin wird der Quellknoten, in diesem Fall C, das Senden einstellen und der Zwischenknoten, hier B, entsprechend mehr Sendemöglichkeiten bekommen. Bei Multipath-Routen reicht dies jedoch nicht. Hier ist es wichtig, verschiedenen Knoten bewusst Sendezeit oder weiterzuleitende Pakete zuzuweisen, um einen optimalen Durchsatz zu erreichen. Ein Protokoll, das Network Coding betreibt, kann das IEEE 802.11 ARQ-Protokoll nicht kaum verwenden und muss daher auch im Unipath-Fall ein ähnliches Problem lösen. Dieses Problem stellt sich folgendermaßen dar: Jeder Knoten kann beliebig viele Linearkombinationen seiner Pakete erzeugen. Sendet ein Knoten mehr Pakete als nötig sind, um die gewünschte Menge an Information an designierte Nachfolgerknoten zu übertragen, so wird unnötige Redundanz gesendet. Gleichzeitig werden andere, wichtigere Knoten vom Senden abgehalten.

Die Defizite der DCF können auf verschiedene Weise angegangen werden. Eine Möglichkeit ist eine indirekte Beeinflussung über die Regelung der Senderate jedes Knotens auf der Sicherungs- oder Netzwerkschicht bei sonstiger Beibehaltung von zumindest Teilen der DCF-Funktionalität. Dabei können die Sendezeitpunkte für einzelne Frames nicht zwischen Knoten koordiniert werden. Falls versteckte Knoten vorhanden sind, wird es demnach weiterhin zu Kollisionen kommen. Zudem kann es passieren, dass durch die zufällige Wahl der Zeitpunkte und Backoffs während eines großen Teils der Zeit das Medium ungenutzt bleibt, obwohl rechnerisch die addierten Senderaten eine vollständige Ausnutzung suggerieren. Der Vorteil dieses Verfahrens ist, dass es auf Standardhardware umsetzbar ist. Diesen Ansatz wählen Chachulski u. a. (2006) mit MORE und Radunovic u. a. (2007b) mit der *random scheduling* Variante ihres Algorithmus.

Eine weitere Möglichkeit stellt die bewusste Koordination aller Sendezeitpunkte dar. Eine Funktion bestimmt für jeden Knoten zu welchem exakten Zeitpunkt er welches Paket zu senden hat. Hier wird also die DCF komplett ausgetauscht. Der Vorteil dieses Ansatzes ist, dass keine Kollisionen mehr stattfinden können und das Medium vollständig ausgenutzt wird, sofern die neue Koordinationsfunktion in diesem Sinne optimal arbeitet. Allerdings ist die DCF bei Standardhardware nicht modifizierbar, weshalb die Erprobung solcher Verfahren vorerst auf Simulationen und spezialisierte Geräte beschränkt bleibt. Dieser Ansatz wird von Radunovic u. a. (2007b) mit der optimalen Variante ihres Algorithmus verfolgt.

3.2 nach Chachulski u. a.

3.2.1 Funktionsweise

Chachulski u. a. (2006) stellen ein Optimierungs-Framework vor, in dem ein Paket mit statistisch möglichst wenig Sendeoperationen von einem Quellknoten zu einem Zielknoten übertragen werden soll. Zwischen Quelle und Ziel stehen eine Reihe von Zwischenknoten zur Verfügung, die Pakete weiterleiten können. Zwischen den Knoten herrschen unterschiedliche Empfangswahrscheinlichkeiten. Eine Empfangswahrscheinlichkeit $p(i, j)$ ist die Wahrscheinlichkeit, dass eine Sendeoperation bei Knoten i zu einer Übertragung des gesendeten Pakets von i nach j führt. Es handelt sich hier also um den Kehrwert der ETX-Metrik. Analog ist $p(i, J)$ die Wahrscheinlichkeit, dass die Sendeoperation zu einer Übertragung von i zu mindestens einem Knoten aus der Knotenmenge J führt. Die Zahl der Sendeoparationen des Knoten i wird dabei als z_i bezeichnet und ist von der Zahl der tatsächlich von i an einen anderen Knoten j übertragenen Pakete x_{ij} zu

unterscheiden. Als Einschränkung der Optimierung tritt die Flusserhaltung hinzu. Ein Knoten kann nicht mehr Pakete an andere übertragen, als er selbst empfängt. Insgesamt wird also $\sum_{i \in N} z_i$ minimiert, wobei N die Gesamtmenge der Knoten ist. Dies geschieht mit folgenden Einschränkungen:

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} 1 & \text{falls i die Quelle ist} \\ -1 & \text{falls i das Ziel ist} \\ 0 & \text{sonst} \end{cases} \tag{1}$$

$$x_{ij} \geq 0 \; \forall (i,j) \tag{2}$$

$$z_i p(i,J) \geq \sum_{j \in J} x_{ij} \tag{3}$$

In Chachulski u. a. (2007a) wird, hier in der Notation von Chachulski u. a. (2006) wiedergegeben, folgende Lösung dieses Optimierungsproblems mittels rekursiver Berechnung der Werte für z_i gegeben. Sei nun $L_i = \sum_{j \in N} x_{ij}$, s die Quelle eines Flusses und d sein Ziel. Die Relation $i < j$ für zwei Knoten i und j sei folgendermaßen bestimmt:

$$i < j \Leftrightarrow ETX(i,d) < ETX(j,d)$$

Weiterhin sei $\bar{p}(i,j)$ die Wahrscheinlichkeit, dass j ein von i gesendetes Paket nicht empfängt:

$$\bar{p}(i,j) = 1 - p(i,j)$$

$\bar{P}(i,j)$ sei die Wahrscheinlichkeit, dass kein Knoten $k < j$ ein von i gesendetes Paket empfängt:

$$\bar{P}(i,j) = \prod_{k<j} \bar{p}(i,k)$$

Klar ist nach (1), dass $L_s = 1$. Es wird implizit angenommen, dass $i \leq s \; \forall i \in N$. Weiterhin wird davon ausgegangen, dass die Übertragungswahrscheinlichkeiten für alle Knotenpaare bei der Berechnung zur Verfügung stehen. Nun wird folgende Berechung für jeden Knoten j auf der Basis der z-Werte aller Knoten $i > j$ durchgeführt.

$$L_j = \sum_{i>j} z_i p(i,j) \bar{P}(i,j) \tag{4}$$

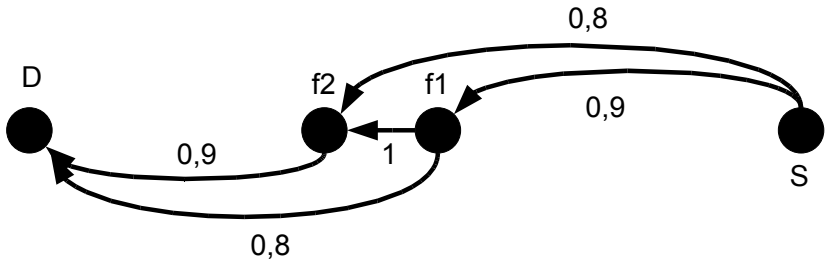


Abbildung 5 — Dicht benachbarte Knoten

Von Knoten $i > j$ empfangene Pakete - $z_i p(i, j)$ - soll j also weitergeben, falls kein anderer Knoten $k < j$ sie empfangen hat. Um eine erfolgreiche Weitergabe zu garantieren, muss z_j folgendermaßen beschaffen sein:

$$z_j = L_j/(1 - \bar{P}(j, j)) \tag{5}$$

3.2.2 Mängel des Ansatzes

Zunächst fällt bei der Betrachtung des Algorithmus auf, dass die paarweisen Empfangswahrscheinlichkeiten bekannt und hinreichend konstant sein müssen, um ein sinnvolles Scheduling zu ermöglichen. In der Praxis erweist sich diese Bedingung als schwierig. Die Empfangswahrscheinlichkeiten sind aufgrund des Fadingeffekts starken Schwankungen ausgesetzt und zudem wird ihre Messung durch die vorliegenden Sendeoperationen und daraus resultierende Kollisionen und verringerte Sendemöglichkeiten selbst beeinflusst. Zubow u. a. (2006) sprechen letzteres Problem in der Diskussion ihrer *local neighbour discovery* (LND) an. Ein weiteres Problem zeigt sich anhand von nah bei einander liegende Knoten mit je hohen Empfangswahrscheinlichkeiten in beide Richtungen. Diese werden stark unterschiedlich behandelt. In der in Abbildung 5 gezeigten Topologie würde beispielsweise $f1$ kaum Pakete weiterleiten, während $f2$ fast die ganze Last trägt, obwohl der Pfad über $f2$ nicht besser ist. Dieses Verhalten ist nicht sehr robust. Fällt $f2$ aus oder ändern sich seine Empfangswahrscheinlichkeiten stark, so bleibt der Fluss praktisch stehen bis die Schedulingwerte neu berechnet werden. Nachdem Mobilität und Kanalveränderungen jedoch offenbar kein Thema beim Entwurf von MORE waren, können diese Punkte zunächst außen vor bleiben. Mobilität spield allgemein in dieser Arbeit keine Rolle. Alle Knoten werden als räumlich fixiert angesehen. Dem Kanal inhärente Effekte wie Fading können jedoch eigentlich nicht ignoriert werden.

Zudem gilt, wie auch Radunovic u. a. (2007b) anmerken, dass ein Scheduling ohne

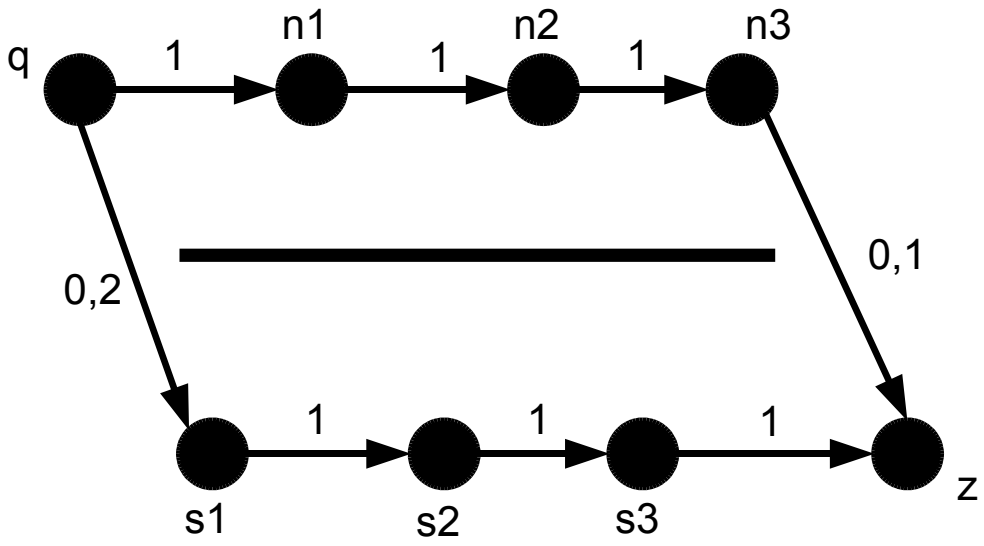


Abbildung 6 — Beispiel für Fehlfunktion des MORE-Scheduling

globalen Fairnessanspruch in echten Szenarien mit mehreren Flüssen zu ungünstigen Ergebnissen führen kann. Radunovic u. a. (2007b) zeigen dies anhand eines Vergleichs von MORE, als Scheduling ohne Fairnessanspruch, mit ihrem eigenen, MC^2. MC^2 weist dabei nicht nur einen höheren Gesamtdurchsatz auf, sondern schafft es auch, den vorhandenen Durchsatz gerechter auf verschiedene Flüsse zu verteilen, was mittels einer Nützlichkeitsfunktion gemessen wird. Auch dieses Problem kann an dieser Stelle im Rahmen des MORE-Frameworks jedoch nicht weiter verfolgt werden. Es wird im Folgenden davon ausgegangen, dass nur ein einzelner Strom im Netzwerk vorhanden ist und über diesen allein optimiert.

Ein bedeutenderes Problem ist in den Zielsetzungen der Optimierung angelegt. (3) legt fest, dass ein Knoten i die ihm zugeschriebenen Paketübertragungen x_{ij} im statistischen Mittel auch vollbringt. $z_i p(i, J)$ sind die insgesamt von i an Knoten aus J übertragenen Pakete bei z_i Sendeoperationen. Dabei spielt es keine Rolle, welche genauen Knoten welche Pakete empfangen.

Deshalb ist es möglich, dass bei z_i Sendeoperationen ein Knoten $j \in J$ mit hohem $p(i, j)$ mehr Pakete empfängt als x_{ij} vorgibt, während ein Knoten $k \in J$ mit niedrigerem $p(i, k)$ weniger Pakete empfängt als x_{ik} vorgibt. Trotzdem werden in der Summe genug

Pakete empfangen, um die formulierte Bedingung zu erfüllen. Daraus ergibt sich, dass die Werte für z_i häufig zu klein sind, um eine tatsächliche Flusserhaltung zu erzeugen. Am Beispiel in Abbildung 6 kann das illustriert werden. Die Annotationen der Kanten geben die Empfangswahrscheinlichkeiten p an. Es seien nun 10 Pakete bei q zum Senden vorhanden. Die Summe der x_{qj} mit $j \in \{n1, s1\}$ ist also nach (1) 10, $p(q, J)$ ist offensichtlich 1. q wird demnach 10 Pakete senden, folglich sind anschließend 10 Pakete nach $n1$ übertragen. Das reicht nach (3). Von diesen 10 Paketen hat $s1$ 2 empfangen, die mit 6 Sendeoperationen an z weitergeleitet werden können. $n1$ kann 8 Pakete über $n2$ nach $n3$ weiterleiten und braucht dafür 16 Sendeoperationen. n_3 muss nun 80 Mal senden, um die 8 Pakete an z zu übertragen. Insgesamt finden also 122 Sendeoperationen statt. Anzumerken bleibt, dass das unabhängig von den x_{ij} bei $i \in \{n1, n2, n3, s1, s2, s3\}$ geschieht. Diese können aufgrund der von q angebotenen Paketmenge nicht realisiert werden, was in der Optimierung aber nicht berücksichtigt wird. Optimal wären nach dem ausgeführten Schema offensichtlich folgende Parameter:

$$x_{qn1} = x_{n1n2} = x_{n2n3} = x_{n3z} = 0$$

$$x_{qs1} = x_{s1s2} = x_{s2s3} = x_{s3z} = 1$$

Eine Veranschaulichung bestätigt das. Würde q 50 Mal senden, so kämen bei s 10 Pakete an, die mit 30 Sendeoperationen weitergeleitet werden könnten. Damit fänden in diesem Szenario nur 80 Sendeoperationen statt. MORE hingegen würde das selbe Szenario auf 10 Sendeoperationen bei q, also insgesamt 40, optimieren und fälschlicherweise annehmen, dass $s1$ den vorgesehenen Fluss von 1 realisieren kann.

Der tatsächlich in (4) und (5) vorgestellte Algorithmus folgt jedoch nicht dem Optimierungsschema, sondern dem Zwang der Flusserhaltung und würde stattdessen $L_{n1} = 0,8$ und $L_{s1} = 0,2$ mit den entsprechenden Konsequenzen für z_{n1} und z_{s1} berechnen.

Um dieses Problem auszugleichen, müsste (3) folgendermaßen umformuliert werden:

$$z_i p(i, j) \geq x_{ij} \; \forall j \in J \tag{6}$$

Am Beispiel von Abbildung 6 zeigt sich aber noch ein weiteres Problem. Es ist nicht optimal, q 50 Mal senden und den ganzen nördlichen Pfad währenddessen brachliegen zu lassen, selbst wenn dies die Gesamtzahl der Sendeoperationen minimiert. Unter Nutzung des nördlichen Pfades könnte q sich beispielsweise auf 35 Sendeoperationen beschränken. Auf dem südlichen Pfad würden dann die dort ankommenden 7 Pakete weitergeleitet, während auf dem nördlichen Pfad 3 Pakete mit 30 Sendeoperationen bei $n3$ übertragen

würden. Das würde zu einer Verringerung der Last am vorigen Flaschenhals q und zu einer schnelleren Übertragung des Batches führen. Dies wiederum wäre mit höherem Durchsatz verbunden, da der nächste Batch früher gestartet würde.

3.2.3 Korrektur des Optimierungsziels

Um ein sinnvolleres Scheduling auf der Basis von MORE zu erstellen, bietet es sich an, neben der angesprochenen Korrektur von (3) auch das Optimierungsziel zu ändern. Anstatt insgesamt möglichst wenige Sendeoperationen zu fordern, wäre es offenbar sinnvoll, möglichst wenige Sendeoperationen in der selben Kollisionsdomäne anzustreben. Nicht kollidierende zusätzliche Sendeoperationen wären, zumindest für die lokale Optimierung eines Flusses, kein Problem. Ein Flaschenhals, wie q in Abbildung 6 führt hingegen offensichtlich zu Performancenachteilen. Für die Definition dieses Ziels stellt sich zunächst die Frage, was die Kollisionsdomäne ausmacht. Hier kann einerseits explizit die Trägerprüfung herangezogen werden. Andererseits kann, falls dies nicht möglich ist, indirekt über die An- oder Abwesenheit einer messbaren ETX-Metrik bestimmt werden, welche Knoten potenziell kollidierend senden. Letzteres ist allerdings eine sehr grobe Abschätzung, da die Trägerprüfung üblicherweise auch die Existenz sehr weit entfernter Knoten feststellen kann, deren Frames nie empfangen werden können. Weiterhin muss geklärt werden, inwiefern Kollisionsdomänen reziprok sind, also ob die Anwesenheit eines Knotens a in der Kollisionsdomäne von b bedeutet, dass a die Anwesenheit von b feststellen kann, oder umgekehrt. Weiterhin muss unterschieden werden zwischen Knoten, die einen Knoten a am Senden hindern und solchen, die den Empfang der Frames von a bei b stören, da dies nicht unbedingt die selben sind (vgl. Lee u. a. 2007). Um all das möglichst genau zu erfassen, sei $K_{ij} \subset N$ die Kollisionsdomäne zweier Knoten i und j, also die Menge aller Knoten, deren Trägersignal eine Übertragung von i nach j stört. Einerseits sind das diejenigen Knoten, die i mittels Trägerprüfung entdecken kann und die i daher vom Senden abhalten. Andererseits sind das auch die Knoten, die mit ihrem Signal den Empfang eines Signals von i bei j stören. Damit kann das folgende primäre Optimierungsziel festgelegt werden:

$$argmin \left(\max_{i,j \in N | x_{ij} > 0} \left(\sum_{k \in K_{ij}} z_k \right) \right) \tag{7}$$

Es soll also die maximale Last auf einer Verbindung zweier beliebiger Knoten i und j minimiert werden. Die Last umfasst in diesem Zusammenhang nicht nur die Sende-

operationen zur Übertragung von Paketen auf dieser Verbindung selbst, sondern auch alle potenziell damit kollidierenden Sendeoperationen. In dieser Formulierung ist es allerdings möglich, optimale Lösungen zu finden, die große Umwege mit vielen unnötigen Hops enthalten. Daher wird das ursprüngliche Optimierungsziel, die Minimierung der Gesamtzahl der Sendeoperationen, als sekundäres beibehalten. Gibt es also mehrere optimale Lösungen nach (7), so wird diejenige mit den insgesamt wenigsten Sendeoperationen präferiert. Eine Lösung für dieses Problem ist leider nicht direkt offensichtlich.

3.2.4 Lösungsansätze nach Kong u. a.

Kong u. a. (2008) können für einen Lösungsansatz herangezogen werden. Sie definieren nach Cover und Thomas (1991) die Kapazität eines Links in einem Drahtlosnetzwerk über die *signal to interference and noise ratio* (SINR) β_{ij}.

$$C_{ij} = \begin{cases} 1/2log(1+\beta_{ij}) & \beta_{ij} \geq \beta, \\ 0 & \beta_{ij} < \beta \end{cases}$$

β ist der Schwellenwert ab dem ein Empfang prinzipiell möglich ist. β_{ij} ist dabei so definiert, dass von gleichzeitiger Aktivität aller Knoten ausgegangen wird.

$$\beta_{ij} = \frac{P_i L(d_{ij})}{N_0 + \gamma \sum_{k \neq i,j} P_k L(d_{kj})}$$

P_i ist dabei die Sendeleistung des Knotens i, $L(\cdot)$ gibt den Pfadverlust eines Signals in Relation zur zurückgelegten Strecke an, N_0 ist das konstante Hintergrundrauschen und γ ist die Inverse des *system processing gain*, der Resistenz der Modulation gegenüber Störsignalen (vgl. Sklar 2001, Kap. 12). Das Problem stellt sich in dieser Form als die Frage, welche Knoten mit welcher Sendeleistung senden müssen, um über die Einzelkapazitäten C_{ij} die Kapazität des kleinsten Schnittes C_k zwischen Quelle und Ziel zu maximieren. Einschränkung bleibt die Flusserhaltung.

Diese Formulierung erzeugt nur ein Optimierungsziel und ist damit praktikabler als die vorige. Jedoch war die Sendeleistung eines Knotens bisher kein Thema. Stattdessen wurde davon ausgegangen, dass interferierende Knoten den Kanal nicht mittels Sendeleistung, sondern zeitlich untereinander aufteilen. Hier müsste also noch eine Abbildung vorgenommen werden.

Auch die grundlegende Schwierigkeit bei der Lösung des Problems bleibt in dieser Formulierung erhalten. Kein greedy Algorithmus kann hier auch nur eine annähernd op-

timale Lösung erzeugen, da mit der Betrachtung der Interferenz eine Rückkopplung der Qualität jedes Hops mit vorhergehenden Hops stattfindet. Ein Brute-Force-Algorithmus, der alle Kombinationen der Sendeleistungen oder zeitlichen Anteilen am Medium der Knoten auf der Route vergleicht, wäre entsprechend aufwändig und somit unpraktikabel. Kong u. a. (2008) zeigen, dass die Kapazität eines kleinsten Schnittes stark um einen Erwartungswert konzentriert ist. Allerdings gehen sie dabei probabilistisch vor und setzen eine Zufallsverteilung der Knoten und Sendeleistungen voraus. Hier jedoch wäre der umgekehrte Weg nötig. Die Kapazität des Schnittes soll maximiert werden, während die Knoten und Sendeleistungen bewusst gewählt werden können. Daher kann dieses Ergebnis hier nicht ohne Weiteres genutzt werden.

Zudem bleibt das allgemeine Problem der potenziell veralteten Link-Metriken bei jeder statischen Lösung des Schedulingproblems erhalten. Ebenfalls unberücksichtigt ist auch weiterhin die Fairness zwischen verschiedenen Flüssen. Die von Radunovic u. a. (2007b, a) vorgestellte dynamische Lösung kommt ohne Rückgriff auf global bekannte Linkmetriken aus, berücksichtigt explizit die Fairness und impliziert keinen großen rechnerischen Aufwand. Sie erscheint daher grundsätzlich attraktiver.

3.3 nach Radunovic u. a.

Radunovic u. a. (2007b, a) greifen bei ihrer Lösung des Scheduling- oder *rate control* Problems auf einige ältere Werke zum Thema zurück. Sie modifizieren die dort vorgestellten Verfahren hinsichtlich der Broadcast-Eigenschaft des Mediums und der Tatsache, dass aufgrund des Network Coding eine exakte Koordination einzelner Pakete nicht nötig ist. Verwendet wird dabei einen Backpressure-Algorithmus. Die Länge der Queue eines nachfolgenden Knotens wird als das Maß der Überlastung dieses Knotens interpretiert. Dieses Maß leitet die Entscheidung über die Verteilung der zu übertragenden Pakete. Je mehr Pakete ein Knoten weiterzuleiten hat, desto mehr muss er senden, weshalb dies implizit eine Lösung des Schedulingproblems darstellt. Das Routingproblem, also die Auswahl der für die Verteilungsentscheidung verfügbaren Knoten, wird auch hier zunächst vom Schedulingproblem unabhängig betrachtet.

Zur Charakterisierung des Backpressure-Algorithmus werden zwei grundlegende Arten von *Credit* definiert. $C(n, c)$ ist die Zahl der Pakete, die ein Knoten n für eine Generation eines Flusses c übertragen soll. C-Credits werden an der Quelle des Flusses mit den Paketen erzeugt und an seinem Ziel durch Auslieferung der Pakete konsumiert. Als Absichtserklärungen werden sie vor den eigentlichen Paketen an weiterleitende Knoten übertragen. Dies kann durch Annotation der sowieso zu sendenden Pakete geschehen.

Der *transmission credit* $TC(n, c, J)$ ist die Zahl der Pakete, die von einem Knoten n für eine Generation aus Fluss c an mindestens einen Knoten $m \in J$ übertragen werden sollen. TC-Credits werden genutzt, um Pakete, die von verschiedenen Knoten gleichzeitig empfangen wurden nicht als Übertragung verschiedener Pakete zu zählen. Sie werden an jedem Knoten n für jeden Fluss c und jede Teilmenge der Menge der direkt nachfolgenden Knoten $J \subseteq DST(n, c)$ mitgezählt. Nachdem von typischerweise zwei oder drei nachfolgenden Knoten pro Knoten und Fluss ausgegangen wird, bleibt die Zahl der verschiedenen TC-Credits handhabbar.

Wird nun ein C-Credit von n nach m übertragen, so inkrementiert n die TC-Credits für alle Knotenmengen J, in denen m enthalten ist. Empfängt m ein Paket von n, so dekrementiert n die TC-Credits für alle diese Mengen. Pro gesendetem Paket wird der TC-Credit aber für jede Teilmenge J höchstens einmal dekrementiert. Falls also mehrere Credits an verschiedene Knoten aus einer Teilmenge J verteilt wurden, so sind ebenso viele erfolgreiche und verschiedene Paketübertragungen nötig, um die entstandenen TC-Credits für J wieder abzubauen. Die TC-Credits geben damit außerdem an, in welche *Richtung* ein Fluss c von n aus übertragen werden soll, während C-Credits nur die allgemeine Zuständigkeit von n für eine Anzahl Pakete aus c ausdrücken.

Die Übertragung von C-Credits wird von der Menge der bereits vorhandenen C- und TC-Credits abhängig gemacht, um den Fluss auf möglichst wenig belastete Knoten zu legen. Hierzu wird zunächst der *cumulative transmission credit* CTC definiert:

$$CTC(n, c, m) = \sum_{J \subseteq DST(n,c), m \in J} |TC(n, c, J)|$$

$CTC(n, c, m)$ ist ein Maß für die Zahl der Pakete einer Generation aus Fluss c, die noch von n nach m übertragen werden müssen. Diese Eigenschaft wird für die Bestimmung der Creditübertragung genutzt. Ein Knoten n überträgt einen C-Credit an einen Knoten m, falls:

$$|C(n, c)| > |C(m, c)| + |CTC(n, c, m)| \tag{8}$$

Die für die Berechnung von TC nötigen Empfangsbestätigungen werden entweder an Pakete anderer, in die Gegenrichtung laufender, Flüsse annotiert oder an solche Pakete, die der nächste Knoten weiterleitet. Bei letzterem Verfahren wird die Tatsache genutzt, dass diese Pakete auch von vohergehenden Knoten mitgehört werden können. Hier wird also keine zuverlässige Übertragung der Bestätigungen angestrebt. Die Signalisierung wird vielmehr konzeptionell außerhalb des eigentlichen Mediums angesiedelt.

Um die tatsächlichen Senderaten bzw. Sendezeitpunkte der Knoten zu bestimmen, wird ein kompliziertes übergreifendes Optimierungsproblem über Scheduling, Bitraten und Sendeleistung gestellt. Dessen Lösung wird aber als schwierig zu implementieren und NP-schwer erkannt. Eine vereinfachte Version wird als Abbildung auf die Bedingungen des IEEE 802.11 Standards beschrieben, das Random Scheduling. Dabei wird davon ausgegangen, dass die DCF die Zuordnung von Sendegelegenheiten übernimmt, und jeder Knoten mit maximaler Leistung sendet. Eine lokale Qualitätsmetrik, die einen Knoten und seine Nachbarn im Netzwerk umfasst, wird herangezogen: $Q(n, J)$ ist die Wahrscheinlichkeit, dass mindestens ein Knoten aus J ein von n mit der für diesen Hop günstigsten Bitrate gesendetes Paket empfängt. Ist dieser Wert bei n bekannt, so kann für jeden Strom c eine Priorität berechnet werden:

$$W(n, c) = \sum_{J \subseteq DST(n,c)} TC(n, c, J)Q(n, J)$$

$W(n) = max_c(W(n, c)$ ist nun die Priorität eines Knotens und $c(n) = arg\, max_c W(n, c)$ ist der Strom mit der höchsten Priorität an Knoten n. Ist (8) an einem Knoten n erfüllt, so generiert dieser Knoten Pakete für den Strom $c(n)$, stellt diese in die Sendequeue und überlässt der DCF die Details. Für diese Variante der Lösung des Scheduling-Problems wird jedoch keine Qualitätseigenschaft gezeigt. Mittels Simulation wird lediglich ein Vergleich mit MORE durchgeführt, der MC^2 als hinsichtlich des erreichbaren Durchsatzes als deutlich überlegen darstellt. Auf welche Art die widersprüchliche Darstellung des Scheduling bei MORE aufgefasst wurde, wird jedoch nicht erläutert.

Es bleibt ebenfalls anzumerken, dass hier nicht genau geklärt wird, wie ein Knoten rechtzeitig über den Empfang seiner Pakete bei einem designierten Empfänger und damit die entsprechende Reduktion der TC-Credits informiert wird. Trifft diese Information verspätet ein, so wird der Knoten überflüssige, nichtinnovative Pakete senden. Ein reines Warten auf die nächste Sendegelegenheit des Empfängerknotens muss hier, aufgrund der Unberechenbarkeit der 802.11 MAC-Schicht und der Ungewissheit über die Menge der Credits am Empfänger, als unzureichend angesehen werden.

4 Bitfehlermessungen

Im Vorfeld dieser Arbeit wurden, ausgehend von den von Aguayo u. a. (2004), Chebrolu u. a. (2006), Souryal u. a. (2006), Willig u. a. (2002) und Dubois-Ferriere u. a. (2005) beschriebenen Ergebnissen, einige Messungen zur Häufigkeit und Verteilung von Bitfehlern

vorgenommen. Die genannten Werke widersprechen sich teilweise in ihren Ergebnissen über diese Verteilungen. So stellen beispielsweise Chebrolu u. a. (2006) nur sehr wenige und kaum gehäufte Bitfehler fest. Die anderen genannten Werke legen nahe, dass speziell Verbindungen von mittlerer Qualität für Maschennetzwerke von entscheidender Bedeutung sind. Gleichzeitig kommt diese mittlere Qualität durch häufige Fehler beim Empfang von Frames auf der Datenübertragungs- und Sicherungsschicht zu Stande. Knoten sind in diesem Fall also in der Lage, den Frame an sich zu erkennen, können ihn jedoch nicht korrekt entschlüsseln. Dies kann erstens an einem inkorrekt empfangenen PLCP-Header liegen. Zweitens kann ein Empfangsfehler durch Verkürzung des Frames zu Stande kommen. In diesem Fall wird der Frame nur bis zu einem bestimmten Punkt empfangen, danach verliert der Knoten die Trägersynchronisation. Drittens schließlich kann ein Fehler durch einzelne inkorrekt decodierte Bits im Frame hervorgerufen werden. Die so entstandenen Bitfehler führen dazu, dass die Checksumme am Ende des Frames nicht mehr mit der aus dem Inhalt des Frames berechenbaren Checksumme übereinstimmt. In all diesen Fällen wird der Frame normalerweise verworfen. Je nach Charakteristik der Bitfehler sind unterschiedliche Protokolle zum Ausgleich der Bitfehler sinnvoll. Daher wurden noch einmal eigene Messungen vorgenommen, um die genannten Ergebnisse zu prüfen. Der MADWIFI-Treiber für Atheros-Chipsätze kann so modifiziert werden, dass zumindest verkürzte Frames und solche mit inkorrekter Checksumme ausgeliefert werden. Diese Funktion wurde genutzt, um gesendete Daten mit empfangenen zu vergleichen und so Bitfehlermuster zu erkennen.

Die verwendete Testumgebung bestand aus Netgear wgt634u APs, die in zwei benachbarten Gebäuden verteilt wurden. Es wurden drei Szenarien betrachtet. Zunächst wurden Frames von einem Knoten bei verschiedenen Bitraten im Broadcast-Modus gesendet und von allen anderen aufgezeichnet. Anschließend wurde das Experiment wiederholt, wobei ein weiterer Knoten als Interferenzquelle ebenfalls im Broadcast-Modus Pakete aussandte. Schließlich wurde eine Unicast-Übertragung entlang einer festgelegten Route über mehrere Knoten hinweg getestet. Die aufgezeichneten Pakete wurden zentral gesammelt und nachträglich analysiert. Als zentrale Ergebnisse zeigten sich:

- Bitfehler treten gehäuft auf. Die Häufungen sind zwischen 4 und 64 Bytes lang. Außerhalb dieser Häufungen treten kaum Bitfehler auf.

- In den meisten fehlerhaften Paketen finden sich nur wenige Bitfehler. Eine Häufung von 64 Bytes ist im Vergleich zur typischen Länge der Pakete - etwa 1500 Bytes - relativ kurz.

- Bei manchen Verbindungen, speziell wenn die Empfänger nah beieinander liegen, sind die Bitfehler, beim Empfang des selben Pakets, zwischen verschiedenen Knoten korreliert. Es kann jedoch kein genereller Zusammenhang zwischen Entfernung der Knoten und Korrelation der Bitfehler ausgemacht werden.

5 Problembeschreibung

Ausgehend von den bisherigen Erkenntnissen kann nun eine genauere Beschreibung des zu entwickelnden Systems gegeben werden. Ziel ist die Entwicklung eines prototypischen Coding- und Schedulingprotokolls für Multihop- und Multipath-Übertragungen in Maschennetzwerken, das auf festgelegten Routen einen möglichst hohen Durchsatz erzielt, wobei die Latenz der Übertragung als Sekundärcharakteristikum im Auge behalten wird. Im Rahmen des Prototyps reicht es, Singlepath-Routen auf opportunistische Art zu verwenden. Opportunistischer Empfang ist auf auf jeden Fall sinnvoll, um den Durchsatz zu verbessern, wie aus Abschnitt 2.4 hervorgeht. Auch die Ergebnisse aus den Abschnitten 2.2 sowie 4 können in einem kombinierten Ansatz nutzbar gemacht werden. Zunächst ergibt sich aus den verschiedenen Bitfehlermessungen, dass eine Fragmentierung der Pakete, wie beispielsweise von Dubois-Ferriere u. a. (2005) dargestellt günstig ist. Aufgrund der Tatsache dass Bitfehler gehäuft auftreten, ist es nicht unbedingt nötig, wie von Katti u. a. (2007) und Pu u. a. (2008) vorgeschlagen, die Fragmentierung direkt auf Symbolebene anzusetzen. Es kann vielmehr auch eine Aufteilung in größere Fragmente gewählt werden. Die Fragmentierung wird dann in einem Network Coding Schema genutzt, um inkrementelle Redundanz zu erzeugen. Dieses Verfahren kann an Chachulski u. a. (2007b) oder Radunovic u. a. (2007b) angelehnt sein. Es wird auf jeden Fall den zentralen Vorteil des Coding gegenüber einfacher Paketkombination nutzen (vgl. Tse und Viswanath 2005, Kap. 3). Zur Kontrolle der inkrementellen Redundanz ist ein HARQ-Mechanismus nötig. Hier kann beispielsweise auf die Ideen von Subramanian u. a. (2007) zurückgegriffen werden. Spezielles Augenmerk wird darauf gerichtet, dass nur soviel Redundanz übertragen wird, wie tatsächlich nötig ist. Im Umfeld des Network Coding heißt dies, dass die Zahl der nichtinnovativen Pakete oder Fragmente minimiert wird. End-To-End ACK-Pakete, wie beispielsweise von Chachulski u. a. (2007b) vorgeschlagen erfüllen dieses Kriterium nicht. Es wird vermutet, dass an dieser Stelle auch bei Radunovic u. a. (2007b) noch Optimierungspotenzial vorliegt. Schließlich muss ebenfalls ein Scheduling-Verfahren entwickelt werden, um mit Multipath-Routen oder opportunistischem Empfang umzugehen. Die vorhandenen Verfahren weisen verschiedene Defizite

auf, weshalb hierfür ein eigenes Verfahren entwickelt werden muss.

6 Design

Unter Berücksichtigung der genannten Problembeschreibung wird *hybrid ARQ with limited fragmentation* (HALF) implementiert. HALF zeigt unter Verwendung von Network Coding ein HARQ-Schema, dass mit Hop-By-Hop ACKs auf Batch-Ebene arbeitet und dabei partiell defekte Pakete durch Fragmentierung, sowie unabhängige Prüfsummen auf den Fragmenten weiterverwerten kann. Im Folgenden werden die verwendeten Konzepte als abstrakte Aspekte von HALF diskutiert und zu einander in Bezug gesetzt. In Abschnitt 7 wird dann erläutert, wie diese Konzepte in konkrete Software-Modulen umgesetzt werden.

6.1 Eliminierung von Bitfehlern durch Fragmentierung und Network Coding

Das Network Coding findet jeweils innerhalb eines Datenstroms statt. Diese Einschränkung wird angesichts der leichteren Implementierbarkeit und der Fokussierung auf die Überwindung durch Bitfehler erzeugter Paketverluste akzeptiert. Durch eine Codierung zwischen verschiedenen Strömen könnten potenziell weitere Durchsatzgewinne erzielt werden, wie beispielsweise mit COPE gezeigt wurde. Je eine Anzahl von Paketen eines Datenstroms wird zu einem Batch zusammengefasst und fragmentiert. Die daraus resultierende Menge an Fragmenten wird mittels *random linear network codes* (RLNC) in endlichen Feldern kombiniert und die codierten Fragmente wieder zu Paketen voller Länge zusammengesetzt. Die Fragmente und zu sendenden Pakete sind in ihrer Länge festgelegt. Quellpakete werden so mit Nullbits aufgefüllt, dass sie in eine ganze Zahl Fragmente zerlegt werden können. Zu sendende Pakete werden aus einer festen Zahl codierter Fragmente zusammengesetzt. Jedes Fragment erhält einen Header, der angibt, welche Originalfragmente mit welchen Koeffizienten darin enthalten sind. Diese Annotation der Daten mit ihren Koeffizienten ist an Lun u. a. (2006) angelehnt und wird in ähnlicher Form in MORE und MC^2 implementiert. Das Verfahren der statischen Koeffizientenbelegung nach Ho u. a. (2006) könnte dank des geringeren Overheads effizienter sein, es ist jedoch nicht offensichtlich, wie es auf verlustbehaftete Kanäle und insbesondere für HARQ adaptiert werden kann. Empfängt ein Knoten genug innovative Pakete, so kann er durch gauss'sche Elimination die Originalfragmente wieder decodieren und ausliefern.

Dies geschieht jedoch ausschließlich am Zielkonten des Datenstroms. Die Zwischenknoten prüfen lediglich anhand der in den Fragment-Headern übermittelten Koeffizienten, ob eine Decodierung möglich wäre. Die Recodierung findet bei den Zwischenknoten auf den empfangenen codierten Fragmenten als Grundmenge statt. Dies ist möglich, da Addition und Multiplikation auf endlichen Feldern transitiv und distributiv sind. Insgesamt kann auf diesem Weg auch aus stark von Bitfehlern betroffenen Frames noch nützliches Datenmaterial extrahiert werden. Da Bitfehler, wie oben gezeigt, gehäuft auftreten, ist zu erwarten, dass auch defekte Frames viele unbeschädigte Fragmente enthalten, die hier verwendet werden können. Durch das Scheduling wird versucht sicherzustellen, dass ein Informationsgefälle zwischen Sender und Empfänger besteht und nichtinnovative Pakete selten sind (vgl. Ho u. a. 2006). Die Zahl der Fragmente pro Batch, die Zahl der Fragmente pro gesendetem Paket und die Länge der Fragmente sind konfigurierbar.

6.2 Differenzierte Redundanzkontrolle durch Hybrides ARQ

Das ARQ wird mittels Hop-By-Hop STOP-Paketen implementiert. Dabei sendet ein Knoten nach der erfolgreichen Decodierung eines Batches oder bei Zwischenknoten der Feststellung, dass dies möglich wäre, sofort ein spezielles STOP-Paket an seinen Vorgänger auf der Route. Im Gegensatz zu anderen Systemen, wie beispielsweise MORE, wird diese Lösung gegenüber der End-To-End-Methode bevorzugt, weil damit eine bessere Nutzung der räumlichen Diversity möglich ist.

Im Gegensatz dazu wird bei der End-To-End-Methode das STOP erst gesendet, wenn der Zielknoten den gesamten Batch decodiert hat. Dieses STOP reist dann die gesamte Route zurück und der Quellknoten fängt erst mit dem nächsten Batch an, wenn das STOP ankommt. In der Zwischenzeit werden von den Zwischenknoten auf der Route nichtinnovative Pakete gesendet, die nicht mehr vom Zielknoten empfangen oder verwertet werden können. Die so verwendeten Sendemöglichkeiten werden demnach mit überflüssiger Redundanz verschwendet.

Der Hop-By-Hop-Ansatz vermeidet dies, indem der Quellknoten schon früher mit dem nächsten Batch auf der selben Route anfangen kann, und somit die entstehenden Sendemöglichkeiten produktiv genutzt werden. Hier findet eine feingliedrigere Kontrolle der Redundanz statt.

Dieses einfache Protokoll kann natürlich nicht mit völlig beliebigen Multipath-Routen umgehen. Es wird davon ausgegangen, dass alle Knoten, die an einer Übertragung beteiligt sind, linear geordnet sind. In Absatz 9 wird eine mögliche Erweiterung des Protokolls auf beliebige Multihop-Routen vorgestellt. HARQ wird hier im Zusammenhang

mit Scheduling gedacht. Deshalb werden die Details im Abschnitt 6.4 vorgestellt.

6.3 Erkennung von Bitfehlern

Die Datenübertragungsschicht verwirft standardmäßig Frames, die eine CRC-Überprüfung auf Frameebene nicht bestehen. Bei diesen Frames ist also die FEC fehlgeschlagen. Trotzdem kann ein solcher Frame, wie die Bitfehlermessungen zeigen, große Bereiche ohne defekte Bits enthalten. Um diese zu verwenden, wird die Datenübertragungsschicht angewiesen, Frames mit Bitfehlern nicht zu verwerfen, sondern trotzdem als Pakete auszuliefern. Das funktioniert nicht mir jedem Gerät und Treiber. Insbesondere der MADWIFI-Treiber für Atheros-Chips beherrscht jedoch dieses Verfahren. Um den genauen Ort der Bitfehler festzustellen, wird die bereits erläuterte Fragmentierung genutzt. Jedes Fragment wird mit einer eigenen CRC ausgestattet und beim Empfang separat geprüft. Je mehr Fragmente erzeugt werden, desto weniger gute Bits müssen verworfen werden, desto mehr Overhead entsteht jedoch auch durch die Koeffizienten im Header jedes Fragments.

Dieses Verfahren wurde vor allem gewählt, da es auf verfügbarer IEEE 802.11 Hardware nicht möglich ist, die Modulation und Demodulation einzelner Bits oder Symbole zu beeinflussen. Wäre dies möglich, so stünde eine große Bandbreite anderer Verfahren zur Verfügung, die noch weniger korrekt empfangene Bits verwerfen (vgl. Katti u. a. 2007; Pu u. a. 2008). Allerdings legen die Erkenntnisse über die Häufung der Bitfehler nahe, dass es reicht, einen Frame in relativ große Fragmente aufzuteilen, um den größten Teil der korrekt empfangenen Bits zu rekonstruieren.

6.4 Scheduling

Es wird ein einfaches eigenes Scheduling-Schema, *really unfair batch scheduling* (RUBS) entwickelt, das auf allgemeinen Multipath-Routen zwar nicht optimal wäre, die vorhandenen *dynamic source routing* (DSR)-Routen aber gut nutzen kann. In Abschnitt 9 wirden eine Abwandlung von RUBS für allgemeine Routen diskutiert. RUBS ist zunächst als temporäre Lösung zu verstehen, die gerade ausreichende Funktionalität bietet, um die wichtigeren Aspekte des HALF-Systems zu demonstrieren.

RUBS baut dabei auf einigen Erkenntnissen und Annahmen auf, die bei der Analyse der anderen Ansätze bestimmt wurden. Für die allgemeine Notwendigkeit eines Scheduling werden zwei häufig zwei Argumente vorgebracht. Eines davon ist die Idee, dass durch gleichzeitige Regelung der Senderate bei mehreren Knoten die Zahl der von ver-

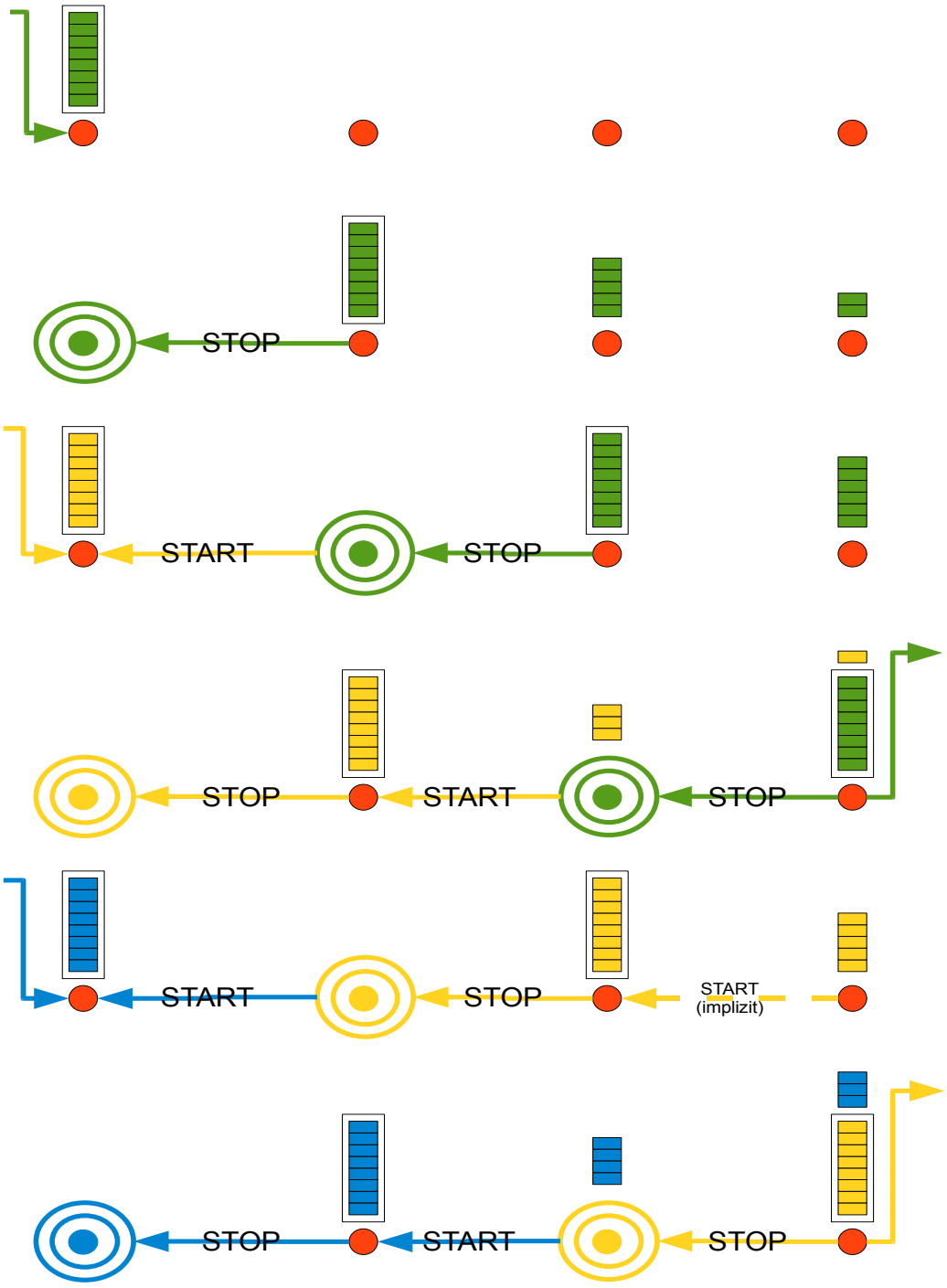


Abbildung 7 — Illustration des RUBS-Protokolls

steckten Knoten verursachten Kollisionen reduziert werden kann (vgl. Lee u. a. 2007). Bei fehlendem Einfluss auf die Wahl der Sendezeitpunkte ist das Verfahren hier jedoch nur mäßig effektiv, wie sich ebenfalls aus den Ergebnissen von Lee u. a. (2007) ergibt. Können zum Beispiel zwei kollidierende Sender ihre jeweiligen Signale nicht per Kanalprüfung identifizieren und regelt der Scheduling-Algorithmus zudem beide auf 50% ihrer maximalen Senderate herunter, so passiert Folgendes: Ein Viertel der Zeit bleibt das Medium ungenutzt, ein Viertel der Zeit kollidieren die Sender und die Hälfte der Zeit können Daten übertragen werden. Leider betrifft eine Kollision jedoch nicht nur den direkt kollidierenden Bereich, sondern kann bei mindestens einem Paket auch den Header zerstören. Ein Paket mit defektem Header muss vollständig verworfen werden. Der Prozentsatz der Zeit, zu der einer der beiden Sender kollisionsfrei arbeiten kann, sinkt also weiter. Lee u. a. (2007) stellen in diesem Fall eine jeweilige Durchsatzrate von etwa einem Fünftel der theoretischen Kanalkapazität fest, also wird der Kanal insgesamt zu nur 40% sinnvoll genutzt. Durch Hinzufügen mehrerer Sender sinkt zwar die Zeit, die das Medium ungenutzt bleibt, gleichzeitig erhöhen sich jedoch die Kollisionsmöglichkeiten. Aus diesem Grund wird hier nicht versucht, das Problem der Kollisionen mittels Senderatenkontrolle zu lösen.

Das zweite Argument ist die schon ausführlich dargestellte Durchsatzoptimierung auf Multipath-Routen unter Berücksichtigung potenziell unbeschränkter inkrementeller Redundanz. Dieses Argument wurde schon als stichhaltig erkannt. Eine Regelung der Senderate eines Knotens kann also nur dazu dienen, den Anteil eines Batches zu steuern, den ein Knoten in einer bestimmten Zeit übertragen soll. Da der sendende Knoten angesichts zufällig verteilter Sendemöglichkeiten und Paketverluste jedoch nicht wissen kann, wann dieser Anteil übertragen ist, ist dieses Verfahren auch hierfür ineffizient, falls es zu wenig Rückmeldungen gibt. Je höher die Varianz des Paketverlustes, desto unwahrscheinlicher ist es, dass beispielsweise eine auf dem mittleren Paketverlust beruhende Paketrate in begrenzter Zeit tatsächlich zur Übertragung der gewünschten Datenmenge führt. Bei RUBS werden daher, im Gegensatz zu beispielsweise MORE, explizite Empfangsbestätigungen - *STOP* - und eine explizite, möglichst vollständige Zuweisung des Mediums - *START* - favorisiert. Nicht zufällig kann hier das selbe STOP wiederverwendet werden, das auch der Realisierung des HARQ dient.

RUBS ist folgendermaßen aufgebaut. Jeder Batch wird immer nur von einem Knoten gleichzeitig gesendet. Sendet ein Knoten einen Batch so sendet weder sein Vorgänger noch sein Nachfolger einen Batch aus dem gleichen Datenstrom. Der Vorgänger und Nachfolger wären in einer gemeinsamen Kollisionsdomäne mit dem Sender und somit

ist es unnötig, sie senden zu lassen. Der Knoten, der sendet, sendet dafür so viel wie möglich. Kann ein Nachfolger einen Batch vollständig decodieren, so sendet er STOP, wie schon beschrieben. Bekommt ein Knoten STOP für einen Batch, sendet er START für den nächsten Batch aus dem selben Datenstrom an seinen Vorgänger. Gleichzeitig stellt er jegliches Senden von Batches aus diesem Datenstrom ein. Bekommt ein Knoten START, so fängt er an, den entsprechenden Batch zu senden, sobald er ihn vollständig empfangen hat. Da ein Zwischenknoten erst anfängt Daten zu senden, nachdem er selbst STOP gesendet hat, ist es nach Lun u. a. (2006) höchst unwahrscheinlich dass er - oder analog irgendein anderer Zwischenknoten - nichtinnovative Pakete generiert - es sei denn, er hat ein für ihn gedachtes STOP nicht bekommen.

Batches sind dabei die kleinsten Einheiten des Scheduling zwischen Flüssen. Für dieses Scheduling wird ein einfaches Queue-Verfahren verwendet. Ein neuer Batch stellt sich hinten an. Der erste Batch in der Reihe wird ununterbrochen gesendet, bis das STOP eintrifft.

Die allgemeine Arbeitsweise des HARQ- und Scheduling-Protokolls wird in Abbildung 7 illustriert. Zunächst erhält der erste Knoten ganz links eine Reihe Pakete zur Weitervermittlung über zwei weiterleitende Knoten an den Empfänger ganz rechts. Diese werden in Fragmente aufgeteilt und zu einem Batch (grün) gruppiert. Linearkombinationen dieses Batches werden sofort gesendet, denn für den ersten Batch aus einem Strom ist kein START notwendig. Der erste weiterleitende Knoten stellt nach einer Weile fest, dass er den Batch decodieren könnte und somit vollständig erhalten hat. Er sendet daraufhin STOP und fängt seinerseits an zu senden. In der Zwischenzeit haben die anderen Knoten durch opportunistischen Empfang schon Teile des Batches erhalten. Deshalb muss der nun sendende Knoten nicht den ganzen Batch übertragen, sondern nur soviel, wie dem nächsten Knoten noch fehlt. Anschließend bekommt er selbst ein STOP und sendet umgehend ein START an die Quelle, um den nächsten Batch (gelb) anzufordern. Dieses Verfahren kann beliebig lange fortgesetzt werden.

6.5 Integration in das BRN-Framework

Das BRN-Framework stellt verschiedene nützliche Dienste zur Verfügung, die als Umgebung für HALF genutzt werden können. So sind die verteilte Vergabe von IP-Adressen, das Auffinden von Knoten mittels ARP, die Doppelfunktion von Knoten als STA im BRN und AP für periphere Geräte sowie verschiedene weitere Funktionen bereits fertig implementiert und nutzbar. Aus diesem Grund wurde HALF ebenfalls als Teil des BRN-Frameworks implementiert. Da dieses Framework wiederum auf dem *click* Router-

Framework aufbaut, wurde auch HALF als eine Ansammlung von Click-Elementen implementiert. Der Vorteil der Verwendung von Click gegenüber anderen Umgebungen ist, dass Click sowohl als Modul des Linux-Kernels auf echter Hardware als auch zur Simulation in NS2 oder JiST/SWANS verwendet werden kann, ohne dass der Quellcode angepasst werden muss.

Als Routing-Protokoll kommt bei HALF DSR zum Einsatz. Diese Wahl fand hauptsächlich aus pragmatischen Überlegungen heraus statt. Es sollte vermieden werden, ein eigenes Routing-Protokoll zu schreiben und DSR ist im BRN-Framework schon integriert. Die durch DSR generierten Routen werden dabei jedoch opportunistisch verwendet. Jeder Knoten kann also Pakete verwenden, die für seine Vorgänger auf der Route bestimmt waren. Klar ist, dass hier eine Einschränkung stattfindet. Die von DSR generierten Routen sind linear geordnet. Es gibt keine parallelen Pfade. Jegliche opportunistische Funktionalität ist also dadurch bedingt, dass Nachfolgerknoten des Senders ausreichend nah zusammen stehen. Diese Einschränkung wurde in Kauf genommen, da keine passende Implementation eines ursprünglich für opportunistisches Routing gedachten Protokolls zur Verfügung stand. DSR ist allerdings hinsichtlich der minimalen Metriken zwischen Knoten auf der Route konfigurierbar, so dass *dichte* Routen mit vielen opportunistischen Empfangsmöglichkeiten generiert werden können.

Der Protokollablauf stellt sich insgesamt folgendermaßen dar. Empfangene Pakete werden hardwareseitig demoduliert. Von den HALF-Elementen werden sie in Fragmente zerteilt. Anschließend werden sie mittels CRC-Prüfsummen auf Fehler geprüft und schließlich der Decodierung oder Weiterleitung zugeführt. Umgekehrt werden zu sendende Pakete zuerst durch die DSR-Elemente geschickt und so mit einem Routen-Header versehen, dann codiert, mit Prüfsummen versehen und schließlich hardwareseitig moduliert. Das HALF-Protokoll findet also auf der Sicherungsschicht statt.

6.6 Simulationsumgebung

HALF wurde mittels Simulation in der JiST/SWANS-Simulationsumgebung getestet. Zu diesem Zweck wurden verschiedene Bitfehlermodelle implementiert. Sie richten sich nach der, aus der SNR hergeleiteten, Bitfehlerrate und sollen eine Eingrenzung der real zu erwartenden Bedingungen darstellen.

Uniform erzeugt gleichverteilt über jeden Frame Bitfehler. Jedes Bit in einem Frame ist mit der gleichen Wahrscheinlichkeit defekt. Dies ist die pessimistischste Variante, da hier mit Fragmentierung kein großer Gewinn zu erwarten ist.

Accumulated rechnet zuerst, analog zu Uniform aus, wie viele Bitfehler in einem Frame auftreten werden. Diese werden dann jedoch nicht über das ganze Paket verteilt, sondern alle hintereinander, beginnend an einer gleichverteilt ausgewählten Startposition angorndet. Accumulated ist die optimistischste Variante. Hier bleibt der größtmögliche Teil eines Frames von Bitfehlern unberührt und die Fragmentierung kann ihr volles Potenzial entfalten.

MaskedPackets nutzt gemessene Bitfehler als Maske für simulierte Bitfehler. Aus den bei Messungen gesendeten originalen und den empfangenen fehlerhaften Frames werden dabei Masken generiert, die die Positionen der Bitfehler enthalten. Diese werden nach gemessener SNR und Länge der Frames sortiert. Bei jeder simulierten Einzelübertragung wird aus dem Pool der Masken gleichverteilt eine zu Länge und SNR passende ausgewählt und auf den simulierten Frame angewandt. Dieses Modell scheint das realistischste, ist jedoch wegen der Ungenauigkeit der Messungen, insbesondere was die SNR angeht, nur unter Vorbehalt aussagefähig.

None schließlich behält die bisherige Funktionalität bei. Auch hier wird aus SNR und Modulationsverfahren die Bitfehlerrate hergeleitet. Anschließend wird die Wahrscheinlichkeit berechnet, dass der fragliche Frame keinen Bitfehler enthält. Anhand dieser Wahrscheinlichkeit wird entschieden, ob der Frame ausgeliefert oder unterdrückt wird.

7 Architektur

7.1 Click-Elemente

Im Folgenden wird der Aufbau des HALF-Protokolls als Ansammlung von Click-Elementen beschrieben. Click[3] ist ein in C++ geschriebenes Framework zur Implementation von Netzwerkprotokollen. Es lässt sich einerseits als Modul des Linux-Kernels und so beispielsweise auf AP-Hardware, die mit Linux betrieben wird, verwenden. Auf verschiedenen kommerziell verfügbaren Geräten wurde dies im Rahmen anderer Projekte schon erfolgreich getestet. Andererseits kann Click als Bibliothek für Userspace-Anwendungen kompiliert und so in eine Simulationsumgebung integriert werden. Entsprechende Adaptierungen liegen bereits für JiST/SWANS und NS2 vor. Click stellt verschiedene häufig benötigte Bausteine, wie beispielsweise Queues oder Paketzähler als Elemente bereit.

[3] siehe http://read.cs.ucla.edu/click/

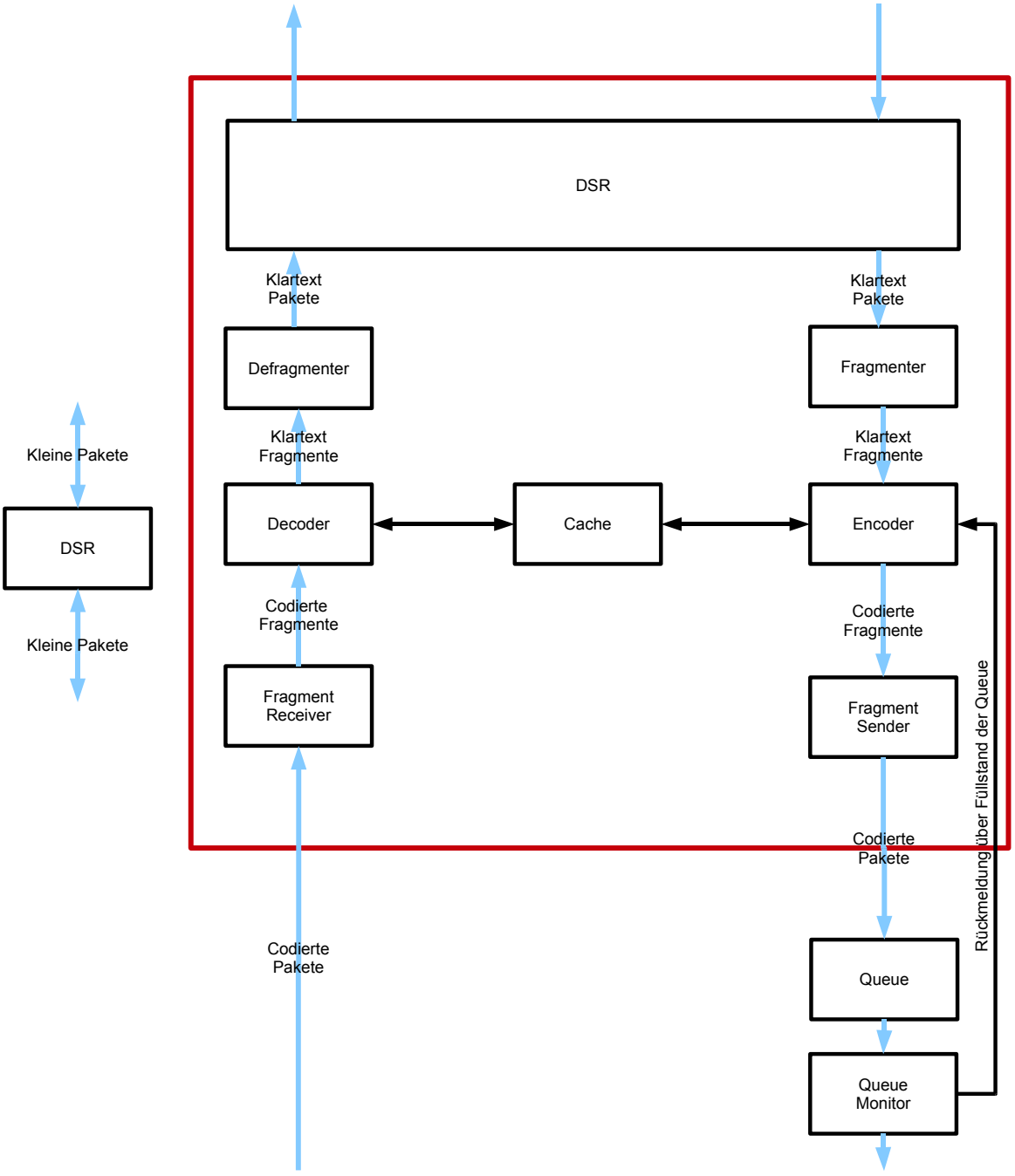


Abbildung 8 — Beziehungen der Click-Elemente

Verschiedene bekannte Netzwerkprotokolle, wie beispielsweise Ethernet sind ebenfalls schon vorimplementiert. Eigene Protokolle können in Form von Click-Elementen definiert werden. Ein Element ist eine in C++ geschriebene, von click::Element abgeleitete Klasse, die über Ein- und Ausgabeports Pakete empfangen oder absenden kann. Pakete sind im Wesentlichen Speicherpuffer, die von den Elementen auf verschiedene Arten geschrieben, gelesen, vergrößert, verkleinert oder annotiert werden können. Die Ports der Elemente können dann in einer Click-Konfiguration miteinander verbunden werden, was bei Ausführung der Konfiguration zu einer entsprechenden Übermittlung der Pakete führt. Mehrere, potenziell verbundene, Elemente können ebenfalls mittels einer Click-Konfiguration zu einem Superelement zusammengefasst werden, für das wieder Ein- und Ausgabeports definiert werden können.

7.1.1 Überblick über BRN und DSR

Im Rahmen des BerlinRoofNet-Projektes (BRN) wurden verschiedene Netzwerkkomponenten als Click-Elemente implementiert. Das IAPP-Superelement sorgt dafür, dass ein Knoten gegenüber einfachen Stationen nur als vollwertiger AP auftritt, mit kompatiblen anderen BRN-Knoten aber mittels BRN-eigenen Protokollen kommuniziert. Eine *diststributed hash table* (DHT) steht als gemeinsame Datenstruktur aller verbundenen BRN-Knoten zur Verfügung. Diese wird von einem verteilten DHCP-Service genutzt, um eindeutige IP-Adressen zu verteilen. Umgekehrt gibt es einen DHT-basierten, verteilten ARP-Service, um diese in MAC-Adressen aufzulösen. Der für diese Arbeit wichtigste Teil des BRN-Frameworks ist jedoch das DSR-Superelement. Es implementiert das DSR-Protokoll, angelehnt an Johnson und Maltz (1996) und arbeitet auf der Ebene der Sicherungsschicht.

7.1.2 Integration in die BRN-Umgebung

Die Network Coding Elemente sind als Erweiterung des DSR-Superelemnts gedacht. Das kombinierte Superelement aus HALF und DSR soll dabei als Drop-in-Ersatz für DSR funktionieren. In der Praxis hat sich jedoch herausgestellt, dass die Codierung der verschiedenen, von anderen Bestandteilen des BRN-Frameworks erzeugten, Protokollnachrichten aufgrund ihres geringen Volumens ungünstig ist. Daher wurde die Konfiguration der BRN-Umgebung so angepasst, dass Protokollnachrichten über eine eigene Instanz des unveränderten DSR-Superelements, und echte Datenpakete durch das kombinierte Superelement aus DSR und HALF geroutet werden.

7.1.3 Interner Aufbau des HALF-Superelements

Das HALF-Superelement besteht neben den unveränderten DSR-Elementen aus Paketcache, Encoder, Decoder, sowie einigen Elementen, die für die Fragmentierung zuständig sind und einigen Hilfselementen. Abbildung 8 zeigt die Beziehungen der Elemente. Blaue Pfeile bezeichenen Verbindungen zwischen Ports, über die Click-Pakete ausgetauscht werden können. Schwarze Pfeile bezeichnen C++-Methoden, die dem Element am Anfang des Pfeils vom Element am Ende des Pfeils zur Verfügung gestellt werden. Sowohl Encoder als auch Decoder können so auf den gemeinsamen Cache zugreifen. Abbildung 9 zeigt den Fluss der Kontrollpakete. Fragmente die nicht für den empfangenden Knoten bestimmt sind, sondern entsprechend der Routinginformationen weitergeleitet werden müssen, werden nie decodiert und nach oben weitergeleitet. Deshalb können die DSR-Elemente die an diese Pakete annotierten Linkmetriken nicht inspizieren. Um die Linktabelle trotzdem mit Metriken aus neu empfangenen Paketen zu aktualisieren, werden Route-Updates, kleine Pakete, die nur aus DSR- und HALF-Headern bestehen, durch eine separate Instanz des SrcForwarder-Elements aus DSR geleitet.

7.1.4 Paketcache

Der Paketcache verwaltet einen Pool von Batch-Objekten, die wiederum je mehrere Fragmente enthalten. Es gibt dafür drei Batch-Klassen: SourceBatch für Pakete, die lokal oder bei einer der assoziierten Stationen erzeugt wurden, ForwardBatch für solche, die weiterzuleiten sind und DestBatch für Pakete, die an die lokale oder an eine der assoziierten Stationen ausgeliefert werden sollen. Der Cache bietet Funktionalität, um codierte und Klartextfragmente einzufügen und gibt die dazugehörigen Batches zurück. Er kann auch genutzt werden, um Batches nach Nummer zu suchen. Ist ein Batch entweder am Decoder oder am Encoder vollständig abgearbeitet, so kann dies dem Cache mittels entsprechender C++-Methoden mitgeteilt werden. Ist ein Batch auf beiden Seiten abgearbeitet, so wird er gelöscht. Erhält der Cache ein codiertes Fragment mit bisher unbekannter Batchnummer oder ein Klartextfragment mit unbekanntem Ziel, so legt er einen entsprechenden Batch neu an.

7.1.5 Coding

Die Batchklassen enthalten Methoden um codierte Fragmente aus Linearkombinationen der vorhandenen Fragmente zu erzeugen. Andere Methoden können genutzt werden, um mittels gauss'scher Elimination Klartextfragmente als Lösungen eines linearen Glei-

chungssystems über die erhaltenen codierten Fragmente wiederherzustellen. Als Implementation der hierfür benötigten Operationen auf endlichen Feldern wird die *Fast Galois Field Arithmetic Library in C/C++*[4] von James S. Plank verwendet. Die Codierungs- und Decodierungsmethoden der vom Cache zurückgegebenen Batches werden von Decoder und Encoder genutzt, um die jeweils nötigen Fragmente zu erzeugen. Jedes Fragment ist dabei eine Instanz der vordefinierten Klasse *click::Packet* und kann somit über die jeweiligen Ports weitergegeben werden. Der Encoder speist demnach auf der einen Seite Klartext-Fragmente in den Cache ein und extrahiert aus den dafür erhaltenen Batches codierte Fragmente, die er an den FragmentSender weitergibt. Auf der anderen Seite speist der Decoder codierte Fragmente ein, für die er von den dafür erhaltenen Batches Klartextfragmente erhält, die er wiederum an den Defragmenter weitergibt.

7.1.6 Fragmentierung

Die Fragmentierung und Defragmentierung wird durch die Elemente Fragmenter, FragmentSender, FragmentReceiver und Defragmenter geregelt. Der Fragmenter teilt eingehende Klartextpakete in Fragmente von fester Größe auf, die nach Bedarf mit Nullbytes aufgefüllt werden. Der FragmentSender konkateniert eine feste Zahl Fragmente in ein codiertes Paket und ergänzt Fragment- und und Paketheader entsprechend. Ein eingehendes codiertes Paket wird vom FragmentReceiver wieder in seine Fragmente aufgeteilt. Der Defragmenter schließlich rekonstruiert das Originalpaket aus Klartextfragmenten. Die Information über Paketgrenzen im Klartext wird über Click-Paketannotationen und im Paketheader übermittelt. Die Parameter der Fragmentierung, also die Länge der Fragmente und die Zahl der Fragmente pro Paket, sind zwar für eine jeweilige Instanz der Elemente festgelegt, können aber im Allgemeinen konfiguriert werden. Auf diese Art können Klartextpakete beliebiger Größe zu codierten Pakete beliebiger anderer Größe kombiniert werden und so an die optimale Paketlänge für die jeweilige Umgebung angepasst werden. Das Verfahren hat jedoch auch einen Nachteil. Eine ungünstige Relation von Fragmentzahl, eingehender und ausgehender Paketgröße erzeugt unter Umständen unnötige Redundanz. Sollen beispielsweise je 32 Fragmente in einem Batch zusammengefasst werden und wird ein Klartextpaket aufgrund seiner typischen Größe und der festgelegten Fragmentgröße in je 5 Fragmente aufgeteilt, so wird jeder Batch unweigerlich aus 30 Fragmenten bestehen, da ein Batch nur ganze Pakete enthalten kann. Werden nun je 4 Fragmente zu einem codierten Paket zusammengefasst, so müssen insgesamt mindestens 32 Fragmente übertragen werden, um die 30 Klartextfragmente zu rekonstruieren.

[4] siehe http://www.cs.utk.edu/~plank

Hinzu kommen die aufgefüllten Nullbytes, falls Fragmentgröße und Klartext-Paketgröße ungünstig zueinander dimensioniert sind.

7.1.7 Scheduling

Die Implementation des START-STOP-Protokolls ist als Teil des Encoders und Decoders realisiert. Nachdem die eigentliche Codierung von den Batch-Klassen selbst vorgenommen wird, haben Encoder und Decoder faktisch kaum andere Aufgaben, daher war dies die natürliche Wahl. Der Encoder hält eine Liste der momentan sendeberechtigten Batches. Wird ein STOP-Paket für einen dieser Batches empfangen, so wird er aus der Liste gestrichen und ein START-Paket für den nächsten Batch an den Vorgängerknoten gesendet. Wird ein START-Paket für einen Batch empfangen, der noch nicht in der Liste steht, so wird dieser am Ende der Liste hinzugefügt. Gesendet werden grundsätzlich Pakete aus dem ersten Batch in der Liste. Da die Generierung codierter Pakete eine nicht zu vernachlässigende Zeit in Anspruch nimmt, werden diese proaktiv erzeugt, bevor eine Sendemöglichkeit vorliegt. Um zu bestimmen, wie viele Pakete jeweils nötig sind, wird ein zusätzliches Element zur Überwachung der Sendequeue eingesetzt. Immer wenn ein Paket aus der Queue gesendet wird, wird der Encoder über die Länge der Queue benachrichtigt und kann ein neues Paket generieren. Der Decoder erkennt, wenn ein Batch vollständig empfangen wurde und sendet dementsprechend STOP-Pakete. START- und STOP-Pakete werden über eine separate, priorisierte Sendequeue gesendet. Zusätzlich wird dem Encoder die Möglichkeit gegeben, Pakete aus der regulären Queue zurückzuziehen, wenn ein STOP eintrifft. Dies geschieht, um die Zahl der gesendeten nichtinnovativen Pakete zu minimieren. START und STOP werden im Unicast-Modus unter Verwendung von Senderückmeldungen (*TX Feedback*) gesendet, da hier absolute Sicherheit über den Empfang erzielt werden muss. Kommt ein STOP-Paket nicht beim vorgesehenen Empfänger an, so wird dieser nie aufhören, den entsprechenden Batch zu senden, was dazu führt, dass der Datenstrom zum Erliegen kommt. Kommt umgekehrt ein START nicht an, so wird der Knoten nie anfangen, den nächsten Batch zu senden, was zum selben Ergebnis führt. Um dies unter allen Umständen zu vermeiden, wird dem Decoder eine fehlschlagende Übertragung mittels einer Senderückmeldung signalisiert und daraufhin ein neues START- oder STOP-Paket generiert. Die Datenpakete werden im Gegensatz dazu im Broadcast-Modus gesendet, um die paketweisen ACKs zu vermeiden. Dies führt, wie im Abschnitt 3.1 erläutert, zu einer Benachteiligung der START- und STOP-Pakete in den niedrigeren Bereichen der Sicherungsschicht, da die Datenpakete mit konstantem Backoff arbeiten, während START und STOP exponenti-

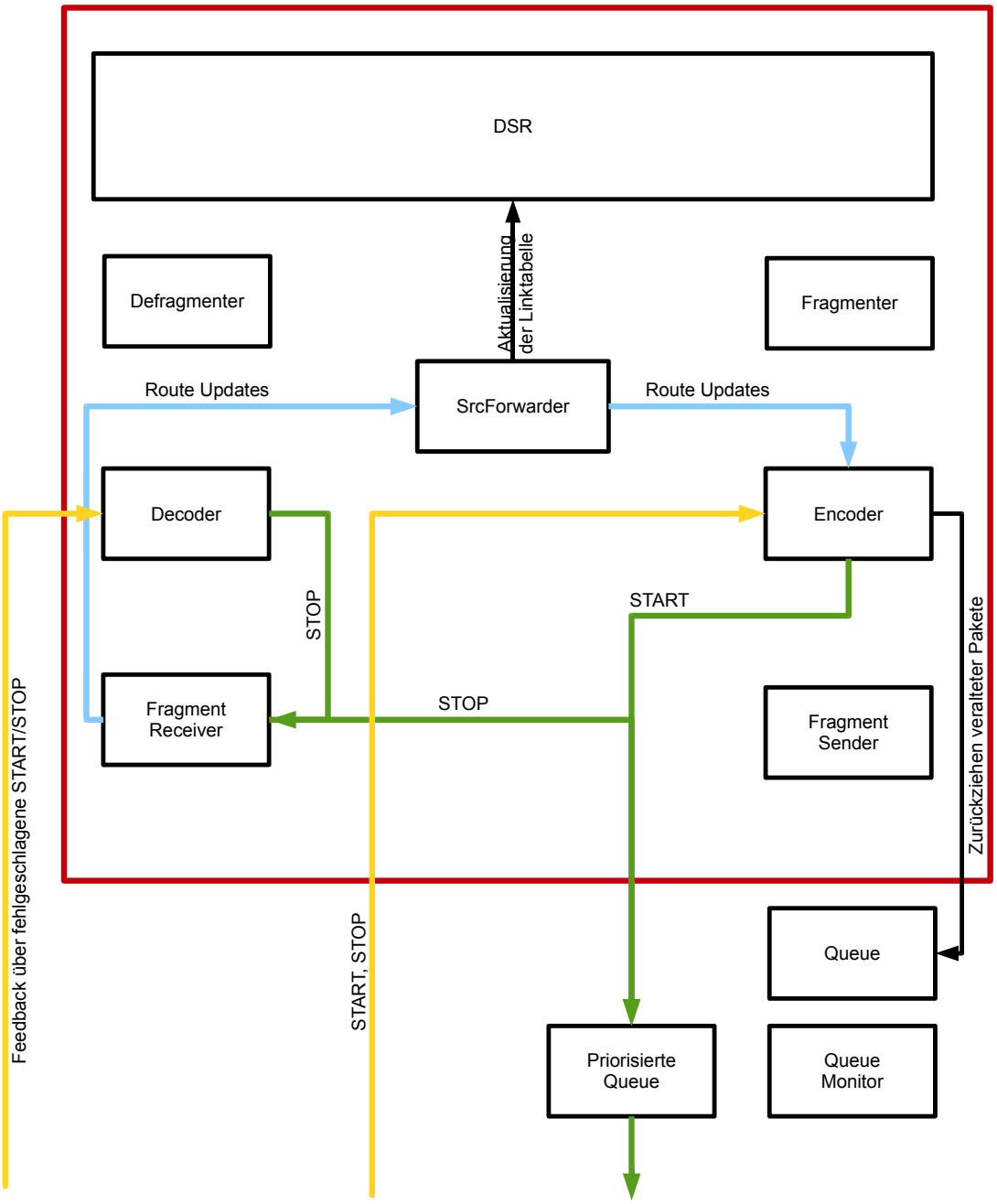


Abbildung 9 — Erzeugung und Verteilung der Kontrollpakete

elles Backoff verwenden. Dies wiederum führt zu verspätet eintreffenden STOP-Paketen und somit zu vielen nichtinnovativen Paketen. Um dieses Problem zu lindern, wurden zwei kleine Modifikationen am ursprünglichen Protokolldesign vorgenommen.

Erstens werden alle Knoten in den *promiscous* Modus gesetzt und empfangen somit auch START-Pakete, die für andere Knoten gedacht sind. Wird ein solches empfangen, so hält der Knoten sich mit weiteren Sendeoperationen solange zurück, wie der Empfänger des START mindestens braucht, um einen kompletten Batch zu übertragen. Dies verringert die Zahl der Stationen, die um das Medium konkurrieren und erhöht somit die Chance, dass ein wartendes STOP übertragen werden kann. Natürlich lässt dieses Verfahren das Medium auch eine Weile brachliegen, wenn der eigentliche Empfänger des START schon Teile des Batches kennt und somit weniger Zeit erforderlich ist, um den gesamten Batch zu übertragen. Dies erweist sich in den meisten Fällen als kontraproduktiv, daher wird diese Modifikation standardmäßig abgeschaltet.

Zweitens macht ein Knoten jedes Mal eine kurze Pause, wenn er so viele Pakete gesendet hat, wie mindestens nötig sind, um einen vollständigen Batch an den nachfolgenden Knoten auf der Route zu übertragen. In dieser Pause kann das Medium für ein eventuelles STOP genutzt werden. Sind Fragmente verloren gegangen und wird daher kein STOP gesendet, so wird auch hier Kapazität verschwendet.

Beide Modifikationen sind optional und können entfallen, wenn durch direkte Beeinflussung der Backoffzeiten die inhärente Benachteiligung der Unicast-Pakete gegenüber Broadcast-Paketen eliminiert werden kann. Eine Erweiterung des IEEE 802.11 Standards, 802.11e, führt die Möglichkeit ein, Frames mittels verschieden langer Backoff-Zeiten zu priorisieren (vgl. 802.11 2007). Damit könnte das zu Grunde liegende Problem gelöst werden. Leider liegt bisher keine passende Implementation dieses Verfahrens vor.

7.2 Bitfehlermodelle

7.2.1 Überblick über JiST/SWANS

JiST [5] ist eine in Java geschriebene Simulationsumgebung. Sie nutzt die Java Virtual Machine und die Apache *byte code engineering* Bibliothek [6], um ereignisorientierte Simulationen zu erzeugen. Dabei werden nach der Kompilierung bestimmte Sequenzen des Java-Bytecodes umgeschrieben. Mit entsprechenden Methoden kann dann der Programmablauf unterbrochen und der Stack gespeichert, bzw. ein gespeicherter Stack

[5] siehe http://jist.ece.cornell.edu/
[6] siehe http://jakarta.apache.org/bcel/

reaktiviert und der Programmablauf an der selben Stelle fortgeführt werden. Mit Hilfe dieser Methoden wird ein Ereigniskalender aufgebaut.

SWANS baut auf JiST auf und implementiert verschiedene Modelle und Protokolle zur Simulation von Drahtlosnetzwerken. Insbesondere sind hier ein Feld- und verschiedene Radiomodelle verfügbar. Ein Feldmodell legt fest, mit welcher Energie ein Radiosignal, das an einem Knoten gesendet wird, an einem anderen ankommt. Das Feldmodell wird durch konfigurierbare Fading- und Pfadverlustmodelle ergänzt, die die Dämpfung und Verstärkung von Signalen simulieren. Ein Radiomodell nimmt vom Feldmodell berechnete Signale entgegen und entscheidet, ob ein Knoten den zugehörigen Frame empfangen kann. Die in SWANS enthaltenen Radiomodelle wurden bereits vor dieser Arbeit durch das Modell RadioAdditiveNoiseBER ergänzt. Dieses Modell geht von einem AWGN-Kanal aus, verrechnet also das eingehende Signal mit Hintergrundrauschen und Interferenzsignalen. So ist es in der Lage, die SINR für Frames zu berechnen, die zwischen simulierten Knoten übertragen werden. Die SINR wiederum wird verwendet, um unter Berücksichtigung des Modulationsverfahrens, die Bitfehlerrate zu errechnen. Diese wird dann als Kriterium für den Empfang eines Frames herangezogen, indem eine Zufallszahl zwischen 0 und 1 gezogen und mit der Wahrscheinlichkeit verglichen wird, dass kein Bit im Frame fehlerhaft ist. Im ursprünglichen Modell wird ein Frame verworfen, sobald er ein fehlerhaftes Bit enthält. Dieses Verhalten wurde als primitives Bitfehlermodell *None* beibehalten. Die Bitfehlermodelle ersetzen im Allgemeinen den Mechanismus zur Akzeptanz oder Ablehnung von Frames und allozieren zusätzlich Bitfehler in den Frames.

7.2.2 Realisierung der Bitfehlermodelle

Die Bitfehlermodelle wurden als Ergänzung des RadioNoiseAdditiveBER-Radiomodells für die JiST/SWANS-Simulationsumgebung geschrieben. Das Uniform-Modell generiert für jedes Bit in einem Frame eine gleichverteilte Zufallszahl zwischen 0 und 1. Liegt diese Zahl unter der Bitfehlerrate, so wird das Bit als defekt markiert. Dadurch entsteht eine Bitfehlermaske. Ausgehend von dieser Maske werden die entsprechenden Bits bei der Auslieferung an die MAC-Schicht geändert. Das Accumulated-Modell funktioniert weitgehend äquivalent. Die Maske wird hier vor der Anwendung noch einmal nachbearbeitet und alle Bitfehler in einem zusammenhängenden Bereich konzentriert. Grundlegend anders aufgebaut ist das MaskedPackets-Modell. Die gemessenen Bitfehlermasken werden hier direkt, ohne Rückgriff auf die modellgestützte Berechnung der Bitfehlerrate, verwendet. Dazu werden vor der Simulation die gemessenen Bitfehlermasken geladen und nach *received signal strength indication* (RSSI) und Länge sortiert. Die RSSI kann, ent-

sprechend der simulierten Hardware, aus der SINR berechnet werden. Aus diesem Grund kann hier zur Simulationszeit statt einer synthetisierten, eine gemessene Bitfehlermaske aus dem Vorrat der geladenen Masken gewählt werden. Diese Auswahl erfolgt gleichverteilt.

8 Simulationsergebnisse

Um die Vor- und Nachteile des HALF-Systems zu evaluieren wurden ausgiebige Simulationen durchgeführt. Im folgenden Abschnitt wird zunächst erörtert, nach welchen Fragestellungen vorgegangen wurde. Anschließend werden die Rahmenbedingungen der Simulationen beschrieben und schließlich die Ergebnisse präsentiert.

8.1 Evaluationskriterien

Die Hauptfrage, die mit den Simulationen erörtert werden soll, ist ob die Nutzung der fehlerhaften Pakete einen wesentlichen Vorteil gegenüber der bisherigen Praxis bringt. Dieser Vorteil wird in Form von End-to-End-Durchsatz an Paketen zwischen zwei Knoten beziffert. Zu diesem Zweck wird das HALF-Protokoll unter verschiedenen Bitfehlermodellen beobachtet. Zusätzlich wird betrachtet, inwiefern opportunistischer Empfang und Scheduling einen Durchsatzvorteil gegenüber klassischen Routing-Ansätzen bedeuten. Zu diesem Zweck wird HALF ohne Allokation der Bitfehler mit DSR verglichen. Als dritte Fragestellung wird untersucht, inwiefern das HALF-Protokoll die Latenz der Übertragung beeinflusst. Als Vergleichsobjekt dient auch hier DSR, da dieses Protokoll aufgrund des geringen Koordinationsaufwandes und der fehlenden protokollinhärenten Verzögerungen günstige Latenzeigenschaften aufweist. Zuletzt werden verschiedene HALF-Konfigurationen bezüglich der Menge der auftretenden nichtinnovativen Pakete verglichen. Jedes nichtinnovative Paket stellt im Rahmen des Network Coding eine verpasste Übertragungsmöglichkeit dar. Durch Einbezug weiterer oder anderer Frames kann nach Lun u. a. (2006) grundsätzlich ein Fragment erzeugt werden, das mit hoher Wahrscheinlichkeit für alle Empfänger innovativ ist. Insofern können viele nichtinnovative Pakete als Optimierungspotenzial angesehen werden.

8.2 Aufbau der Experimente

Alle Simulationen werden grundsätzlich in einer quadratischen Anordnung von 5 mal 5 Knoten durchgeführt. Das gesamte Feld ist 150m breit und lang, somit sind die Knoten

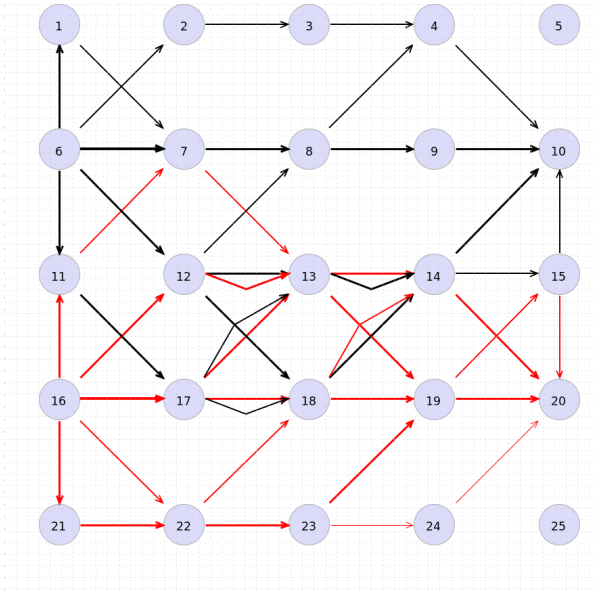


Abbildung 10 — Illustration der Route Selection mit DSR

in der Senkrechten und Wagrechten 37,5m, in der Diagonalen etwa 53m von einander entfernt. Dieser Abstand ist in etwa die Grenze, ab der ein fehlerfreier Empfang mit einer Bitrate von 54Mbps schwierig wird. Als Fadingmodell wird ein zeitlich schnell variierendes Punnoose-Rician-Fading genutzt. Der K-Faktor wird dabei auf 6dB gesetzt, wodurch eine relativ gute LOS-Verbindung simuliert wird. Die Fadinggeschwindigkeit beträgt 16m/s. Der Pfadverlust wird durch das LogDistance-Modell mit einem Exponenten von 2,8 modelliert. Dies entspricht einer städtischen Umgebung mit leichtem Shadowing. In diesem Szenario stehen für DSR eine große Anzahl von Routen zwischen jedem Knotenpaar zur Verfügung. Üblicherweise wird aber nur zwischen direkt benachbarten Knoten geroutet. Gleichzeitig ist damit zu rechnen, dass auf jeder Route ein gewisses Maß an opportunistischem Empfang möglich ist. Abbildung 10 illustriert die Verteilung der Knoten und zeigt verschiedene Routen zwischen den Knoten 6 und 10 sowie 16 und 20, die DSR in diesem Szenario wählen kann. Üblicherweise wird nun ein einzelner Datenstrom von Knoten 11 nach Knoten 15 erzeugt. Soll der Effekt der Interferenz näher betrachtet werden, so werden jedoch auch Experimente mit mehreren Datenströmen, wie in Abbildung 10, zu Rate gezogen. Bei jedem Experiment werden zunächst 80 Simulationssekunden lang die verschiedenen BRN-Protokolle laufen gelassen, damit sich die DHT und die Routentabellen aufbauen. Anschließend werden jegliche Link Probes unterbunden und 10 Simulationssekunden lang das eigentliche Experiment durchgeführt. Während dieser 10 Sekunden werden am Quellknoten mit konstanter Ra-

te pro Sekunde 3000 Pakete von 1460 Bytes Länge generiert und mittels UDP an den Zielknoten gesendet. Das reicht in jedem Fall, um die Verbindung auszulasten. In Experimenten mit DSR ist ein Routenwechsel in diesen 10 Sekunden möglich, da DSR aus verschiedenen Quellen abseits der Link Probes seine Linktabellen aktualisiert. Speziell in Gegenwart von Fading führt das im Allgemeinen zu besseren Ergebnissen als festgeschriebene Routen. In Experimenten mit HALF wird der Routenwechsel unterbunden, indem die erste Route für jeden Strom aufbewahrt und immer wieder verwendet wird. HALF kann auch mit wechselnden Routen auf Batchebene umgehen, es hat sich jedoch gezeigt, dass der opportunistische Empfang die meisten Routenwechsel unrentabel macht. Zudem müssen alle Pakete eines Batches auf der selben Route übertragen werden, was dazu führt, dass bei häufigem Routenwechsel auch häufig veraltete Routen verwendet werden. DSR hingegen kann prinzipiell jedes Paket auf einer eigenen Route übertragen und daher schneller auf Änderungen reagieren.

Variante	35 Experimente	100 Experimente
Standard	1555 Pakete	1724 Pakete
recvCorrupt	1765 Pakete	1706 Pakete

Tabelle 1 — Erwartungswert des Durchsatzes mit DSR bei zwei funktional gleichen Varianten des None-Modells

Jedes Experiment wird ungefähr 20 Mal mit unterschiedlichen Zufallsstartwerten durchgeführt, um den Einfluss von Zufallsvariationen auf das Ergebnis einzuschränken. Die Zufallsstartwerte beeinflussen vor allem das Fading und die Verteilung der Bit- und Paketfehler. Wirklich stichhaltige statistische Aussagen lassen sich bei dieser Zahl leider noch nicht treffen, wie ein Vorversuch zeigt. Im oben beschriebenen Aufbau wurden 70 Simulationen mit DSR ohne Allokation der Bitfehler, also mit dem None-Modell durchgeführt. Bei 35 davon wurden die als defekt erkannten Pakete trotzdem an den Click-Router ausgeliefert, der sie dann umgehend verwarf. Dies hatte den Nebeneffekt, dass zusätzliche Zufallszahlen generiert wurden. Damit wurde der Ablauf jeder einzelnen Simulation gegenüber der Version ohne Auslieferung der defekten Pakete modifiziert. Natürlich ist dies trotzdem keine funktionale Modifikation. Deshalb sollten die aggregierten Ergebnisse der beiden Varianten auf lange Sicht konvergieren. Leider ergaben sich für den Erwartungswert des Durchsatzes trotzdem signifikante Unterschiede, wie Tabelle 1 zeigt. Das Experiment wurde anschließend mit je 100 Instanzen der selben beiden Konfigurationen wiederholt. Der nun noch resultierende Unterschied im Erwartungswert des Durchsatzes ist deutlich kleiner. Es muss also davon ausgegangen werden, dass hier ei-

ne Zufallsverteilung vorliegt, die erst bei sehr vielen Experimenten sinnvolle aggregierte Ergebnisse zulässt. Leider sind die Bitfehlermodelle Uniform und Accumulated in der Berechnunge um Größenordnungen aufwändiger als das Basismodell None. Deshalb war es nicht möglich, eine ausreichend hohe Zahl von Experimenten für jedes untersuchte Szenario durchzuführen. Die starken Schwankungen treten vor allem bei den Durchsatzzahlen für DSR im Allgmeinen und für HALF bei mehreren parallelen Datenströmen, sowie bei den Latenzzahlen für DSR auf. Bei der Analyse werden diese Zahlen nach Möglichkeit nicht verwendet.

Prot.	#Str.	Bitfehler	Frag.	Durchs.	empf.	nichtinn.	opp.	Latenz
DSR	1	Accumulated	n/a	1696	12667	0	0	0.243
DSR	1	None	n/a	1693	12676	0	0	0.249
DSR	1	Uniform	n/a	1575	12018	0	0	0.240
DSR	3	None	n/a	1564	12793	0	0	0.305
HALF	1	Accumulated	2	6734	27026	2247	10353	0.100
HALF	1	None	2	5111	20524	1033	2427	0.141
HALF	1	Uniform	2	5207	20904	1366	2935	0.138
HALF	1	Accumulated	4	6702	26802	5422	14477	0.054
HALF	1	None	4	4613	18473	1790	2198	0.082
HALF	1	Accumulated	8	4483	17834	7372	12349	0.038
HALF	1	None	8	3347	13355	2204	1683	0.058
HALF	3	Accumulated	2	5708	22747	3402	10641	0.326
HALF	3	None	2	3897	15907	827	3125	0.496
HALF	3	Uniform	2	4050	16396	1006	3229	0.489
HALF	3	Accumulated	4	5955	24139	6012	14074	0.159
HALF	3	None	4	3402	13526	1392	2991	0.296
HALF	3	Uniform	4	3653	14700	1997	3114	0.282
HALF	3	Accumulated	8	4739	18516	9947	13908	0.098
HALF	3	None	8	2758	11158	2011	2287	0.190
HALF	3	Uniform	8	2987	12018	3360	3054	0.171

Tabelle 2 — Zusammenfassung der aggregierten Ergebnisse

Um einen, wenn auch groben, Überblick zu geben, werden hier dennoch alle Ergebnisse der verschiedenen Experimente in aggregierter Form präsentiert. Tabelle 2 charakterisiert die Art der Experimente auf der linken Seite und führt verschiedene aggregierte Resultate auf der rechten Seite auf. Auf der linken Seite wird in der ersten Spalte das verwendete Protokoll - HALF oder DSR, in der zweiten die Anzahl der parallelen Datenströme, in der dritten das verwendete Bitfehlermodell und in der vierten die Anzahl der Fragmente pro codiertem Paket aufgeführt. Die Anzahl der Fragmente pro Klartext-

Paket ist prinzipiell gleich der Anzahl der Fragmente pro codiertem Paket. Alle anderen Konfigurationen stellten sich als ungünstig heraus. Die Länge der codierten Pakete ist dementsprechend angepasst. Auf der rechten Seite ist in der fünften Spalte die mittlere Anzahl der insgesamt in den genannten 10 Simulationssekunden End-To-End übertragenen Pakete aufgeführt. Bei mehreren Strömen ist dies die Summe der für die verschiedenen Ströme übertragenen Pakete. In der sechsten Spalte wird die Zahl der insgesamt an allen Knoten während dieser Zeit empfangenen verwendbaren innovativen Pakete gezeigt. Ein Paket ist verwendbar, wenn der empfangende Knoten auf der Route des Pakets näher am Ziel der Route liegt als der sendende Knoten. Dies umfasst auch opportunistische empfangene Pakete. In der siebten Spalte steht die Anzahl der insgesamt an allen Knoten empfangenen nichtinnovativen Pakete, also prinzipiell verwendbare Pakete, die aber keinen Informationsgewinn für den empfangenden Knoten darstellen. In der achten Spalte ist die mittlere Zahl der insgesamt opportunistisch empfangenen verwendbaren Pakete aufgeführt. In der letzten Spalte schließlich wird die mittlere End-To-End Latenz der Übertragungen angegeben. Diese wird auf der Transportschicht gemessen. Die Zeit zwischen der Übergabe eines Pakets an das UDP-Protokoll und der Auslieferung des selben Paketes am Empfänger in Sekunden dient hier als Kriterium. Es werden bei den Durchsatz- und Empfangszahlen allgemein Pakete angegeben, obwohl natürlich auch der Empfang einzelner Fragmente möglich ist. Die angegebenen Paketzahlen sind aus der entsprechenden Gesamtzahl der Fragmente und der Zahl der Fragmente pro Paket synthetisiert. Dies geschieht zum Zweck der Vergleichbarkeit. Wenn also beispielsweise zwei aus je zwei Fragmenten bestehende Pakete empfangen werden und von diesen je ein Fragment verwertet werden kann, so wird dies als ein empfangenes und ein verworfenes Paket gezählt.

8.3 Durchsatzgewinne durch Verwendung korrupter Pakete

Die bereits vorgestellten theoretischen Überlegungen und Messergebnisse legen nahe, dass durch die Verwendung bitfehlerbehafteter Pakete mittels der erläuterten Fragmentierung ein deutlicher Gewinn gegenüber der üblichen Praxis erzielt werden kann. Um dies zu verifizieren wurden drei Experimente durchgeführt. Das HALF-Protokoll wurde jeweils im oben beschriebenen Szenario mit den Bitfehlermodellen None, Uniform und Accumulated simuliert. Die Erwartung ist, dass der Gesamtdurchsatz unter Verwendung von None am niedrigsten, bei Accumulated am höchsten ist. Uniform kann bei niedrigen Fehlerraten immer noch einen gewissen Gewinn gegenüber None bringen, da bei Gleichverteilung einzelne Fragmente von Bitfehlern unberührt bleiben können. Die Simulation

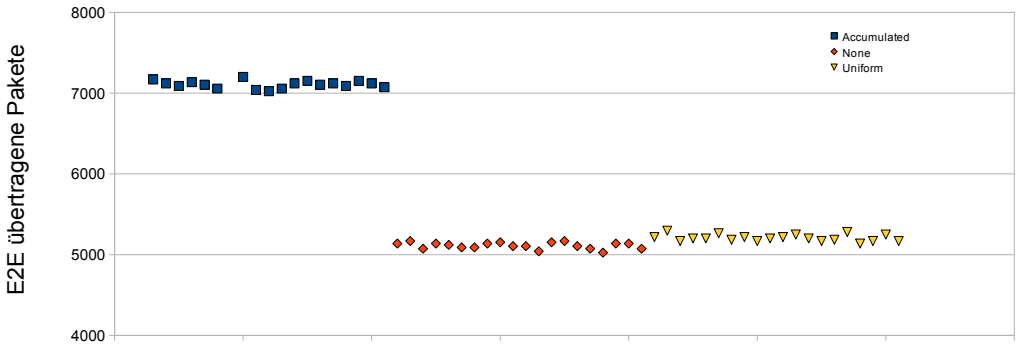


Abbildung 11 — Durchsatz mit None, Uniform und Accumulated unter Verwendung von HALF

bestätigt diese Annahme, wie Abbildung 11 zeigt. Als Gegenprobe wurden die selben Umgebungen mit DSR simuliert, das die korrupten Pakete verwirft. Für DSR macht es demnach keinen großen Unterschied, welches Bitfehlermodell verwendet wird, wie Abbildung 12 bestätigt. Hier liegen zwar die bereits erwähnten starken Zufallsschwankungen vor, das Vorexperiment stützt jedoch die Vermutung, dass bei allen drei Experimenten in etwa die selbe Verteilung vorliegt. Das Modell None geht, wie oben beschrieben, intern von der selben Verteilung der Bitfehler auf Pakete aus, wie Uniform und Accumulated. Die Fehler werden nur nicht konkret alloziert. Es wird demnach aus dem Experiment mit DSR klar, dass die Bitfehlermodelle in der selben Gesamtzahl an korrupten Paketen resultieren. Damit findet durch Gleichverteilung auf der einen und Konzentration der Bitfehler auf der anderen Seite tatsächlich eine Eingrenzung der möglichen Bitfehlerverteilungen innerhalb der Pakete statt. Die Verwendung korrupter Fragmente würde im vorgestellten Szenario also zwischen 2 und 40 Prozent Durchsatzgewinn gegenüber der bisherigen Praxis bringen.

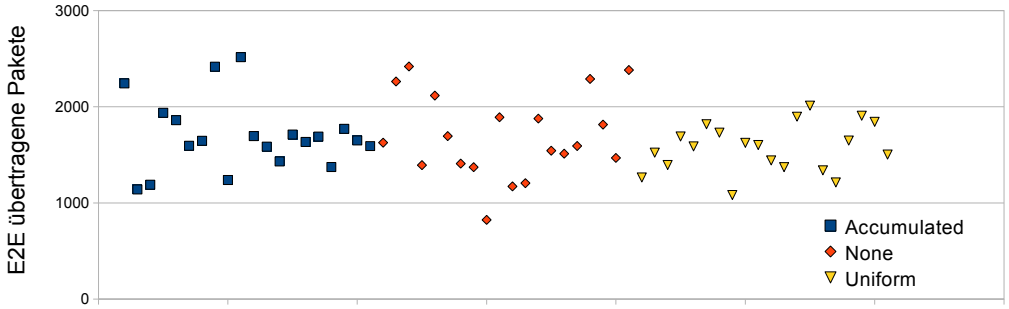
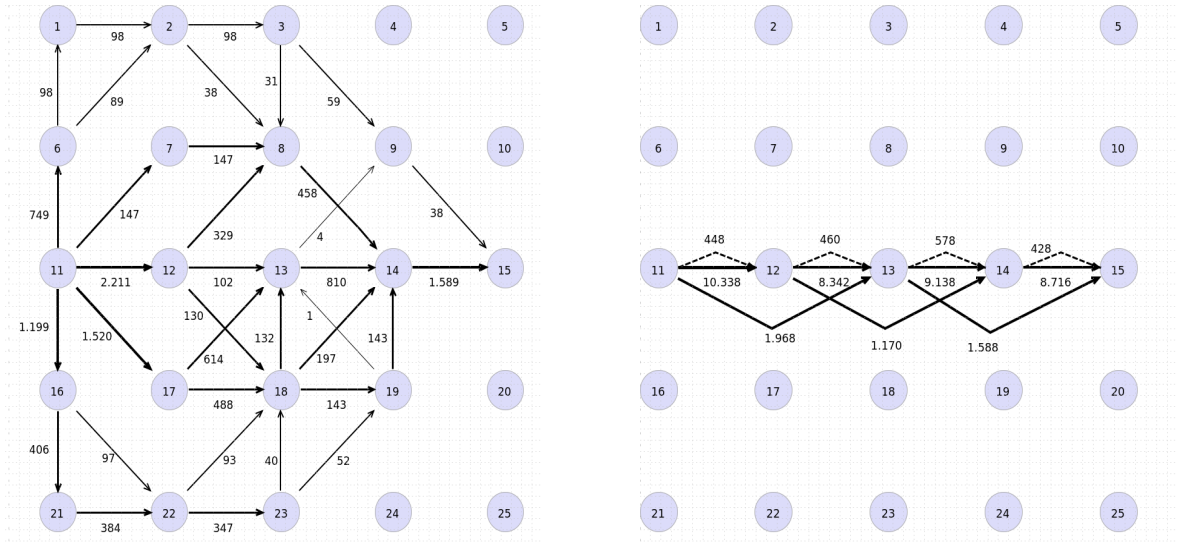


Abbildung 12 — Durchsatz mit None, Uniform und Accumulated unter Verwendung von DSR

8.4 Durchsatzgewinne durch opportunistischen Empfang und Scheduling



(a) Suboptimales Verhalten von DSR, Zahlen sind Pakete

(b) Opportunistischer Empfang und Senderatenkontrolle bei HALF, Zahlen sind Fragmente

Abbildung 13 — Scheduling bei DSR und HALF

Beim vorigen Experiment fällt auf, dass auch ohne Bitfehler HALF DSR beim Durchsatz weit überlegen ist. Dieser Effekt kann nur auf opportunistischen Empfang und Senderatenkontrolle zurückzuführen sein. Insbesondere verwirft DSR häufg Pakete nach dem ersten oder zweiten Hop, wenn keine Weiterleitung möglich scheint. Dieser Effekt wird in Abbildung 13 auf der linken Seite sichtbar. Hier werden mehr als 2200 Pakete von Knoten 11 an Knoten 12 übertragen. Von diesen werden jedoch insgesamt nur 561 an andere Knoten weitergeleitet. Der Rest wird verworfen. Die verworfenen Pakete haben jedoch bereits beim ersten Hop das Medium belegt, weshalb diese Praxis verschwenderisch ist. HALF kann mittels des START-STOP-Protokolls einen überlasteten Knoten erkennen und stellt dann das Senden am vorhergehenden Knoten ein. Daraufhin bekommt der überlastete Knoten mehr Sendemöglichkeiten und kann die aufgestauten Pakete abbauen. Die rechte Seite von Abbildung 13 zeigt dieses Verhalten. Bei jedem Knoten werden hier annähernd gleich viele Pakete empfangen. Das heißt jedoch nicht, dass jeder Knoten auch gleich viele Pakete sendet. Opportunistischer Empfang kann vielmehr genutzt werden, um Sendeoperationen an Zwischenknoten einzusparen. Im vorliegenden Beispiel wurde HALF in einer Konfiguration mit 2 Fragmenten pro Paket, sowohl bei Klartext, wie auch bei codierten Paketen verwendet. Die Zahlen an den Kanten geben

übertragene innovative, bzw. bei unterbrochenen Linien nichtinnovative, Fragmente an. Durch den opportunistischen Empfang kann eine Sendeoperation in mehreren Übertragungen resultieren. Daher kann aus den angegebenen Zahlen nicht direkt die Zahl der gesendeten Pakete abgeleitet werden. Insgesamt werden jedoch mehr als 5000 Pakete End-to-End übertragen. Dem stehen 1627 bei DSR gegenüber. Es wird ersichtlich, dass HALF insgesamt mehr Pakete sendet und die Sendeoperationen günstiger auf die Knoten verteilt, so dass weniger Pakete verworfen werden. Im DSR-Szenario werden insgesamt 34730 Pakete auf MAC-Ebene gesendet. Dem stehen 40181 Pakete bei HALF gegenüber. Diese Effekte lassen sich auch an den über mehrere Experimente zusammengetragenen Durchsatzzahlen in Tabelle 2 sowie den Abbildungen 12 und 11 ablesen.

8.5 Vergleich der HALF-Konfigurationen unter einander

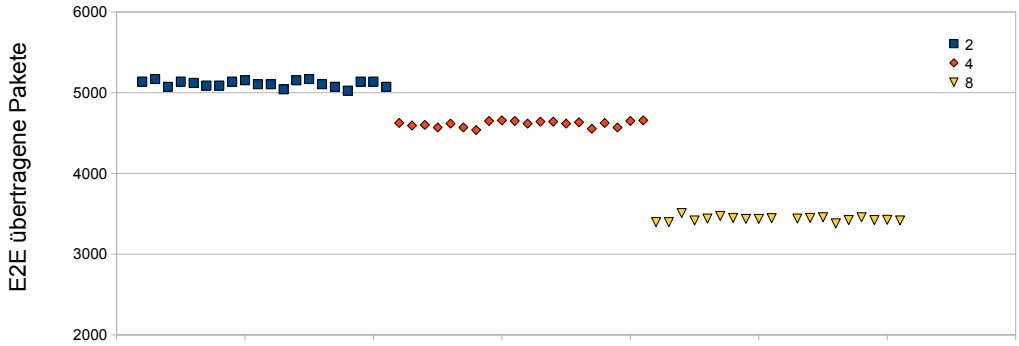


Abbildung 14 — Durchsatz nach Zahl der Fragmente pro Paket unter Verwendung des None-Modells

Um auszuloten, in welche Richtungen sich die dem HALF-System zu Grunde liegenden Konzepte noch weiter entwickeln ließen, wurden verschiedene Konfigurationen des Protokolls im selben Szenario gegeneinander getestet. Die Abbildungen 14 und 15 zeigen, dass mit mehr Fragmenten pro Paket der Durchsatz sinkt. Unabhängig vom verwendeten Bitfehlermodell ist der Durchsatz bei 4 Fragmenten pro Paket um etwa 10%, bei 8 Fragmenten pro Paket um etwa 30% niedriger als bei 2 Fragmenten pro Paket. Dies entspricht nicht den Erwartungen. Eigentlich sollte bei einer größeren Zahl von Fragmenten pro Paket ein größerer Teil jedes korrupten Paketes verwendbar bleiben und somit der Durchsatz steigen. Um den Grund für das beobachtete Verhalten zu finden, muss zunächst die Funktionsweise des START-STOP-Protokolls noch einmal verdeutlicht werden.

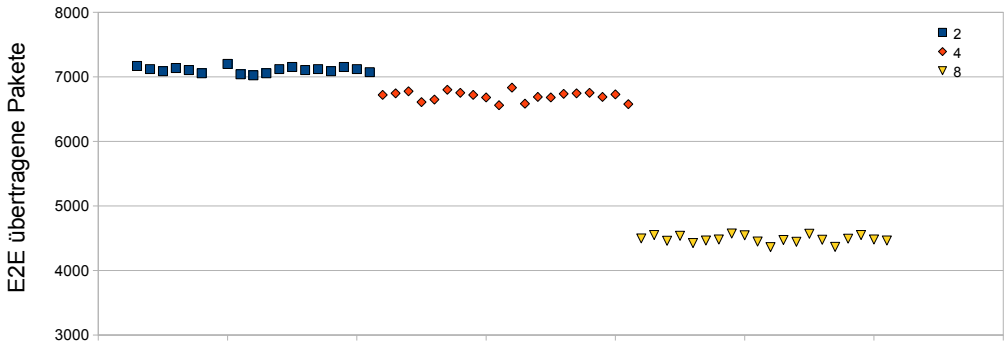


Abbildung 15 — Durchsatz nach Zahl der Fragmente pro Paket unter Verwendung des Accumulated-Modells

START und STOP werden im Unicast-Modus gesendet und sind somit gegenüber den Datenpaketen im Nachteil. Um dies auszugleichen, lässt der Encoder, nachdem er eine Anzahl Fragmente gesendet hat, eine kurze Zeit das Medium ungenutzt und wartet auf ein STOP. Diese Zahl der Fragmente bemisst sich nach der Zahl der Fragmente in einem Batch. In allen hier vorgestellten Experimenten besteht ein Batch aus maximal 32 Fragmenten. Sind nun Pakete aus je zwei Fragmenten aufgebaut, so wird der Encoder 16 Pakete senden, bevor er wartet. Ist ein Paket hingegen aus 8 Fragmenten aufgebaut, so sendet der Encoder 4 Pakete, bevor er wartet. Insgesamt wird demnach bei steigender Anzahl der Fragmente pro Paket öfter gewartet, folglich seltener gesendet und weniger übertragen. Abbildung 16 zeigt das Protokoll im Zeitablauf unter der idealisierenden Annahme, dass jedes STOP in der vorgesehenen Wartezeit ankommt. Die Konfiguration mit 2 Fragmenten verliert einen deutlich geringeren Teil der Zeit auf dem Medium mit gelb dargestellter Wartezeit und als rote und grüne Linien dargestellten START- und STOP-Paketen, als die anderen beiden. Realistisch betrachtet sieht die Bilanz natürlich noch deutlich schlechter aus, da viele START- und STOP-Pakete nicht in der Wartezeit ankommen, sondern stattdessen noch weitere, potenziell nichtinnovative Datenpakete gesendet werden, bzw. zusätzliche Zeit in Erwartung des START verschwendet wird.

Grundsätzlich fällt bereits in Tabelle 2 auf, dass die Zahl der übertragenen nichtinnovativen Fragmente nicht den Erwartungen entspricht. Die theoretischen Modelle suggerieren, dass nichtinnovative Fragmente äußerst selten auftreten, solange ein Informationsgefälle zwischen Sender und Empfänger besteht. Die Beobachtungen zeigen jedoch, dass ein signifikanter Anteil der insgesamt empfangenen Pakete bei jedem Experiment nichtinnovativ ist. Offenbar hängt auch dieser Anteil von der Zahl der Fragmente pro Paket ab. Abbildung 17 veranschaulicht dies. Dies ist ebenfalls ein Grund für die ver-

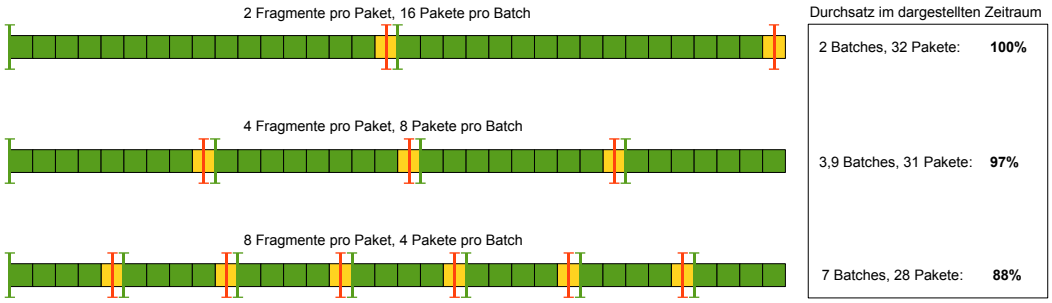


Abbildung 16 — Idealisierte Darstellung des Durchsatzverlustes durch Overhead

ringerten Durchsatzraten bei vielen Fragmenten pro Paket. Sendemöglichkeiten werden hier mit unbrauchbaren, nichtinnovativen Fragmenten verschwendet. Wenn davon ausgegangen wird, dass die Zahl der Pakete, die zwischen dem Absetzen eines STOP und dessen Empfang übertragen werden, nicht von der Fragmentierungsrate abhängt, so wird der Zusammenhang zwischen Fragmentierungsrate und Anteil der nichtinnovativen Fragmente klar. Sind Pakete aus je zwei Fragmenten und Batches aus 16 Paketen aufgebaut, und sendet der Encoder ein Paket zu viel, weil das STOP zu spät kommt, so entspricht dies 5,9% der insgesamt gesendeten Fragmente. Diese 5,9% tauchen dann in der Statistik als nichtinnovativ auf. Ist ein Paket hingegen aus 8 Fragmenten aufgebaut und somit ein Batch aus 4 Paketen, so entspricht ein überflüssiges Paket 20% nichtinnovativen Fragmenten. Die bislang unvermeidlichen verzögerten STOP-Nachrichten sorgen demnach für einen, von der Größe der Batches in Paketen abhängigen, Anteil an nichtinnovativen Fragmenten. Dass das Prinzip der Fragmentierung an sich gut ist, lässt sich ebenfalls aus Abbildung 17 ablesen. Mit steigender Anzahl der Fragmente pro Paket steigt auch der Anteil der Fragmente die opportunistisch empfangen werden. Diese Tatsache ist ein Indiz dafür, dass mit steigender Fragmentierung der Pakete ein höherer Anteil der auf MAC-Ebene empfangenen Pakete von HALF verwendet werden kann. Leider manifestiert sich dies nicht im Durchsatz, da der selbe Effekt dazu führt, dass die STOP-Pakete seltener in der vorgesehenen Wartezeit gesendet werden. Hat ein Knoten einen Teil eines Batches bereits opportunistisch empfangen, so wird er, wenn der direkte Vorgängerknoten sendet, nur noch einen Teil des Batches benötigen und folglich das STOP früher als erwartet senden. Dieses STOP kann sich dann nur schwer gegen die Broadcast-Pakete des Vorgängerknotens durchsetzen, was sich in einer zusätzlich erhöhten Zahl an nichtinnovativen Fragmenten niederschlägt. Letzteres Phänomen tritt natürlich nur auf, wenn die bitfeh-

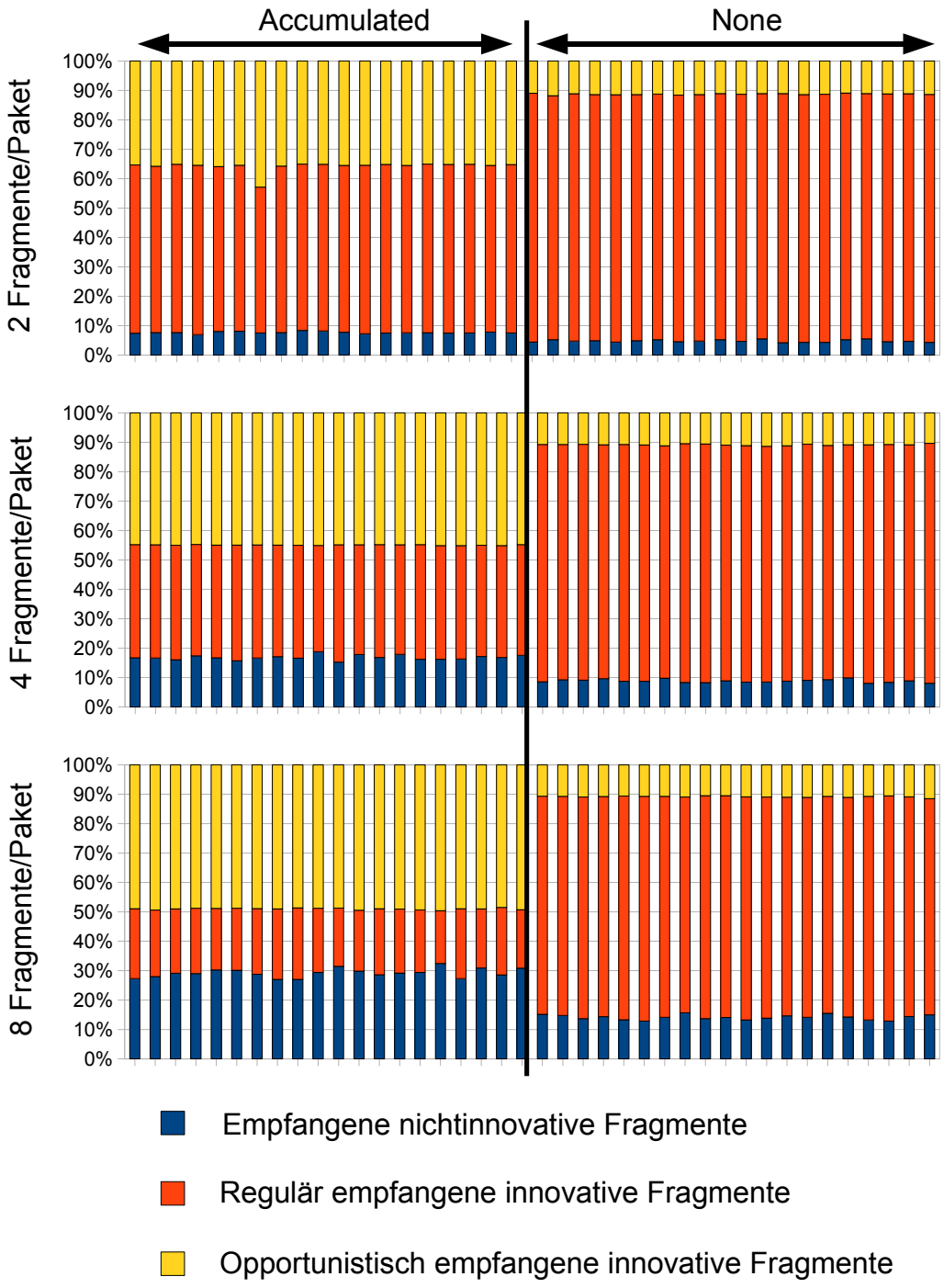


Abbildung 17 — Charakterisierung der empfangenen Fragmente nach Bitfehlermodell und Fragmentierungsgrad der Pakete

lerbehafteten Pakete auch wirklich ausgeliefert werden. Daraus wiederum ergeben sich die Unterschiede zwischen None und Accumulated in Abbildung 17. All diese negativen Effekte sind allerdings nicht in der Funktionsweise von HALF selbst begründet, sondern vielmehr eine Folge der zur Verfügung stehenden Umgebung und des Versuchsaufbaus.

8.6 Latenz

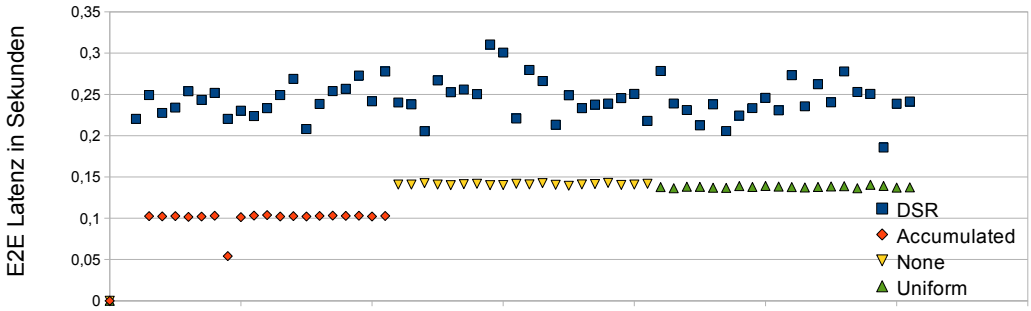


Abbildung 18 — Mittlere Latenz bei verschiedenen Bitfehlermodellen mit DSR und HALF

Für die Messung der Latenz werden zunächst nur die Experimente herangezogen, die auch als Basis für die Durchsatzmessung dienten. Es wird erwartet, dass HALF hier deutlich schlechter abschneidet als DSR. Dies begründet sich im Wesentlichen darin, dass HALF vor der Auslieferung eines Paketes warten muss, bis der gesamte Batch decodierbar ist, während DSR jedes eintreffende Paket sofort ausliefern kann. Abbildung 18 bestätigt dies jedoch nicht. Zwar liegen auch hier wieder starke zufallsabhängige Schwankungen beim Experiment mit DSR vor, erstauliecher weise schneidet HALF jedoch bei keinem einzigen Experiment schlechter ab als DSR. Die Vermutung, dass HALF tatsächlich generell eine geringere Latenz erzielt, liegt demnach nahe. Natürlich können daraus noch keine konkreten Zahlen destilliert werden. Offensichtlich liegen hier zusätzliche bisher unberücksichtigte Effekte vor, die das Ergebnis zu Gunsten von HALF und zu Ungunsten von DSR beeinflussen. Bei Betrachtung von Tabelle 2 fällt zudem auf, dass sich die Latenz bei Verwendung von HALF noch weiter verringert, wenn mehr Fragmente pro Paket verwendet werden. Andererseits steigt die Latenz, wenn mehrere parallele Ströme vorliegen. Diese beiden Effekte sind leicht erklärbar. Bei Verwendung von mehr Fragmenten pro Paket sinkt im vorliegenden Experimentaufbau die Zahl der Pakete pro Batch (vgl. Abschnitt 8.5). Aus diesem Grund kann der Zielknoten mit steigendem Fragementierungsgrad aus weniger Paketen einen vollständigen Batch decodieren und der

durch das Warten auf zusätzliche Pakete induzierte Anteil der Latenz sinkt. Wenn hingegen mehr gleichzeitige Ströme vorliegen, sind zu jeder Zeit entsprechend mehr Batches aktiv. Die Gesamtzahl der möglichen Sendeoperationen im gesamten Netz steigt jedoch nicht wesentlich, da die Ströme untereinander interferieren. Nachdem auch die Zahl der zur Übertragung eines Batches nötigen Sendeoperationen nicht sinkt, dauert es für jeden Batch länger, bis er übertragen ist. Dieser Effekt schlägt sich in der beobachtbaren Latenz nieder.

9 Mögliche weitere Fragestellungen

9.1 Scheduling für Multipath-Routen

Die von Chachulski u. a. (2006) und Radunovic u. a. (2007b) bekannten Scheduling-Ansätze lassen sich mit RUBS kombinieren. So kann ein Scheduling auf einer ungeordneten Multipath-Route mit Hop-By-Hop STOP/START entworfen werden. Die Funktionsweise von RUBS bleibt im Wesentlichen erhalten, bis auf folgende Erweiterungen:

1. Der optimale Fluss durch einen Knoten, den MORE (oder ein anderer Algorithmus) aus der Route und Netzwerktopologie ausrechnet wird als Anteil jedes Batches, der durch den entsprechenden Knoten übertragen werden muss, interpretiert. MORE beispielsweise sagt mit den x_{ij} Werten für jeden Knoten, wie viele effektive Pakete - ohne Übertragungsverlust - er, pro an der Quelle vorhandenem Paket, an seine Nachfolger übertragen soll. Das ist ein Wert zwischen 0 und 1 und ließe sich analog als Anteil eines an der Quelle vorhandenen Batches verstehen.

2. Ein Knoten B sendet STOP an einen anderen Knoten A wenn
   - B noch kein STOP für den entsprechenden Batch an A gesendet hat UND
   - das letzte an B von A empfangene Paket als *stoppbar* ausgewiesen ist UND
   - die Zahl der verarbeiteten Zeilen der Matrix bei B, in Relation zu ihrer Gesamtgröße, größer als der ausgewiesene Anteil des Batches ist, der durch B gehen soll.

3. Ein Knoten hört erst auf zu senden, wenn er von allen Empfangskandidaten ein STOP bekommen hat. START wird analog nicht nur an einen Knoten gesendet, sondern an jeden Kandidaten, von dem Pakete empfangen werden können und von dem noch keine Pakete des neuen Batches gesehen wurden. Ein Knoten fängt beim ersten empfangenen START an zu senden.

4. Ein Knoten markiert ein Paket als stoppbar, wenn er mindestens den für ihn vorgesehenen Anteil des Batches gesendet hat.

Auf diese Art kann es passieren, dass der Zielknoten eine nicht lösbare Matrix erhält, indem Subräume der Matrix unterwegs verloren gegangen sind und dafür andere redundant übertragen wurden. Angesichts der zufälligen Linearkombinationen an jedem Knoten sollte das selten auftreten, muss aber als Spezialfall behandelt werden. Eventuell ist es günstig, jedes gesendete Fragment eindeutig zu identifizieren und in den STOP-Paketen anzuzeigen, welche Fragmente empfangen wurden. Auf diese Art bekommt der sendende Knoten mit, wenn empfangende Knoten redundante Anteile des Batches übertragen und kann entsprechend reagieren.

Nachdem jeder Knoten sich zu jedem Zeitpunkt mit höchstens einem Batch aktiv befasst, ist die Zahl der Pakete auf der Route begrenzt. Nachdem Pakete von hinten beliebig nachgesendet werden, ist das Medium ausgelastet, solange kein Stau auftritt.

Es stellt sich die Frage, ob sicher ist, dass konzeptionell keine unnötigen Sendeoperationen stattfinden. Dies wird mit Hilfe der festgelegten Anteile sichergestellt. Es kann nun passieren, dass ein Anteil eines Batches auf einem ungünstigen Pfad sehr lange unterwegs ist. Wenn ein Knoten auf einem günstigen Pfad gleichzeitig mehr als seinen vorgeschriebenen Anteil kennt, so kann dieser den Zielknoten deutlich früher erreichen. Der Zielknoten sendet nur auf nichtinnovative Pakete hin STOP. Deshalb wird der günstige Zwischenknoten seinen gesamten - zu hohen - Anteil übertragen. Das führt dazu, dass der langsamere Anteil faktisch überflüssig ist und somit unnötige Sendeoperationen stattfinden. Um dies zu verhindern, könnten die Linearkombinationen bei den Zwischenknoten künstlich auf den vorgesehenen Anteil einschränkt werden. So werden durch nichtinnovative Pakete schnellere STOPs provoziert und die Empfänger zum Warten auf langsamere Anteile gezwungen.

9.2 Weitere Optimierungsmöglichkeiten

Bei diesem Schema ist der Overhead durch START- und STOP-Pakete deutlich höher als beim grundlegenden RUBS-Protokoll. Jeder Knoten muss jedem seiner potenziellen Vorgänger für jeden Batch ein START und ein STOP senden, während bei der Grundfunktionalität jeder Knoten nur ein START und ein STOP pro Batch sendet. Eine Optimierungsmöglichkeit besteht demnach in der Reduktion dieses Overheads. Sowohl für STOP als auch für START besteht entsprechendes Potenzial.

STOP-Pakete werden ebenfalls opportunistisch empfangen. Daraus ergibt sich, dass

sendende Knoten bereits wissen können, dass bestimmte Empfangskandidaten keine zusätzlichen Pakete mehr brauchen. Folglich müssen diese Empfangskandidaten kein erneutes STOP mehr senden. Um das zu erreichen kann das *stoppbar*-Flag für jeden Kandidaten einzeln gesetzt oder eben nicht gesetzt werden.

Ebenso verhält es sich mit START. Bekommt ein wartender Knoten das START für einen anderen Knoten mit und ist der Sender des START ein Empfangskandidat, kann der Knoten anfangen zu senden, ohne ein eigenes START abzuwarten. Bekommt ein Knoten Pakete für den zu startenden Batch, so muss er deren Sender nicht mehr explizit STARTen.

10 Ausblick

In der vorliegenden Arbeit wurde die Verwendung von fehlerhaften Paketen unter Voraussetzung des Anycast-Primitivs als grundsätzliche Form der Nutzung des Netzwerks erkundet. Dies erwies sich als sehr günstig, was die Simulationsergebnisse zeigen. Durch Nutzung der korrekt übertragenen Abschnitte fehlerhafter Pakete konnte eine Durchsatzsteigerung von bis zu 40% erreicht werden. Diese Zahl könnte in anderen Umgebungen als der simulierten höher ausfallen, jedoch ist keine dramatische Steigerung zu erwarten. Allerdings drängt sich ein weiteres, gewichtiges Problem in den Vordergrund: die Senderatenkontrolle oder das Scheduling. Durch ein sehr einfaches Scheduling-Verfahren (RUBS) konnte bereits ohne Verwendung der fehlerhaften Pakete eine Vervielfachung des Durchsatzes gegenüber DSR, erzielt werden. Dieser Durchsatzvorsprung konnte durch das fehlende Scheduling bei DSR erklärt werden. Es wurde gezeigt, dass die Übertragungslatenz bei Verwendung des neu entwickelten HALF-Protokolls, selbst ohne Nutzung defekter Pakete, in bestimmten Fällen deutlich niedriger ist, als bei DSR. DSR hingegen wurde bisher, wie alle einfachen store-and-forward Routingprotokolle in diesem Aspekt als günstig angesehen. Es bleibt zu betonenen, dass das allgemeine Design des HALF-Protokolls keinerlei Paketverlust während der Übertragung erlaubt. Zudem sei daran erinnert, dass speziell das TCP-Protokoll gerade unter hoher Latenz und Paketverlust leidet, was allgemein als Grund für die schlechte Leistung dieses Protokolls in drahtlosen Netzwerkren betrachtet wird.

Als grundlegende Schlüsse aus dieser Arbeit kann also Folgendes festgehalten werden. Erstens muss das Schedulingproblem, anders als in den meisten bisherigen Werken, als erstrangige Aufgabe angesehen werden. Ein günstiges Scheduling kann offenbar deutlich höhere Durchsatzgewinne erzielen, als viele andere Ansätze. Zweitens ist somit gezeigt,

dass Paketverlust und hohe Latenz keine inhärenten Eigenschaften des drahtlosen Mediums sind, sondern sehr wohl durch adäquate Maßnahmen in gegenseitiger Vereinbarkeit umgangen werden können. Es ist folglich davon auszugehen, dass hier ein Ansatz vorliegt, um die Leistung von TCP in drahtlosen Netzwerken zu verbessern. Dieser Ansatz behielte die Protokoll-Schichtung des OSI-Modells weitgehend bei.

Abbildungsverzeichnis

Akronyme

acknowledgement **(ACK)**

Empfangsbestätigung für übertragene Pakete 5, 8, 13, 15, 22–24, 39

access point **(AP)**

Zentraler Zugangsknoten in Infrastruktur-WLAN-Netzen 8, 9, 14, 45, 47, 49

automatic repeat request **(ARQ)**

Automatische Neuübertragung von Paketen, falls das ACK ausbleibt 2, 5, 6, 13, 14, 22, 27, 41

additive white gaussian noise **(AWGN)**

Auf der Annahme von additivem Hintergrundrauschen beruhendes Kanalmodell 12

bit error rate **(BER)**

Rate der fahlerhaften Bits im Datenstrom 9

code division multiple access **(CDMA)**

Aufteilung des Medienzugriffs mittels Codierung im gemeinsamen Frequenzbereich 8, 14

continuous network coding **(CNC)**

Verfahren zur Codierung auf Symbolebene 24

cyclic reduncancy check **(CRC)**

Verfahren zur Generierung von Prüfsummen 8, 42, 46

carrier sense multiple access **(CSMA)**

Aufteilung des Medienzugriffs mittels Trägerprüfung 8

carrier sense multiple access with collision avoidence **(CSMACA)**

CSMA-Verfahren, das probabilistisch versucht, Kollisionen zu vermeiden 8

distributed coordination function (DCF)

Kombination von ARQ und CSMA zur Regelung des Medienzugriffs in IEEE 802.11 8, 25, 27, 28, 36, 37

diststributed hash table (DHT)

Verteilte Datenstruktur zur Speicherung von Information auf mehreren Knoten im Netzwerk 49, 56

dynamic source routing (DSR)

Reaktives Routingprotokoll 2, 42, 46, 49, 56, 58–60, 62, 67, 70

extpected transmissions (ETX)

Erwartete Anzal von Sendeoperationen zur Übertragung eines Paketes 9, 10, 25, 28, 33

extreme opportunistic routing (ExOR)

Opportunistisches Routingprotokoll 15, 22, 25

forward error correction (FEC)

Verfahren zum bitweisen Ausgleich von Übertragungsfehlern mittels Redundanz 2, 5, 8, 13, 14, 42

hybrid ARQ with limited fragmentation (HALF)

Kombiniertes Coding-, Scheduling und Fragmentierungsverfahren 40, 42, 45–47, 49, 56, 58–60, 62–64, 67, 70

hybrid ARQ (HARQ)

Form des ARQ, das nicht jedes Paket einzeln bestätigt, sondern ein komplexeres Protokoll verwendet 14, 39–41, 44, 45

low density parity check code (LDPC)

Fehlerkorrekturcode, z.B. für FEC 24

local neighbour discovery (LND)

Verfahren zur Bewertung der Verbindungen zwischen Knoten im Netzwerk 30

line of sight (LOS)

Sichtkontakt 8, 56

layered packet coded cooperative system (LPCCS)

Auf inkrementellen FEC-Phasen beruhendes Codingschema 14

multiuser diversity forwarding (MDF)

Weiterentwicklung von SDF 16

multi radio diversity (MRD)

Nutzung mehrerer Radioempfänger zur Decodierung des selben Frames 14

no line of sight (NLOS)

Fehlender Sichtkontakt 8

packet error rate (PER)

Rate der fehlerhaften Pakete im Datenstrom 9, 11

random linear network codes (RLNC)

Network Coding mittels zufälliger Koeffizienten aus endlichen Feldern 2, 40

Reed-Solomon (RS)

Fehlerkorrekturcode, z.B. für FEC 13

received signal strength indication (RSSI)

Gerätabhängiger Indikator für die Stärke eines empfangenen Signals 55

really unfair batch scheduling (RUBS)

Einfaches Schedulingprotokoll, Teil von HALF 42, 44, 68–70

selection diversity forwarding (SDF)

Opportunistisches Routingprotokoll 15, 16

signal to interference and noise ratio (SINR)

SNR unter spezieller Betrachtung der Interferenz von anderen Teilnehmern 34, 55

signal to noise ratio **(SNR)**

Verhältnis von Signalstärke zu Hintergrundrauschen 7, 11, 14, 46, 47

station **(STA)**

Mit einem AP assoziiertes Gerät 8, 9, 14, 45

Literatur

802.16e 2005

IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2:Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands. New York, NY, USA : The Institute of Electrical and Electronics Engineers, Inc., 2005

802.11 2007

IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. New York, NY, USA : The Institute of Electrical and Electronics Engineers, Inc., 2007

Aguayo u. a. 2004

Aguayo, Daniel ; Bicket, John ; Biswas, Sanjit ; Judd, Glenn ; Morris, Robert: Link-level measurements from an 802.11b mesh network. In: *SIGCOMM Comput. Commun. Rev.* 34 (2004), Nr. 4, S. 121–132

Ahlswede u. a. 2000

Ahlswede, A. ; Cai, N. ; Li, S. Y. R. ; Yeung, R. Y.: Network Information Flow. In: *IEEE Transactions on Information Theory* 46 (2000), Nr. 4, S. 1204–1216

Bianchi 2000

Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. In: *IEEE Journal on Selected Areas in Communications* 18 (2000), Nr. 3, S. 535–547

Biswas und Morris 2004

Biswas, Sanjit ; Morris, Robert: Opportunistic routing in multi-hop wireless networks. In: *SIGCOMM Comput. Commun. Rev.* 34 (2004), Nr. 1, S. 69–74

Biswas und Morris 2005

Biswas, Sanjit ; Morris, Robert: ExOR: opportunistic multi-hop routing for wireless networks. In: *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM Press, 2005, S. 133–144

Chachulski u. a. 2006

Chachulski, Szymon ; Jennings, Michael ; Katti, Sachin ; Katabi, Dina: MORE: A Network Coding Approach to Opportunistic Routing / Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory. URL `http://dspace.mit.edu/handle/1721.1/33230`, 2006. – Forschungsbericht

Chachulski u. a. 2007a

Chachulski, Szymon ; Jennings, Michael ; Katti, Sachin ; Katabi, Dina: Trading Structure for Randomness in Wireless Opportunistic Routing / Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory. URL `http://dspace.mit.edu/handle/1721.1/36345`, 2007. – Forschungsbericht

Chachulski u. a. 2007b

Chachulski, Szymon ; Jennings, Michael ; Katti, Sachin ; Katabi, Dina: Trading structure for randomness in wireless opportunistic routing. In: *SIGCOMM Comput. Commun. Rev.* 37 (2007), Nr. 4, S. 169–180

Chebrolu u. a. 2006

Chebrolu, Kameswari ; Raman, Bhaskaran ; Sen, Sayandeep: Long-distance 802.11b links: performance measurements and experience. In: *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM Press, 2006, S. 74–85

Choudhury und Vaidya 2004

Choudhury, Romit R. ; Vaidya, Nitin H.: MAC-layer anycasting in ad hoc networks. In: *SIGCOMM Comput. Commun. Rev.* 34 (2004), Nr. 1, S. 75–80

Cover und Thomas 1991

Cover, T. M. ; Thomas, J. A.: *Elements of Information Theory*. New York : Wiley-Interscience, 1991

De Couto u. a. 2003

De Couto, Douglas S. J. ; Aguayo, Daniel ; Bicket, John ; Morris, Robert: A high-throughput path metric for multi-hop wireless routing. In: *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, ACM Press, 2003, S. 134–146

Deb u. a. 2006

Deb, Supratim ; Médard, Muriel ; Choute, Clifford: Algebraic gossip: a network coding approach to optimal multiple rumor mongering. In: *IEEE Transactions on Information Theory* 42 (2006), Nr. 6, S. 2486–2507

Dubois-Ferriere u. a. 2005

Dubois-Ferriere, Henri ; Estrin, Deborah ; Vetterli, Martin: Packet combining in sensor networks. In: *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. New York, NY, USA : ACM Press, 2005, S. 102–115

Elias u. a. 1956

Elias, P. ; Feinstein, A. ; Shannon, C. E.: Note on maximal flow through a network. In: *IRE Trans. on Information Theory* 2 (1956), S. 117–119

Ford und Fulkerson 1956

Ford, J. L. R. ; Fulkerson, D. R.: Maximal flow through a network. In: *Canadian Journal of Math.* 8 (1956), S. 399–404

Fragouli u. a. 2007

Fragouli, C. ; Lun, D. ; Medard, M. ; Pakzad, P.: On Feedback for Network Coding. In: *CISS '07. 41st Annual Conference on Information Sciences and Systems*. Baltimore, MD, USA, 2007, S. 248–252

Gilbert 1960

Gilbert, E. N.: capacity of a bursty noise channel. In: *Bell Sys. Tech. J.* 39 (1960), S. 1253–1265

Goldsmith 2005

Goldsmith, Andrea: *Wireless Communications*. New York, NY, USA : Cambridge University Press, 2005

Ho u. a. 2006

Ho, Tracey ; Médard, Muriel ; Koetter, Ralf ; Karger, David R. ; Effros, Michelle ; Shi, Jun ; Leong, Ben: A random linear network coding approach to multicast. In: *IEEE Transactions on Information Theory* 52 (2006), Nr. 10, S. 4413–4430

Jamieson und Balakrishnan 2007

Jamieson, Kyle ; Balakrishnan, Hari: PPR: Partial Packet Recovery for Wireless Networks / Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory. URL `http://dspace.mit.edu/handle/1721.1/35889`, 2007. – Forschungsbericht

Johnson und Maltz 1996

Johnson, David B. ; Maltz, David A.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: Imielinski (Hrsg.) ; Korth (Hrsg.): *Mobile Computing* Bd. 353. Kluwer Academic Publishers, 1996

Katti u. a. 2007

Katti, Sachin ; Gollakota, Shyamnath ; Katabi, Dina: Embracing Wireless Interference: Analog Network Coding / Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory. URL `http://dspace.mit.edu/handle/1721.1/36343`, 2007. – Forschungsbericht

Katti und Katabi 2007

Katti, Sachin ; Katabi, Dina: MIXIT: The Network Meets the Wireless Channel / Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory. URL `http://dspace.mit.edu/handle/1721.1/38871`, 2007. – Forschungsbericht

Katti u. a. 2006

Katti, Sachin ; Rahul, Hariharan ; Hu, Wenjun ; Katabi, Dina ; Médard, Muriel ; Crowcroft, Jon: XORs in the air: practical wireless network coding. In: *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM Press, 2006, S. 243–254

Koetter und Médard 2003

KOETTER, R. ; MÉDARD, M.: Beyond Routing: An Algebraic Approach to Network Coding. In: *IEEE/ACM Transactions on Networking* 11 (2003), S. 782–796

Kong u. a. 2008

KONG, Zhenning ; ALY, Salah A. ; SOLJANIN, Emina ; YEH, Edmund M. ; KLAPPENECKER, Andreas: Network Coding Capacity of Random Wireless Networks under a SINR Model. In: *CoRR* (2008). – URL `http://arxiv.org/abs/0804.4284`

Larsson und Johansson 2005

LARSSON, P. ; JOHANSSON, N.: Multiuser diversity forwarding in multihop packet radio networks. In: *IEEE Wireless Communications and Networking Conference* Bd. 4, 2005, S. 2188–2194

Larsson 2001

LARSSON, Peter: Selection diversity forwarding in a multihop packet radio network with fading channel and capture. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 5 (2001), Nr. 4, S. 47–54

Lee u. a. 2007

LEE, Jeongkeun ; LEE, Sung-Ju ; KIM, Wonho ; JO, Daehyung ; KWON, Taekyoung ; CHOI, Yanghee: *Understanding Inteference and Carrier Sensing in Wireless Mesh Networks*. 2007. – URL `http://www.hpl.hp.co.uk/personal/Sung-Ju_Lee/abstracts/papers/commmag2007.pdf`. – akzeptiert zur Veröffentlichung in: IEEE Communications Magazine

Levorato u. a. 2006

LEVORATO, M. ; TOMASIN, S. ; CASARI, P. ; ZORZI, M.: Analysis of Spatial Multiplexing for Cross-Layer Design of MIMO Ad Hoc Networks. In: *IEEE 63rd Vehicular Technology Conference* Bd. 3, 2006, S. 1146–1150

Li u. a. 2003

LI, Shuo-Yen R. ; YEUNG, Raymond W. ; CAI, Ning: Linear Network Coding. In: *IEEE Transactions on Information Theory* 49 (2003), Nr. 2, S. 371–381

Li und Li 2004

Li, Zongpeng ; Li, Baochun: Network Coding: The Case of Multiple Unicast Sessions. In: *Proceedings of 42nd Allerton conference on Communications*. Monticello, IL, USA, 2004

Lun u. a. 2006

Lun, Desmond S. ; Medard, Muriel ; Koetter, Ralf ; Effros, Michelle: On Coding for Reliable Communication over Packet Networks. In: *International Symposium on Information Theory, 2005. ISIT 2005. Proceedings*, 2006, S. 1848–1852

Miu u. a. 2005

Miu, Allen ; Balakrishnan, Hari ; Koksal, Can E.: Improving loss resilience with multi-radio diversity in wireless networks. In: *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM Press, 2005, S. 16–30

Pu u. a. 2008

Pu, Wei ; Luo, Chon ; Li, Shipeng ; Chen, Chang W.: Continuous Network Coding in Wireless Relay Networks. In: *INFOCOM 2008. 27th IEEE International Conference on Computer Communications. IEEE*, 2008, S. 1535–1542

Radunovic u. a. 2007a

Radunovic, Bozidar ; Gkantsidis, Christos ; Key, Peter ; Gheorghiu, Steluta ; Hu, Wenjun ; Rodriguez, Pablo: Multipath Code Casting for Wireless Mesh Networks / Microsoft Research. URL `http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report\&#38;id=1310`, 2007. – Forschungsbericht

Radunovic u. a. 2007b

Radunovic, Bozidar ; Gkantsidis, Christos ; Key, Peter ; Rodriguez, Pablo ; Hu, Wenjun: An optimization framework for practical multipath routing in wireless mesh networks / Microsoft Research. URL `http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report\&#38;id=1325`, 2007. – Forschungsbericht

Rayanchu u. a. 2008

Rayanchu, Shravan ; Mishra, Aunesh ; Agrawal, Dheeraj ; Saha, Sharad ; Banerjee, Suman: Diagnosing Wireless Packet Losses in 802.11: Separating Collision from Weak Signal. In: *INFOCOM 2008. 27th IEEE International Conference on Computer Communications. IEEE*, 2008, S. 735–743

Riemann und Winstein 2005

Riemann, Reina ; Winstein, Keith: Improving 802.11 Range with Forward Error Correction / Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory. URL `http://dspace.mit.edu/handle/1721.1/30524`, 2005. – Forschungsbericht

Sklar 2001

Sklar, Bernard: *Digital Communications.* 2. Ausgabe. New Jersey, US : Prentice Hall PTR, 2001

Souryal u. a. 2006

Souryal, M. R. ; Klein-Berndt, L. ; Miller, L. E. ; Moayeri, N.: Link assessment in an indoor 802.11 network. In: *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE* Bd. 3, 2006, S. 1402–1407

Subramanian u. a. 2007

Subramanian, Vijaynarayanan ; Kalyanaraman, Shivkumar ; Ramakrishnan, K. K.: Hybrid Packet FEC and Retransmission-based Erasure Recovery Mechanisms for Lossy Networks: Analysis and Design. In: *Wireless Systems: Advanced Research and Development (WISARD)*, 2007, S. 1–8

Sundararajan u. a. 2008

Sundararajan, Jay K. ; Shah, Devavrat ; Médard, Muriel: ARQ for Network Coding. In: *CoRR* (2008). – URL `http://arxiv.org/abs/0802.1754v1`

Tanenbaum 2002

Tanenbaum, Andrew S.: *Computer Networks, Fourth Edition.* New Jersey, US : Prentice Hall PTR, 2002

Tse und Viswanath 2005

Tse, David ; Viswanath, Pramod: *Fundamentals of Wireless Communication*. New York, NY, USA : Cambridge University Press, 2005

Valenti und Correal 2003

Valenti, M. ; Correal, N.: Exploiting macrodiversity in dense multihop networks and relay channels. In: *IEEE Wireless Communications and Networking. WCNC* Bd. 3, 2003, S. 1877–1882

Willig u. a. 2002

Willig, A. ; Kubisch, M. ; Hoene, C. ; Wolisz, A.: Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. In: *IEEE Transactions on Industrial Electronics* 49 (2002), Nr. 6, S. 1265–1282

Zhong und Nelakuditi 2007

Zhong, Zifei ; Nelakuditi, Srihari: On the Efficacy of Opportunistic Routing. In: *Fourth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2007, S. 441–450

Zubow u. a. 2006

Zubow, A. ; Kurth, M. ; Redlich, J. P.: Multi-channel opportunistic routing in multi-hop wireless networks. URL `http://edoc.hu-berlin.de/series/informatik-berichte/204/PDF/204.pdf`, 2006. – Forschungsbericht

Erklärung

Hiermit erkläre ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel, benutzt habe.

16. Juni 2008, Ulf Hermann

Ich bin damit einverstanden, dass die vorliegende Arbeit in der Bibliothek ausgelegt wird.

16. Juni 2008, Ulf Hermann